Feature Selection and Improving Classification Performance for Malware Detection

Chia Tien Dan Lo, Ordóñez Pablo, Cepeda Carlos.
Department of Computer Science
Kennesaw State University
Georgia, USA
E-mail: dlo2@kennesaw.edu; {pordonez; ccepedam}@students.kennesaw.edu

Abstract— **After analyzing the advance of technology, it is clear that use of the Internet, computers, smart phones and tablets has become ubiquitous and therefore, the creation and proliferation of cyber threats and attacks has grown exponentially. Consequently, Anti-Virus companies and researchers have developed new approaches for dealing with discovering and classifying malware. Among these, machine learning and Big Data technologies have been used for feature extraction, detection, and clustering of cyber threats. In this paper, we created and analyzed a dataset of malware and clean files (goodware) from the static and dynamic features provided by the online framework VirusTotal. The purpose is to select the smallest number of features that keep the classification accuracy as high as possible given that the training execution time increase in polynomial time with respect to the number of features. In this research, we found that "9" features are enough to distinguish malware from "goodware" files within an accuracy of 99.60%. Selecting the most representative features for malware detection relies on the possibility of creating an embedded program that monitors the processes executed by the OS looking for the characteristics that match malware behavior. Thus, feature selection was made taking the most important features that keep the accuracy high and allows the creation of monitoring malware detection programs with a low overhead cost. In addition, classification algorithms such as Random Forest (RF), Support Vector Machine (SVM) and Neural Networks (NN) were used in a novel combination that not only showed an increase in accuracy, but also in the training speed from hours to just minutes.**

Keywords—**Malware detection; Machine Learning; Big Data Analytics; Feature selection; Random Forest; Support Vector Machine; Neural Networks; improving accuracy.**

I. Introduction (Big Data, Machine Learning and Security)

Big Data Analytics refers to the process of searching, capturing, storing and processing the data sets in order to extract meaningful information, discovering patterns and relations, marketing and customer trends, and discovering abnormal behaviors. On the other hand, security and privacy are two main topics nowadays, particularly because of growing of Internet and the Big Data Era, which have magnified these trends. Also, Big Data infrastructures are easily accessible to different organizations or individuals across multiple cloud infrastructures [2]. In addition, it is known that The Internet has been an infrastructure which has enabled the expansion of cyber treats and attacks; in fact, the growth of just Malware attacks has been exponential [3], so antivirus companies are no longer able to analyze and create signatures for all these new attacks. However, Big Data Analytics and machine learning could be part of the solution. With the power of extract patterns and dealing with huge amounts of data, platforms for extracting, analyzing and finding patterns is possible to identify normal and abnormal behaviors and then prevents consequences from malware attacks. Therefore Big Data tools for running machine learning algorithms (let say SPARK and Torch), and tools for extracting malware information (VirusTotal) were used to create the Dataset and analyze the information.

II. Malware Overview

There are different types of cyber-attacks such as phishing, botnets, searching poisoning, denial of services, spamming, and Malware. As mentioned before, cyber-attack growth has increased exponentially and has compromise computers, steal information and damage critical structures, producing significant losses, averaging $345.000 per incident [3]. In recent years, Malware proliferation increased not just by the Internet growth but also because developing new malicious programs is becoming easier; in fact, "more than 317 million new pieces of Malware were created last year" (2014) [10], which means that "nearly one million new threats were released each day". As a consequence, Antivirus Companies – AV, are no longer able to process all of them. First, it is not possible to capture all Malware on the network, and even more, it is not possible to generate signatures for all these files-programs collected by the AV companies in a reasonable time due to the huge number of them. Finally, this is also important to mention that along the news and more sophisticated Anti-Virus programs, cyber criminals also have increased the complexity of malicious programs. There are techniques such obfuscation/Metamorphism (code substitution, code reordering, register swapping), Noise Insertion (Garbage instructions, unused functions) and Packing (Cryptors, Protectors, Packers) which decrease the detection rate from the Anti-Virus programs.

III. Big Data Tools for Malware Analysis

Malware is a trend that tends to increase and will remain as the "greatest security threat faced by computer users" [3]. Thus, the necessity for automatic malware detection and classification has allow the creation of tool as CWSandbox, Cuckoo Sandbox, Norman Sand-box, and hybrid platforms as ThreatExpert, ANUBIS, VirusTotal, Metascan ®, Payload Security – VxStream and Malwr. These systems execute the suspicious malware files on a virtual or controlled environment in order to monitor and extract static and behavioral information, which is used for analyses, detection and classification. In this article, we analyzed the static and dynamic data extracted from malicious and "good" files in order to predict classify a test dataset. Because the size of the dataset is relatively big – 9448 cases by 682.936 feature vectors (see section VII to detail information of the Dataset) - for running machine learning algorithms, Big Data tools are required; in our case, SPARK was used for dealing with the whole dataset for making the initial feature selection and then R Cran and TORCH frameworks were used for other stages of feature selection and classification.

IV. Malware Analysis

Two approaches are used to analyze Malware files, Static and Dynamic Analysis. Static Analysis extracts features directly from the byte-code or disassembled instructions, so it is not required to run the program [4]. "Static Analysis includes string signature, byte-sequence n-grams, syntactic library call, control flow graph and opcode (operational code) frequency distribution" [4]. The advantage of this analysis is that it could follow all possible execution paths and it is less resource intensive, but it is sensible to packing techniques, encryption, compression, garbage code insertion and code permutation [4], so malware detection based on static features can be bypassed by obfuscation methods.

Dynamic Analysis is executed on a virtual or insulated environment in order to monitor the malware behavior (file system, registry monitoring, process monitoring, network monitoring, system change detection, function call monitoring, function parameter analysis, information flow tracking, instruction traces and autostart extensibility points) [4]. The advantage of this method is that it is insensitive to packing/obfuscation techniques, however it is time consuming and sometimes it has a limited view of the features that the program could exhibit given different input values.

V. Problem Definition and Proposed Solutions

Having seen the Malware overview, is a fact that current antivirus, which are based on signature method, can be bypassed and are no longer able to deal with the malware detection work [12]. Even more, the rate of detection of malware using data mining method is twice as compared to signature based method [15].

In addition, analysis from static features is not always possible given that several malicious programs are already packed. Consequently Dynamic analysis is also required and having tools or programs that works on execution time along the antivirus programs will help to detect malicious behaviors. Thus, fast strategies for analyzing and detecting malware are required. With this in mind, this paper propose the use of Machine Learning and Big Data tools for selecting the smallest representative number of features that keep the classification accuracy high, allowing possible embedded programs to run fast looking for the characteristics that match malware behavior (it is important to mention that the creation of the embedded program is not the purpose of this paper).

In sum, the work presented on this paper follows four main stages. First, scripts-programs for downloading and parsing the metadata, static and dynamic information given by the VirusTotal platform were created (see section VII). Second, two stages of feature selection for reducing the datasets, removing noise and selecting the smallest number of features keeping accuracy high was accomplished (see section VIII). Finally, the training speed and accuracy on the classification process was improved (see section IX).

VI. Related Work

Before we continue, we will briefly introduce the terminology and nomenclature for comparing results on the field of machine learning. True Positives (TP) are the goodware samples classified correctly. The True Negative (TN) are the malware samples classified correctly. The False Positives (FP) are malware samples classified wrong as goodware. False Negatives (FN) are goodware classified wrong as malware. The True Positive Rate (TPR) is the ratio between True Positives and total number of positives (P), thus TPR = TP/P. The True Negative Rate (TPR) is the ratio between True Negatives and total number of negatives (N), thus TNR = TN/N. Finally, the Accuracy (ACC) is the sum of True positives and True Negatives divided by the total number of samples. ACC = (TP+TN)/(P+N).

A. Static and Dynamic features for Classification

Michal Kruczkowoski and Ewa Niewiadomska, 2014 [19] used a dataset of 10746 samples, they got an overall accuracy (ACC) of 0.958. D. Swathigavaishnave and R. Sarala, 2012 [7], using opcode features on a dataset of 500 malicious and 300 benign files, they got a TPR of 0.992 and a FPR of 0.53. Rafiqul Islam, Ronghua Tian, Lynn M. Batten and Steve Versteeg, 2013 [12] got an accuracy of 0.971 from a dataset of 2939 samples, using Static and Dynamic features. Igor Santos, Jaime Devesa, Felix Brezo, Javier Nieves, and Pablo G. Bringas 2013 [10] using opcodes and API calls got an accuracy of 96.6% on a dataset of 1000 Malicious and 1000 good files. Finally, Ekta Gandotra, Divya Bansal, Sanjeev Sofat 2014 [4] gave an overview of the state of the art on Malware analysis and Classification. Also, as we can see, all of these researches focus on accuracy but not in feature selection or reduction, which we believe is important for fast analysis and training, and prediction required on real-time detection.

TABLE I. Comparison of Related Researches On Accuracy

Reference	Year	Data	Features	Accuracy
Schultz ref [12]	2001	4266	String (printable and not printable)	97.11
[5]	2006	3622	Byte n-grams	96.8
[9]	2010	--	System Call [14]	96.8
[7]	2012	800	Opcode sequences	99.2
[10]	2013	26189	Opcode n-gram + APIs, Function calls	96.22
[8]	2013	12199	Byte n-grams	96.64
[12]	2013	2939	FLF + PSP, + API calls	97.055
[11]	2013	2.6 M	Several static and dynamic	99.58
[13]	2015	121856	API calls and API parameters	97.2
Our approach	2016	14902	Several static and dynamic	99.60

B. Feature selection

Usukhbayar Baldangombo, Nyamjav Jambaljav, and Shi-Jinn Horng, [6] using static features as PE headers, DLLs and

API functions, they selected the best subset of features consisting on 88 PE headers that had the best performance with their classifiers (accuracy of 0.995). The dataset was 236756 malicious and 10592 clean programs. Despite they applied feature reduction, static features are not convenient to detect malware files given that malware detection based on static features can be bypassed by obfuscation methods. Chih-Ta Lin, Nai-Jian Wang, Han Xiao and Claudia Eckert, [18] created n-grams from static and dynamic features. Using around 790.000 n-grams, they applied feature selection-reduction and they got an accuracy near to 90% with 10 features and 96% with 100 features. The dataset was 3899 malware and 389 benign samples. About this article, feature selection and reduction was made on the number of n-grams, however it is not strictly related with the number of features that is required to read from the static-dynamic behavior from the files. Also, compared to our results, we can achieve an accuracy of 99.60% with just 9 features. George E. Dahl, Jack W. Stokes, Li Deng and Dong Yu, 2013 [11] used Deep Neural networks with static and dynamic features. In addition, they used mutual information for selecting 179.000 features from 50 Millions and then Random Projections for reducing to "few" thousand dimensions. The overall accuracy was 99.58% on a dataset of 2.6 Millions of files. Thus, we can see that the accuracy is really high; however we are looking for real-time applications, so thousands of features could be still considered many features.

TABLE II. COMPARISON OF RELATED RESEARCHES ON FEATURE SELECTION AND ACCURACY

Reference	Year	Data Set	Features	Accuracy	#Selected Features
[6]	2012	247348	DLLs, APIs, PE header (static features)	99.5	80
[18]	2015	4288	Static and Dynamic. N-grams features	~96	100
Our approach	2016	14902	Several static and dynamic	99.60	9

C. Real-time malware detection

Micha Moffie, Winnie Cheng and David Kaeli, 2006 [17] developed a security framework as complement to anti-virus programs called HTH. It is able to extract vast amount of runtime information in a "faster" manner (System calls, Library calls, Data flow), which is used for malware detection on real time. Regarding to this article, we believe that the overhead could be significant reduced if just the most important features are collected for detection on real time.

VII. CREATING THE DATASET

In order to get the information from Malware and "good" files, we considered two possibilities, use the Cuckoo Sand-box software [15], which is a platform for Malware analysis that provide different information after running the file on an isolated environment. On the other hand, VIRUSTOTAL [1], which is a free online service for analyzing files or URLs and runs a distributed setup of Cuckoo sandbox, "enabling the identification of viruses, worms, Trojans and other kinds of malicious content detected by antivirus engines and website scanners" [1]. In addition, these tools provide information such as file version and properties, PE info, file metadata, behavioral information, etc. Finally, we decided to use VIRUSTOTAL, given that it is possible to save the time regarding to the analysis because results are already available from the VIRUSTOTAL website.

VIRUSTOTAL provide a public API for scanning, submit, and accessing to results. However, all the information is not public, so it was necessary to get the permission for having access to full scan results. These information was parsed to a tables and finally it was collected 7630 Malware and 1818 "Goodware" scan reports. For each scan report, two files on "JSON" format were gotten. Using R CRAN [16], the files were parsed and 57 different type of features were extracted; just for mentioned some ones, files opened, copied, deleted, etc, DLLs, codesize, Flags, datasize, language-code, file-info, entry-point, PE-info, imports, services, API-info, processes-info, network-info, among others. Finally, the information extracted was storage on matrices and the total number of features was 682.936 with a size of 22 GB.

VIII. FEATURE SELECTION

Feature selection is an important step for this research. As mentioned, one of the objectives is to select the smallest number of features that keep the detection rate as high as possible to allow to use the minimum quantity of resources for the malware detection task. Furthermore, feature selection and reduction is already know that can reduce the noise, improve the accuracy, and of course improve speed for training the classification algorithms as given that time increased in O(n^2) with respect to the number of features as state by Kolter and Maloof [5]. Next, the processes used for feature selection is explained.

First Stage: Due to the large dimensionality we used SPARK given that this platform can deal with large datasets. The machine learning library for SPARK - MLlib – was used as first stage for feature selection. In this library, the Feature Selection "ChiSqSelector" was applied getting the 10 % of the most relevant features. This first reduced matrix contains 68.800 features with 9448 observations with a size of 2.2 GB.

Second Stage: In this point, it was possible to select the best features running the Random Forest library (randomForest) on R – Cran (Rstudio) (ranking by decrease on accuracy and ranking on decrease on node impurity – see "importance" on randomForest package). In this process, six steps of reduction were developed running for each step the algorithms for classification. First, it was reduced from 68.800 features to 10.000 features, next to 5.000, 1.000, 300, 100, 30, 10 and finally 9.

Third Stage: In order to validate how well the feature selection was being accomplished, classification algorithms as SVM, RF and NN were used for tracking the classification accuracy. Charts 1, 2, and 3 compare the accuracy as result of the described process.

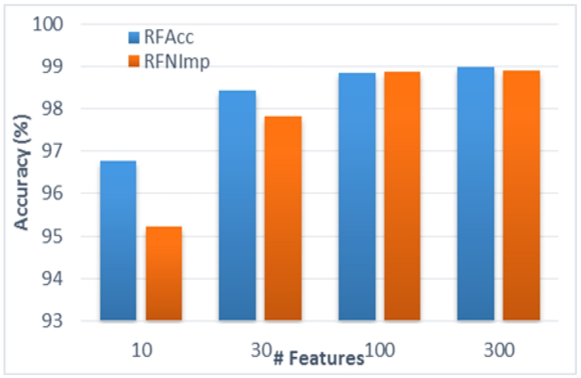

Chart 1. Comparison on accuracy performance for the feature selection algorithms using Random Forest.

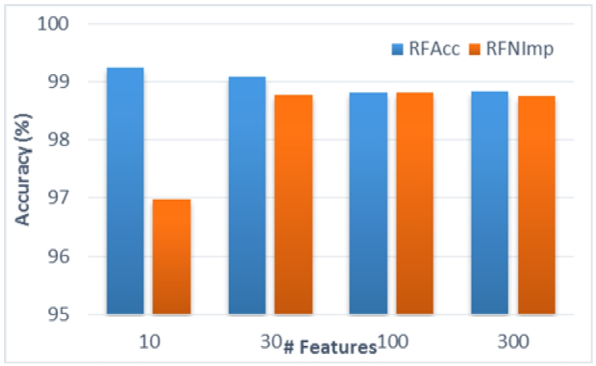

Chart 2. Comparison on accuracy performance for the feature selection algorithms using SVM.

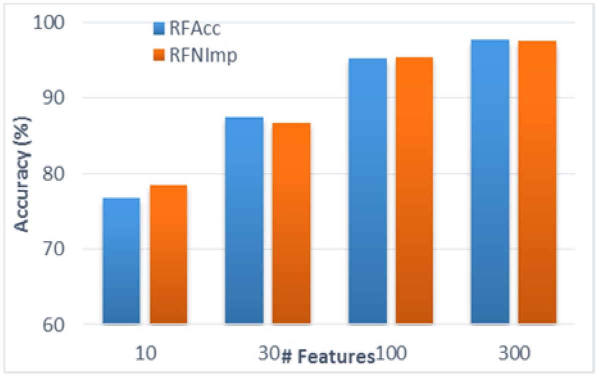

Chart 3. Comparison on accuracy performance for the feature selection algorithms using Neural Network.

From the charts above, it is possible to infer that ranking by decrease on accuracy (RFAcc) performs better that the other one. Furthermore, SVM classifier had a better behavior on accuracy than other methods specially using small amount of features. However, it is important to mention that Neural Networks were trained with just 100 iterations just for saving training time and because the objective in this stage was just compare the performance of the feature selection methods (no comparing the performance between classification algorithms).

Another important consideration to mention is that due to the *unbalanced* data (7630 malware files vs 1818 goodware files), *oversampling* was used (goodware files were duplicated 4 times, and thus the new amount of observations was 14902).

As example, the next table shows the selected best "9" features. (Note: CSUM correspond to the sum of the observation for each feature type, thus row 1 is the total number of "imports" that a specific program make).

TABLE III. FEATURE SELECTION FOR THE BEST 9 FEATURES.

Rank	Feature Type	Feature Name
1	IMPORTS	CSUM
2	NETWORK DNS_HOST	CSUM
3	CALLS ARGUMENTS_NAME	lpFileName
4	IMPORTS	EnumSystemLocalesW
5	PE Overlay Size	512
6	FILE FLAGS MASK	63
7	CALLS CATEGORY	system
8	Language Code	Neutral
9	TRID	Win32 Executable generic

IX. Classification Algorithm Results

A. Preliminary Accuracy Results

In this first stage, two main algorithms have been used, Random Forest (RF) and Neural Network (NN) in R-Cran (packages "randomForest" and "nnet"). The table IV shows the results for each algorithm. Note: Positive refers to "Goodware" (P). To mention, the testing set was 3724 observations that correspond to the 25% of the original "balanced" dataset. In addition, results shows that Random Forest (RF) present similar accuracy to Neural Network (NN) for large number of features, however RF was better in general, especially for smaller number of features. Regarding to the parameters for RF, the number of trees to "grow" was 1000. For NN, the number of iterations was 200 (other parameters as default). It's important to mention that RF had a better True Positive Rate than NN, which implies RF had better performance to classify "goodware" samples. On the other hand, NN was a lower False Positive Rate than RF, so NN classify better the "malware" samples. Finally, regarding to accuracy, it presented a peak around 100 and keep almost the same (in fact decrease) even with the features increase.

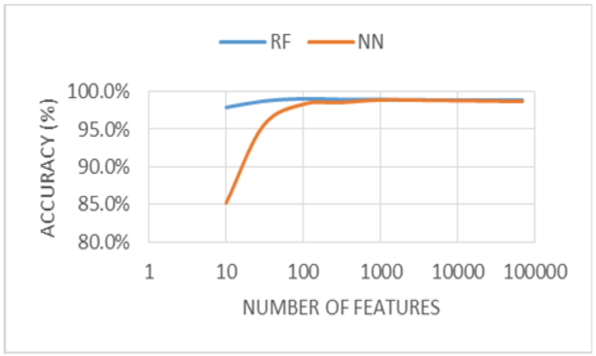

Chart 4. Accuracy performance for RF and NN through the number of features.

B. Improving Neural Network classification performance

On this stage, we decided first to work more on the Neural Network, given that previously we used just 200 iterations. The neural network model used was the nn package on TORCH with one hidden, one Tanh and one logsoftmax layer. Also, we divided the dataset on 70% for Training and 30% for Testing (Note that previously it was divided 75% - 25%; this is more exigent than previously). Using the dataset with the best 30 features selected by RF-decrease on accuracy, and running the nn based on accuracy, results shows that even with a large number of iterations, the Test accuracy does not increase considerably.

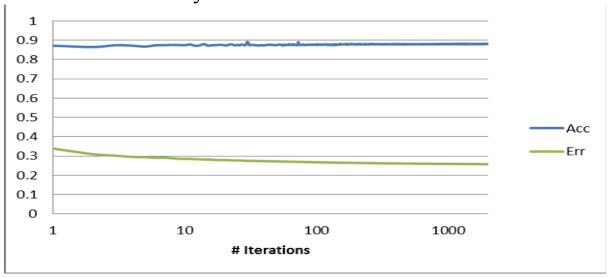

Chart 5. Training Error and Test Accuracy for best 30 features dataset.

Here is interesting to ask why SVM has better performance in this case. Without going deep, simple Neural Networks (with one or some small hidden layers) are based on combinations of sums of their inputs multiplied by weights and some nonlinear functions as tanh or sigmoid or RELU applied on some layers. On the other hand, SVM uses nonlinear kernels (polynomial, radial basis functions, tanh) to transform its inputs into a hyperplane that is easier to separate.

Next, we introduce simples but effective steps to preprocess the dataset to improve not just the speed but also the accuracy of the Neural Network algorithm. First, we need to apply nonlinear functions as the kernels used by the SVM to the dataset. Being said that, we used the SVM kernel already available on the SVM algorithm; these ones were applied on **each feature vector**. Thus, after having tried the polynomials and Radial base functions, this last one presented the best results for transform each feature vector into a hyperspace. After having applied these transformations with the SVM kernels, we assembled all of these new feature vectors into another dataset and the Neural Network was trained again. Next charts compare the behavior of the training error – Err- (the track of the errors or loss during the training process) and the accuracy – Acc - (gotten on the test dataset for each iteration) for the Neural Network applied to the original dataset – Base-, and the NN applied to the transformed dataset –withSVM-.

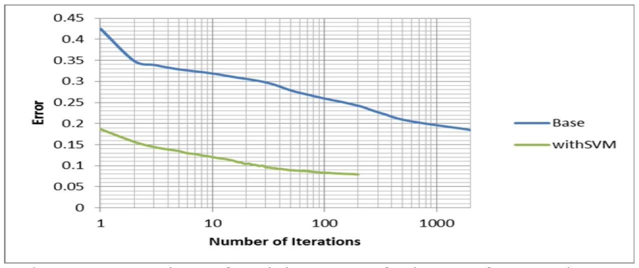

Chart 6. Comparison of Training Errors for best 30 features dataset (75% training – 25% test) (Base dataset and transformed with SVM).

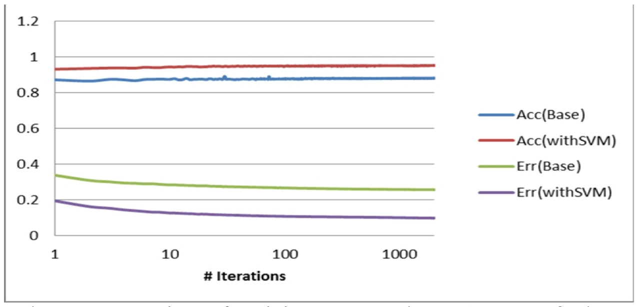

Chart 7. Comparison of Training Errors and Test Accuracy for best 30 features dataset (Base dataset and transformed with SVM. 70% training – 30% test).

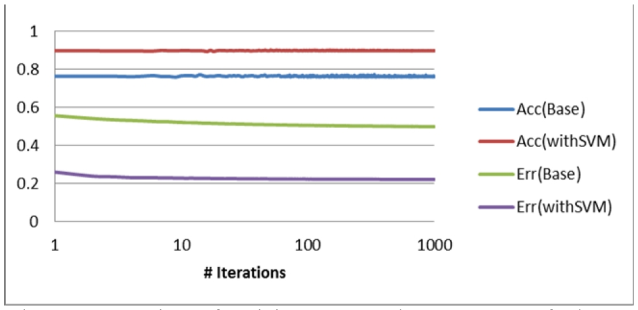

Chart 8. Comparison of Training Errors and Test Accuracy for best 9 features dataset (Base dataset and transformed with SVM70% training – 30% test).

From “Chart 6” we can see that the training error after 2000 iterations on the base dataset is reached with just two iterations on the transformed dataset and it is also the double of the error after 200 iterations on the transformed dataset. Even more, from “Chart 7” and “Chart 8”, the accuracy from the test dataset increased around 8% to 17% and errors decreased around 55% to 60% in average. In fact just with one iteration, results are better that the base dataset after 2000 iterations (see Chart 7) or 1000 (see Chart 8) iterations.

The preprocessing discussed is transforming the feature vectors applying functions to transform to another space easier separable. Concluding this section, preprocessing the feature vectors by separated before training the classification algorithms increase the accuracy and the speed given that with few iterations is possible to get lower errors and better accuracy (at least it was valid on our dataset using the Neural Network algorithm). It is also important to mention that the time to transform the features, let say of our 30 best feature dataset, it was around six minutes, which compared with the time that took to train the NN with this dataset (around a couple of hours or more) is small. Note: SVM was also applied after transform the feature vectors with the SVM kernel, but the accuracy remains quite similar and sometimes it decreased.

C. Improving the Test Accuracy on the Best 9 Features Dataset:

In this section we present the process used for improving the current Test Classification Accuracy, which is 99.24% for the 10 best features using the SVM algorithm. First, SVM was used to track the accuracy sequentially adding feature vectors in the ranked order. We saw that with the first 9 features the higher accuracy is reached (99,24%).

Next, we used Principal Component Analysis – PCA to transform the original dataset (Best 9 features dataset) and then we applied SVM and RF. From this step, the RF accuracy was increased from 96.78% to 97.6%. In this point, we decided combined all the information gathered, so the original dataset (Best 9 features dataset), the dataset transformed by SVM (SVM kernel applied to each feature vector), the dataset transformed by RF, the dataset after PCA transformation, the results by RF from the original dataset, the results by RF from the dataset transformed by PCA and the results by SVM to the original dataset. With this new combined dataset, we applied SVM and NN, but preliminary results show that SVM made overfitting and the accuracy decreased. NN through variable accuracy results near to 99.24%. Consequently, we decided to use the schema shows on the chart 9a, where we tried different combinations to create a new dataset that give us the best result. Finally after around 40 combinations, we found that the schema on chart 9b gives the best result, which increase the Testing Accuracy up to 99.60% from the best 9 features. This schema includes the original dataset, the transformed dataset by SVM kernels applied to each feature and the results by SVM applied to the original dataset. Note: The total time required for create the dataset and training the schema on Chart 9b was around 16 minutes. The NN required 255 iterations on 13 minutes.

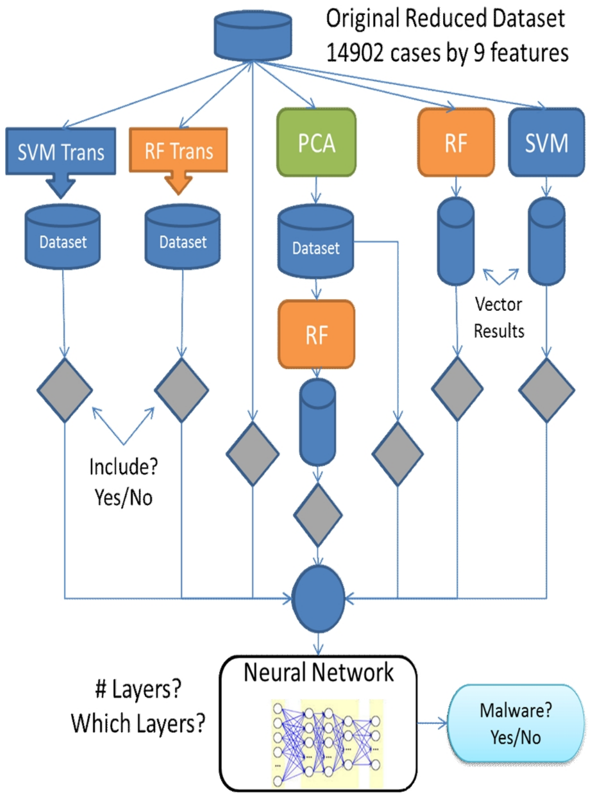

Chart 9. a) Assembly classification schema.

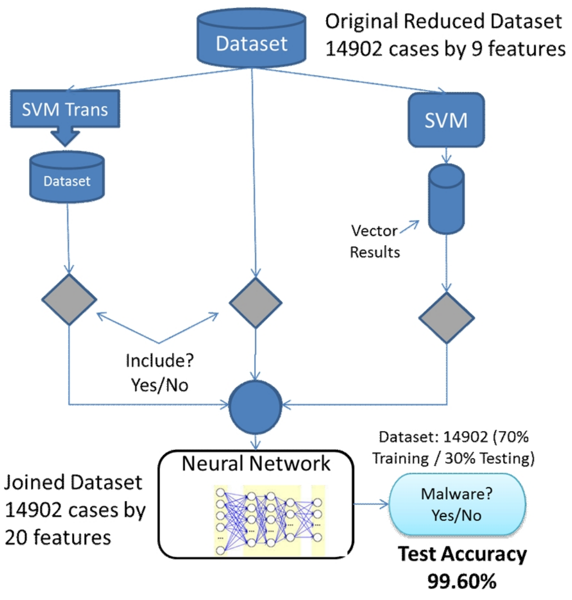

Chart 9. b) Best assembly classification schema

Finally, we decided to build the Receiver Operating Characteristic – ROC curve and then run the created model on a new dataset of 253 malware files downloaded from VIRUSTOTAL that were submitted by first time between October- 2015 to June-2016. Next, we shows the ROC curve which illustrates the good performance of our classifier (the area under the curve - AUC was around 0.997). Regarding to the new dataset, the accuracy was 98.4%.

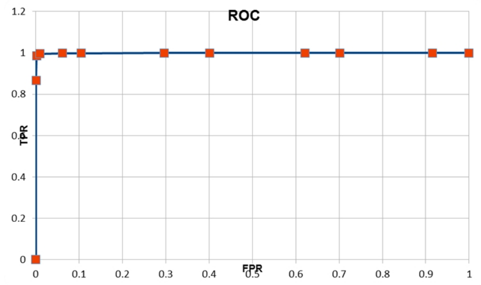

Chart 10. ROC for the model

X. Conclusions

- Features on this particular dataset can be ranked properly by Random Forest by decrease on accuracy.
- The best classification accuracy can be gotten using the 9 first features ranked by Random forest by decrease on accuracy. It will allow on future work to build a monitoring malware detection program with low overhead for the operating System.
- Final accuracy result was 99.60% which is even better than classifications without feature selection with hundreds or thousands of features.
- Preprocessing the dataset transforming each feature vectors using classification algorithms or equivalent transformation functions can increase the accuracy and the speed given that with few iterations is possible to get

lower errors and better accuracy (at least it was valid on our dataset using the Neural Network algorithm).

- Neural Networks can be used as final stage of an assembly of classification algorithms to improve accuracy results.
- The vector feature transformation open the possibility to explore building new architecture layers on the Neural Network. We believe that structures that allows emulate more complex functions as polynomials, exponentiation, etc. will help to increase the overall performance of the Neural Networks.

XI. Future Work

The next stage that will be held on summer-2016 will be design the structure of the monitoring malware detection program. Additionally, an implementation of the greedy algorithm for feature selection will be made to confirm or even more reduce the number of features.

References

[1] VirusTotal. https://www.virustotal.com/en/

[2] Cloud Security Alliance, "Expanded Top Ten Big Data Security and Privacy Challenges," April, 2013.

[3] Xin Hu, "Large-Scale Malware Analysis, Detection, and Signature Generation," The University of Michigan, 2011.

[4] Ekta Gandotra, Divya Bansal, and Sanjeev Sofat, "Malware Analysis and Classification: A Survey," IPEC University of Technology, Chandigarh, India. Journal of Information Security. April 2014.

[5] J. Zico Kolter, and Marcus A. Maloof, "Learning to Detect and Classify Malicious Executables in the Wild," Stanford University. Journal of Machine Learning Research 7. 2006.

[6] Usukhbayar Baldangombo, Nyamjav Jambaljav, and Shi-Jinn Horng, "A Static Malware Detection System Using Data Mining Methods," National University of Mongolia, and University of Science and Technology, Taipei, Taiwan. 2012.

[7] D. Swathigavaishnave, and R. Sarala, "Detection of Malicious Code-Injection Attack Using Two Phase Analysis Technique," Pondicherry Engineering College Puducherry, India. May 2012.

[8] Ohm Sornil (National Institute of Development Administration, Bangkok, Thailand), and Chatchai Liangboonprakong (Suan Sunandha Rajabhat University, Bangkok, Thailand), "Malware Classification Using N-grams Sequential Pattern Features," International Journal of Information Processing and Management(IJIPM) Volume4, Number5. July 2013.

[9] Firdausi, I., Lim, C. and Erwin, Analysis of Machine Learning Techniques Used in Behavior Based Malware Detection, Proceedings of 2nd International Conference on Advances in Computing, Control and Telecommunication Technologies (ACT), Jakarta, 2010, 201-203. 2010

[10] Igor Santos, Jaime Devesa, Felix Brezo, Javier Nieves, and Pablo G. Bringas, "OPEM: A Static-Dynamic Approach for Machine-learning-based Malware Detection," Deusto Institute of Technology University of Deusto, Bilbao, Spain. Year: --.

[11] George E. Dahl (University of Toronto, Toronto, ON, Canada), and Jack W. Stokes, Li Deng, Dong Yu (Microsoft Research, One Microsoft Way Redmond, WA 98052, USA), "Large-Scale Malware Classification Using Random Projections And Neural Networks," ICASSP. 2013.

[12] Rafiqul Islam (Charles Sturt University, NSW 2640, Australia), Ronghua Tian (Deakin University, Victoria 3125, Australia), Lynn M. Battena and, Steve Versteeg (CA Labs, Melbourne, Australia), "Classification of malware based on integrated static and dynamic features," Journal of Network and Computer Applications 36 (2013) 646–656. 2013.

[13] Mariano Graziano, Davide Canali and Davide Balzarotti (Eurecom), Leyla Bilge (Symantec Research Labs), Andrea Lanzi (Universita' degli Studi di Milano), "Needles in a Haystack: Mining Information from Public Dynamic Analysis Sandboxes for Malware Intelligence,". 2015.

[14] Smita Ranveer, and Swapnaja Hiray, "Comparative Analysis of Feature Extraction Methods of Malware Detection," Sinhgad College of Engineering, Savitribai Phule Pune University, India. International Journal of Computer Applications (0975 8887) Volume 120 - No. 5. June 2015.

[15] Cuckoo-Sandbox. http://www.cuckoosandbox.org/

[16] R Cran. https://cran.r-project.org/index.html

[17] Micha Moffie, avid Kaeli (Northeastern University, Boston, MA) and Winnie Cheng (Massachusetts Institute of Technology Cambridge, MA), "Hunting Trojan Horses". 2006.

[18] Chih-Ta Lin, Nai-Jian Wang, Han Xiao and Claudia Eckert. "Feature Selection and Extraction for Malware Classification". National Taiwan University of Science and Technology, Taipei. Taiwan. 2015.

[19] Michal Kruczkowoski (Research and Academic Computer Network NASK, Warsaw, Poland) and Ewa Niewiadomska (Institute of Control and Computation Engineering, Warsaw University of Technology, Warsaw, Poland), "Comparative Study of Supervised Learning Methods for Malware Analysis". 2014