Journal of Engineering Science and Technology
Vol. 15, No. 2 (2020) 967 - 981
© School of Engineering, Taylor's University

AN EXPERIMENTAL STUDY TO EVALUATE THE PERFORMANCE OF MACHINE LEARNING ALGORITHMS IN RANSOMWARE DETECTION

YAP L. DION, SARFRAZ N. BROHI*

School of Computing & IT, Taylor's University, Lakeside Campus, Selangor, Malaysia
*Corresponding Author: SarfrazNawaz.Brohi@taylors.edu.my

Abstract

The research in the domain of ransomware is rapidly emerging, and the application of machine learning algorithms in ransomware detection is one of the recent breakthroughs. In this research, we constructed an experimental platform using ransomware datasets to compare the performance of various machine learning algorithms such as Random Forest, Gradient Boosting Decision Tree (GBDT), Neural Network using Multilayer Perceptron as well as three types of Support Vector Machine (SVM) kernels in ransomware detection. Our experiment is based on a combination of different methodologies reported in the existing literature. We used complete executable files in our experiment, analyzed the opcodes and measures their frequencies. The objective of this research was to discover the algorithms that are highly suitable to develop models as well as systems for ransomware detection. Consequently, we identified that Random Forest, GBDT and SVM (Linear) have shown optimal results in detection of ransomware.

Keywords: Decision tree, Gradient boosting, Machine learning, Neural network, Random forest, Ransomware, Support vector machine.

1. Introduction

Cyber ransomware became a major point of focus in the research world since the major WannaCry attack that occurred in May 2017. Based on Kaspersky Lab's Report 2017, WannaCry was ranked as the highest among other encrypting ransomware attacks which accounted 13.4% of all workstations in the first half of that year [1]. Existing commercial solutions provide various analysis approaches to detecting ransomware. In static analysis, the threats are detected by relying on unique pattern recognition [2]. Despite that, many solutions adopt this approach due to its inexpensive and fast detection. Signature-based detection falls short in countering with zero-day attacks that essentially means that new attacks cannot be detected. In contrast, behavioral-based detection adopts a profiling method by creating profiles for each malicious behavior [3, 4]. Even though it is substantially superior over signature-based, it generates issues where there is a disconnection in malicious behavior and the attack's goal [5]. Thus, it fails to detect obfuscation techniques such as polymorphism [6, 7].

The reality of research in the domain of ransomware is scarce. Hence, by conducting this research, it tackles the issue and impacting the research community by creating more ideas and opportunities to explore. The experiment was conducted in detecting ransomware from good-ware using four types of classification machine learning algorithms, and three out of the seven types of feature selection methods were chosen for the best results using real ransomware samples. Our research was inspired by studies of automatic labeling of malware samples and using a whole executable to conduct analysis [8, 9]. Many of the commercial antivirus providers are facing issues in labeling malicious samples. We thus adopted the idea of using the former study in predicting ransomware. The latter study gave us process guidance in working with analyzing ransomware samples in an executable file. The motivation of this experiment is to determine which machine learning algorithms work best with the ransomware datasets. The findings of this research will assist the industries and researchers of the domain in designing and developed systems as well as models using the most appropriate machine learning algorithms for ransomware detection. The rest of the paper is organized as followed: Section 2 critically discusses the existing research in the domain. Section 3 describes the research methodology. Section 4 demonstrates the experimental factors and Section 5 discusses the results. Lastly, Section 6 concludes the research with the discussion on its future direction.

2. Related Work

In this research, we reviewed both supervised and unsupervised machine learning algorithms. There are many studies on using machine learning to detect malware, but few studies focus specifically on ransomware, although ransomware is one of the latest threats to our data [10]. Unsupervised learning algorithms learn from unlabelled data, and one of the commonly used methods is clustering, where responses are grouped into clusters [11]. One of the works we looked into was Mutant-X, a static analysis malware clustering algorithm [12]. Mutant-X uses a clustering algorithm and emphasizes on the use of its' generic unpacking algorithm in detecting malware that uses obfuscation. It also reduces the number dimension to improve its efficiency. A different paper suggests the use of Balanced Iterative Reducing and Clustering using Hierarchies (BIRCH) by creating a summary of the huge dataset and clustering the data accordingly to reduce computational time [13], and then the summarized dataset can be complemented with other clustering methods [14]. One of the drawbacks of BIRCH is that it requires large dataset to be accurate; thus it is resource-intensive.

Supervised learning algorithms are powerful in classifying heterogeneous data by taking known datasets to produce a trained model [15, 16]. We studied MalCov Architecture using Convolutional Neural Network (CNN). This technique examines the whole .exe file to analyze whether the file is malware [9]. CNN is a feed-forward type of neural network, which can reduce the complexity of the network model with the number of weights [17, 18]. The architecture uses raw bytes extracted and placed in an embedding layer, training with the CNN giving it a degree of robustness despite facing minor alterations in byte values. Kotler and Maloof [3] and Ioffe and Szegedy [19] mentioned the use of byte n-grams, however, the model lacks robustness. There are several limitations in the model as the architecture has obstacles in working with batch-normalization because it prevents the model from learning the problem causing a downgraded performance. This is rather interesting because according to Singh et al. [20], batch normalization often improves generalization. The accuracy with batch normalization obtains 60% during training and a drop of 10% in the test process. Another study proposed a hybrid of a statistical and dynamic approach to improving efficiency and accuracy using Support Vector Machine (SVM) [21]. Applying both techniques to extract opcodes and system calls in parallel, it was possible to achieve a True Positive Rate (TPR) from 0.95 to 1 and False Positive Rate (FPR) in the range of 0.00300. However, the false positives grew from 0.04 to 0.28 as the data sets increased from 15% to 50%.

Chumachenko and Juutilainen [22] suggested the use of a random forest scheme. Furthermore, Zhang also mentioned that random forest with n-gram analysis employed high accuracy obtaining 91.43% against other algorithms that include Decision Tree (DT), Random Forest (RF), K-Nearest Neighbour (KNN), Naive Bayes (NB), and Gradient Boosting Decision Tree (GBDT) [23]. On the other hand, Kolter's works showed that boosted decision tree achieves a rate of 98% in detecting new malicious executables with a 0.05 false-positive rate by selecting relevant n-grams [3]. Kolter used a four-byte sequence of n-grams which produced 255 million distinct n-grams, reduced to 500 n-gram features while Zhang used 2, 3 and 4 grams as its features. In this research, we adopted a statistical approach along with supervised algorithms using op-code frequency analysis [24]. Opcodes are extracted through disassembly and exist in the software's assembly code. This allowed us to analyze malware without simulating execution as proposed by Sornil and Liangboonprakong [25].

3. Methodology

Figure 1 depicts the method that we applied in our experiment. It starts with the collection of datasets that contain ransomware. The datasets are pre-processed, and feature selection is performed using the most appropriate techniques. Once the data is ready, we implemented and applied machine learning algorithms using Microsoft Azure and Python programming packages to detect whether the file is ransomware and perform analysis in various visual representations. Supervised machine learning was chosen because in general unsupervised learning is computationally complex, thus requiring more time and data. We have collected several datasets that were available on the internet that contain various kind of malware. theZoo provided the necessary files that enable us to proceed with the process of data collection [26]. We obtained opcodes from the files as our dataset. Meanwhile, we also collected 200 good files to be disassembled to get the opcodes of the benign files. Due to the limitations of ransomware data, we generated dummy records of ransomware. We also collected several good files to be disassembled to get the opcodes of the benign files for the machine learning algorithm. We cleaned the data by only focusing on ransomware executable files. Opcodes are operation codes in a file that tells the computer what commands are supposed to be executed to run the files properly. We obtained the opcodes of each file by disassembling the files using IDA version 6.5. By using IDA, we were able to disassemble the files without executing them. Therefore, this method was safe as we were dealing with ransomware that could potentially harm our computers.

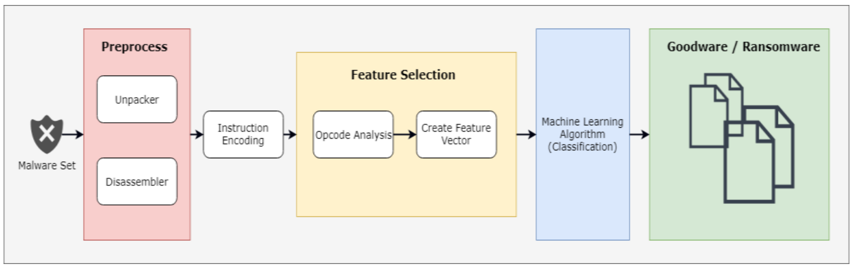


Fig. 1. The methodology of the experiment.

Upon disassembling the files, we found that there were around 3000 different opcodes. Since a huge number of features would take up a high amount of computational time to train. Therefore, to ease the training of the machine learning model, we put our dataset through a feature selection algorithm. Using Azure by Microsoft, we had the choice to choose from Chi-Squared, Principal Component Analysis (PCA), and Pearson's Correlation [27]. Among the algorithms, Chi-Squared was chosen. The process in choosing the feature selection algorithm goes by running all the algorithms with four different types of machine learning test model including decision tree, GBDT, neural network, and SVM. These machine learning algorithms were chosen based on the literature review and given popularity in the research field. We then picked the model which overall scores the best in terms of accuracy, precision, recall, f-score, Area Under the Curve (AUC), average and training log loss. We then made a comparison table among the top three feature selection algorithms.

Table 1. Feature selection method comparison table (Standard deviation).

Feature Selection	Accuracy	Precision	Recall	F-Score	AUC	Average Log Loss
PCA	0.01756	0	0.03226	0.01679	0.002273	0.05
Pearson's	0.0146	0.0093	0.0272	0.0149	0.0119	0.0669
Chi Square	0.0179	0.0167	0.032	0.0180	0.0107	0.0498

PCA is one of the most popular dimensionality reduction algorithms [28, 29]. Pearson's correlation describes the linear relationship between two variables usually shown in a scatterplot to define the strength of correlation [30]. Chi Square is a test for statistical purposes measuring the relation between two categorical variables using normalized values [31]. According to the results in Table 1, PCA achieved a standard deviation of zero in precision. Since precision only considers identification we must also consider recall that shows the actual proportion of positives identified. Pearson had a much better standard deviation than PCA and Chi-Square. Looking at Pearson's F-Score, there is no doubt it is the lowest among the rest due to both low accuracy and precision in terms of standard deviation. Despite that, it has a slightly higher log loss. We thus chose Chi-Square as it has a lower average log loss and although the rest of the results seemed to be not the best score but when we compare it with the other algorithms, it is substantially better in performance. Through the pre-processing of the dataset, we selected the opcodes that determine if the file with a specific opcode is benign or malicious. Huang mentioned that strong relevance in a feature does not suggest optimality [32]. In our case, we have three features indicating strong relevance. The results tend to skew if a file has a certain amount of the first three opcodes it would be considered as ransomware. We thus decided to remove the first three opcodes because although it has strong relevancy, it provides weaker accuracy.

4. Experimental Factors

The factors used in our experiment include the Accuracy, Sensitivity, Precision, F-Score, Receiver Operating Characteristics (ROC), Confusion Matrix, and Precision-Recall. The detailed description of these factors is provided in the following sub-sections.

4.1. Accuracy

Accuracy can be expressed as an overall view based on the number of events being classified correctly. In a high-level scale, it is easy to represent accuracy in the form of the number of true positive and negative events over the total in 100%. However, ultimately it is only designed to judge the accuracy of the specific model rather than considering the other aspects such as the falsely classified events [33]. Provost et al. [34] mentioned that merely using accuracy is misleading. The equation for accuracy is shown in Eq. (1).

$$\text{Accuracy} = \frac{\text{True Positive} + \text{True Negative}}{\text{All Events}} \tag{1}$$

4.2. Recall

The second performance metrics used is recall. A model's prediction can be categorized into four different types that are True Positive, False Positive, True Negative and False Negative. Sensitivity is the measure of the number of true positive events over the total number of true positive and false negative events as denoted in Eq. (2). This metrics is used to show the percentage of the true positive rate that can tell us the ratio of events that truly have ransomware.

$$\text{Sensitivity} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Negative}} \tag{2}$$

4.3. Precision

Precision also is known as specificity, is the measure of the number of true positive events over the total amount of true positive and false positive events as shown in Eq. (3). This metrics tells us the portion of correct positive classifications of ransomware from cases that are predicted as positive.

$$\text{Precision} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Positive}} \tag{3}$$

4.4. F-Score

F-Score takes the combination of both sensitivity and precision into account as denoted in the formula Eq. (4). F-score does not consider true negative events since F-Score is a normalized score [35]. Yedidia [36] also supports this idea where F-Score is difficult to give a sense of idea on its accuracy. Thus, we only use F-Score as a reference.

$$\text{F-Score} = \frac{2(\text{Recall} * \text{Precision})}{\text{Recall} + \text{Precision}} \tag{4}$$

4.5. Receiver Operating Characteristics

Another popular evaluation method is by using ROC. The ROC is used to show the relationship of sensitivity and the false positive rate.

4.6. Confusion Matrix

The confusion matrix gives an overview of the number of correct and incorrect prediction on a classification problem. The key difference of confusion matrix with other accuracy metrics is that we can gain insight into each classes' types of errors made.

4.7. Precision-Recall

Precision is the percentage of positive predictions that were correct while recall is the number of positive cases caught by the prediction model. When plotted on the graph we can see the relationship between these two.

5. Experiment: Results & Discussion

All experiments are conducted with Intel(R) Core(TM) i7-7500U CPU @ 2.70 GHz, 2904 MHz, 2 Core(s), 4 Logical Processor(s), 8GB of DDR4, NVIDIA GeForce 940Mx DDR3. Table 2 summarizes the experimental results on the random forest, neural network, GBDT and SVM. The neural network mentioned is referring to the multilayer perceptron algorithm. According to the result, we can see that random forest and gradient boosting decision has achieved higher accuracy of 0.97 and 0.96 respectively. The ROC curve shows that random forest achieves a slightly better result as compared to gradient boosting. The random forest model was able to detect more true positive events than GBDT and SVM (Linear). It also has the highest sensitivity of 0.96 while GBDT comes in second with 0.93 and SVM with 0.95. SVM using Radial Basis Function (RBF) and sigmoid kernel function achieved a high precision of 1.0 that means the model perfectly detected the true negative events. However, they performed very poorly in detecting the true positive events, thus, making the model unreliable. The precision for the random forest is 0.98734, GBDT is 0.98701 and linear SVM is 0.974683. The two metrics mentioned are then used to find the ROC curve. Each point would represent the two metrics corresponding to the specified threshold. The ROC curve is used to distinguish whether the model can achieve perfect discrimination. Thus, the accuracy rate is higher when the plotted graph is closer to the upper left corner [37].

Table 2. Experiment results.

Model	TP	FP	TN	FN	Accuracy	Sensitivity	Precision	F Score
RF	78	1	68	3	0.97333	0.96296	0.98734	0.975
GBDT	76	1	68	5	0.96	0.93827	0.98701	0.96202
NN	0	0	69	81	0.46	-	0.0	-
SVM Linear	77	2	67	4	0.96	0.95062	0.974683	0.9625
SVM RBF	29	0	69	52	0.65333	0.35802	1.0	0.52727
SVM Sig	2	0	69	79	0.47333	0.02469	1.0	0.04819

All models computed in the python script are taken from the scikit-learn module using its default parameters, except for the GBDT where the estimators are 30 and max depth at 10.

5.1. SVM Kernel Functions ROC

ROC curve is the relationship between Sensitivity plotted against 1-Specificity. The curve acts as a threshold in the use of maximizing true positive while decreasing false positive. However, in this case, we would focus on its AUC as it tells us more interesting things are happening. The AUC under the ROC curve is used to pinpoint a numerical value on the overall location of the ROC relative to the diagonal line [38]. We ran a comparison of ROC between the three different functions for SVM when we tried running it all at once. It took a rather long-time compiling. Thus we decide to go with this method where we would choose the best function and compare it with other classifiers. The polynomial function was excluded since it took a very long time to compute. The linear SVM performed much better as compared to using the sigmoid and RBF function as shown in Fig. 2. This is perhaps because the model uses binary classification where our dataset only has two classes. Thus, the linear function performed much better here. Moreover, since SVM-RBF and SVM-Sigmoid is a non-parametric model, as the dataset grows, overfitting occurs. Thus, when predicting the test datasets, it fails to generalize the new examples. We then chose the linear function for SVM and used it to compare with the rest of the other models.

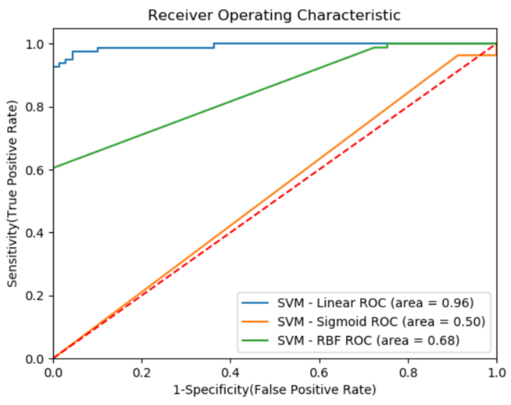


Fig. 2. ROC of SVMs.

5.2. Overall ROC

ROC in Fig. 3 summarizes the ROC curve. Random forest ROC achieved an area of 0.98, GBDT and SVM having 0.96 and Neural Network having only 0.50. Apart from the neural network, there is a steady increase to 1.0, and the models have approximately a sensitivity of 0.9 at a threshold of 0.0. Neural networks however only picked up around a threshold of 0.1 and spiked to above sensitivity of 0.8 from a sensitivity close to 0.0. This is expected because the neural network uses a backpropagation technique to determine the amount of adjustment needed. Thus, in the beginning, the model made many errors, and it slowly adjusts its gradient descent to achieve lesser. By the time it reaches at a threshold of 0.1, it immediately spikes up and slowly goes up to 1.0.

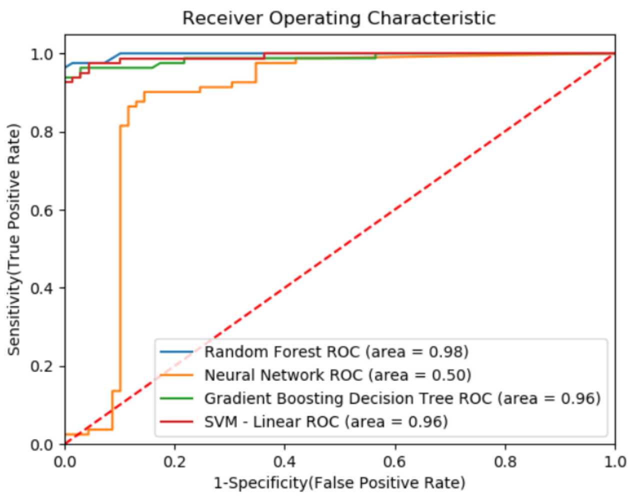
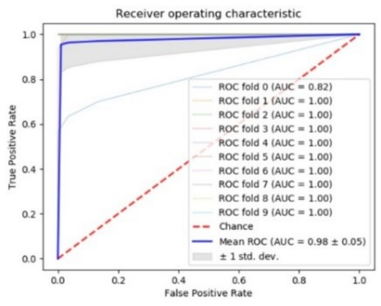


Fig. 3. Overall ROC.

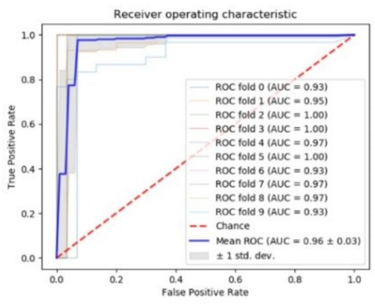
5.3. Cross validation ROC

The ROC plots in Fig. 4 shows each fold during cross evaluation. Cross evaluation is a technique that splits the datasets into different segments; in our case, we have ten segments, better known as folds. The first fold is compared with the rest of the folds, the second time it runs it takes the first and second fold to be compared with the rest, this goes on until 90% of training data is compared against 10% test data. The ROC curve at fold 0 for the random forest is much smoother than GBDT. This is because GBDT uses the boosting technique where each of the tree interdepends on the previous tree while trying to reduce the error as much as possible. For every spike we see is a great error reduction as it gradually increases to reach 1.0. Random forest's curve was smooth due to its bagging technique used where each of the trees run in parallel composing a forest. Since each tree also does not depend on another, the result of the whole forest is taken, hence showing a much smoother curve.

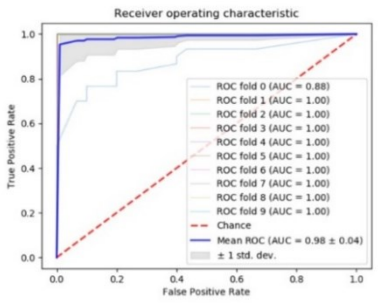
In the neural network, although the average AUC for the ROC is 0.8, the standard deviation shows that it is 0.18 that proves that the model varies very vastly. Since slight changes can cause the model to be unstable, it is unreliable to use the model. Between the three different functions for SVM, the linear function has the highest mean ROC of 0.96 and at a standard deviation of 0.03, as compared to Sigmoid at 0.07 and RBF at 0.14. Although SVM with sigmoid function has a lower standard deviation that RBF but the overall ROC is lower than 0.5 that signifies that its overall accuracy is very low. The linear function has a slightly lower standard deviation than the random forest, which may due to the model consistently achieving around 0.95.



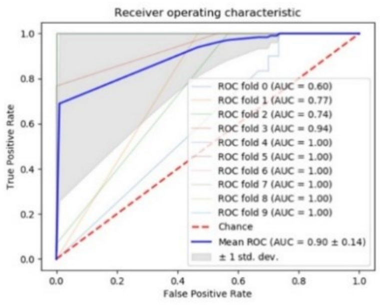
(a) Random forest ROC.



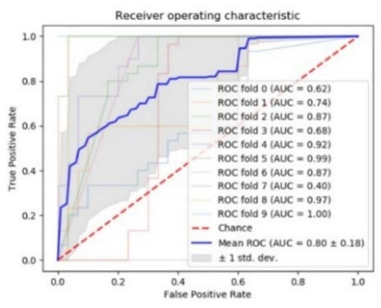
(b) SVM (Linear) ROC.



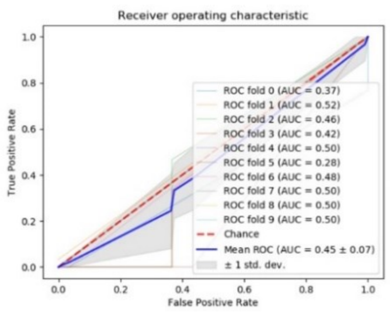
(c) GBDT ROC.



(d) SVM (RBF) ROC.



(e) Neural network ROC.



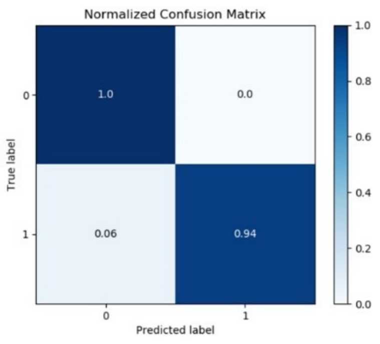
(f) SVM (Sigmoid) ROC

Fig. 4. ROC of cross validation.

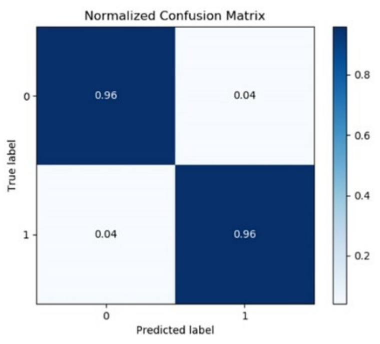
5.4. Confusion matrix

As the name suggests, when generating predictions, the confusion matrix is used to show whereabouts the classification model is confused by breaking down the summarized values into each class. The predictive performance of classification models can be charted out in a confusion matrix [39]. Figure 5 shows the normalized confusion matrix. The random forest has accurately detected all true negative events (good-ware) with 0.94 prediction level in detecting true positive events (ransomware). Upon closer inspection, neural network, SVM with RBF function and SVM with sigmoid function all achieved 1.0 in detecting true negative events but their true positive rates are far beyond accurate. The neural network has the worst outcome where zero ransomware events have been accurately detected.

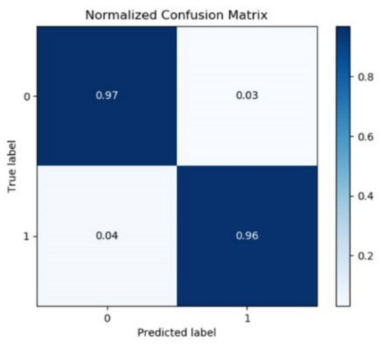
The sigmoid function also did not do well as only 2% of the true positive events were detected, and the RBF has slightly more than half of the ransomware events classified correctly. Since RBF and Sigmoid are non-linear functions, in a feature space of dimension N, if N > n then there will always be a separating hyperplane, but this may not give to good generalization performance, especially if some data that are randomly generated does not exactly reflect the actual ransomware dataset.



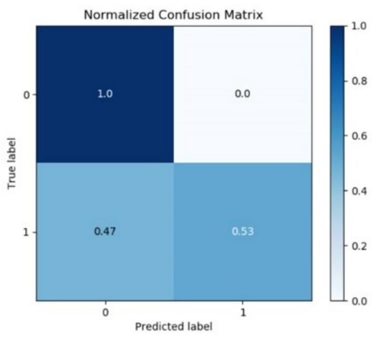
(a) Random forest confusion matrix.



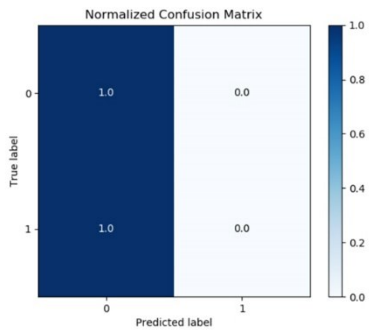
(b) SVM (Linear) confusion matrix.



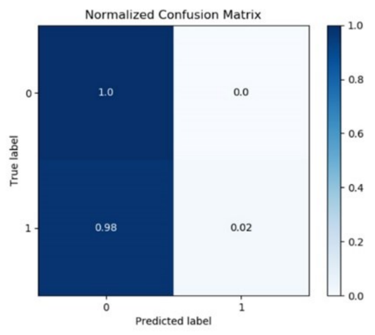
(c) GBDT confusion matrix.



(d) SVM (RBF) confusion matrix.



(e) Neural network confusion matrix.

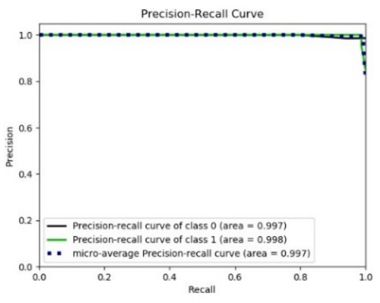


(f) SVM (Sigmoid) confusion matrix

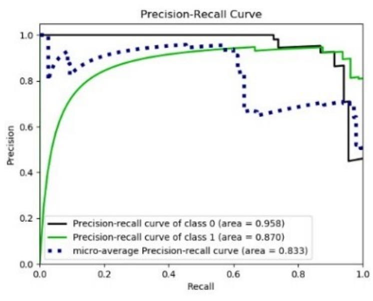
Fig. 5. Confusion matrix.

5.5. Precision-recall

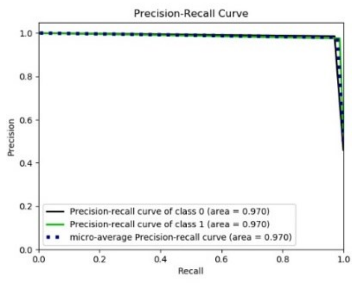
The precision-recall is another useful evaluation metrics in evaluating a model's output quality. Davis. J. and Goadrich. M argued that ROC curve shows a rather optimistic view of the curve due to the exclusion of negative cases [40]. Precision-Recall curves are alternatively used to ROC curves in Information Retrieval [41, 42]. Precision is used to measure the relevancy of a result while recall is to calculate how many relevant events have truly occurred. Thus, the curve shows the trade-off between the two types of measurements. It is desirable to achieve high precision and recall. High precision shows low false positive rates and high recall value represents low false negative rates. In Fig. 6, Class 0 represents good-ware while Class 1 represents ransomware.



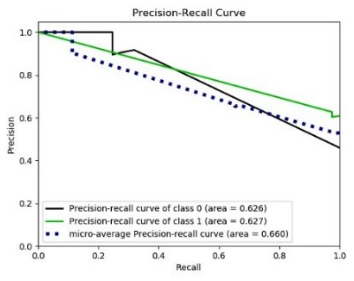
(a) Random forest precision-recall.



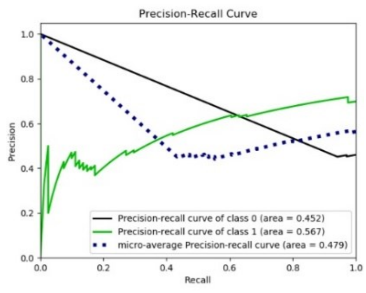
(b) SVM (Linear) precision-recall.



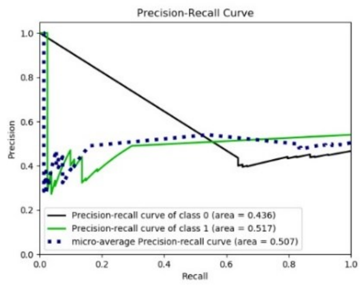
(c) GBDT precision-recall.



(d) SVM (RBF) confusion matrix.



(e) Neural network precision-recall.



(f) SVM (Sigmoid) precision-recall.

Fig. 6. Precision-recall.

Random forest classifier has an astoundingly large AUC of 0.997 for good-ware and 0.998 for ransomware where it has very few false positive and false negative events. GBDT also has high precision and recall of 0.97 for both classes. Both models show similar curve behaviour where both models can precisely select relevant items;

however, as a trade-off, recall rate is low at the beginning because not many relevant items are selected. The precision sharply decreases when the recall is at 1.0 because although the model could detect all relevant events, it was not able to precisely select relevant events. Neural network classifier shows very drastic changes between a good-ware event and ransomware events. There was a linear decrease for Class 0, as the model finds more and more events, its precision goes down because it did not properly select the cases. Class 1 had an opposite curve, as more and more cases are introduced, its precision starts to increase. However, it can only go as high as approximately 0.7. Overall, its precision-recall curve was undesirable as Class 0 only achieved AUC of 0.452 while Class 1 achieves 0.567.

SVM (Linear) has high precision at the start for good-ware events, as more of the events were found, its precision dropped. Ransomware class, on the other hand, had an exponential increase until it hits slightly higher than 0.6, it starts to drop a little. Both classes in SVM (RBF) the precision gradually decreases as the recall increases. As for SVM (Sigmoid), there was a sharp drop at the start and starts to stabilize around the precision of 0.5 as the recall goes up. SVM with linear function was the most desirable function out of the other two as it has 0.958 and 0.87 for Class 0 and Class 1. As a result of the overall experiment, we identified that for opcodes datasets specifically in ransomware, algorithms including random forest, GBDT, and SVM (Linear) had shown great results in the evaluation metrics while neural network and the other two functions used in SVM are not suitable for this type of data.

6. Conclusion and Future Work

In this research, our experimental platform can only detect ransomware that is either exe or ddl format. Our detection model will reject any other file format. In future research, we focus on a variety of file formats that could potentially contain ransomware. Moreover, we used supervised learning methods to detect ransomware. In the future direction of this research, we will focus on unsupervised learning by applying clustering methods such as k-means, hierarchical clustering, and OPTICS algorithms to detect ransomware. Using these algorithms, we will be able to determine which of these algorithms is more suitable to detect ransomware with the highest accuracy. Furthermore, our research would also include the computational complexity to show the differences with higher and lower computational resources when the approach is deployed. The other comparison that can be done is the results of unsupervised and supervised algorithms. Another scope that we will research is in conducting a network traffic analysis for ransomware because they are targeted for networks as well. Such a step would provide comprehensive information to the researchers in the domain on selecting the best machine learning algorithms to develop systems or statistical models for ransomware detection.

Abbreviations

FN	False Negative
FP	False Positive
GBDT	Gradient Boosting Decision Tree
NN	Neural Network
RBF	Radial Basis Function
RF	Random Forest
ROC	Receiver Operating Curve

SVM	Support Vector Machine
TN	True Negative
TP	True Positive
FN	False Negative

References

1. Kaspersky. (2017). The Rise of Ransomware. Retrieved October 30, 2018, from http://www.kaspersky.com.
2. Ranveer, S.; and Hiray, S. (2015). SVM Based Effective Malware Detection System. *International Journal of Computer Science and Information Technologies*, 6(4), 3361-3365.
3. Kotler, J.Z.; and Maloof, M.A. (2006). Learning to Detect Malicioud Executables in the Wild. *Journal of Machine Learning Research*, 7, 2721-2744.
4. Mujumdar, A.; Masiwal, G.; and Meshram, B.B. (2013). Analysis of Signature-Baed and Behavior-Based Anti-Malware Approaches. *International Journal of Advanced Research in Computer Engineering and Technology (IJARCET)*, 6 (2), 2278-1323.
5. Christodorescu, M.; and Kruegel, C. (2007). Mining Specifications of Malicious Behavior. *Proceedings of the the 6th joint meeting of the European software engineering conference and the ACM SIGSOFT symposium on The foundations of software engineering "Flight dynamics of unmanned vehicles"*. Dubrovnik, Croatia, 5-14
6. Zakaria, W.Z.A.; Abdollah, M.F.; Mohd, O.; and Ariffin, A.F.M. (2017). The Rise of Ransomware. *Proceedings of the 2017 International Conference on Software and e-Business.* Hong Kong, Hong Kong, 66-70.
7. Chakraborty, S. (2017). A Comparison study of Computer Virus and Detection Techniques. *International Journal of Scientific Research in Computer Science, Engineering and Information Technology (IJSRCSEIT)*, 2(1), 2456-3307.
8. Hu, X.; Griffin, K.; and Shin, K.G. (2018). MutantX-S : scalable malware clustering based on static features MutantX-S : Scalable Malware Clustering Based on Static Features. *USENIX Annual Technical Conference*, Washington, D.C., 187-198.
9. Raff, E.; Barker, J.; Sylvester, J.; Brandon, R.; Catanzaro, B.; and Nicholas, C. (2017). Malware Detection by Eating a Whole EXE. *arXiv preprint*. arXiv:170.09435
10. Edgar, P.; Torres, P.; and Yoo, S.G. (2017). Detecting and Neutralizing Encrypting Ransomware Attacks by Using Machine-Learning Techniques: A Literature Review. *International Journal of Applied Engineering Research, 12(18), 7902-7911.*
11. Amruthnath, N.; and Gupta, T. (2018). A Research Study on Unsupervised Machine Learning Algorithms for Fault Detection in Predictive Maintenance. *5th International Conference on Industrial Engineering and Applications,* Singapore, Singapore, 355-361.

12. Dutta, A.K. (2016). Detection of Malware and Malicious Executables Using E-Birch Algorithm. *International Journal of Advanced Computer Science and Applications (IJACSA),* 7(1), 124-126.
13. Zhang, T.; Ramakrishnan, T.; and Livny, M. (1997). BIRCH: A New Data Clustering Algorithm and Its Applications. *Data Mining and Knowledge Discovery,* 1, 141-182.
14. Kruczkowski, M.J.; and Niewiadomska-Szynkiewicz, E. (2014). Comparative study of supervised learning methods for malware analysis. *Journal of Telecommunications and Information Technology,* 4, 1-10.
15. MathWorks. (2016). What is Machine Learning?. Retrieved November 13, 2018 from https://www.mathworks.com/discovery/machine-learning.html.
16. Liu, T. Fang, S. Zhao, Y. Wang, P. and Zhang, J. (2015). *Implementation of Training Convolutional Neural Networks.* University of Chinese Academy of Sciences, Beijing, China.
17. Kabanga, E.K.; and Kim. C.H. (2018). Malware Images Classification Using Convolutional Neural Network. *Journal of Computer and Communications,* 6, 153-158.
18. Raff, E.; Zak, R.; Cox, R.; Sylvester, J.; Yacci, P.; Ward, R.; Tracy, A.; McLean, M.; and Nicholas, C. (2016) An Investigation of Byte N-Gram Features for Malware Classification. *Journal of Computer Virology and Hacking Techniques,* 14(1), 1-20.
19. Ioffe, S.; and Szegedy, C. (2015). Batch Normalization : Accelerating Deep Network Training by Reducing Internal Covariate Shift. *Proceedings of the 32nd International Conference on Machine Learning,* Lille, France, 448-456.
20. Singh, T.; Di Troia, F.; Corrado, V.A.; Austin, T.H.; and Stamp, M. (2015). Support vector machines and malware detection. *Journal of Computer Virology and Hacking Techniques,* 12(4), 203-212.
21. Agarap, A.F.; and Pepito, F.J.H. (2017). Towards Building an Intelligent Anti-Malware System: A Deep Learning Approach using Support Vector Machine (SVM) for Malware Classification. *arXiv preprint*. arXiv:1801.00318
22. Chumachenko, K.; and Juutilainen, M. (2017) *MACHINE LEARNING METHODS FOR MALWARE DETECTION AND CLASSIFICATION*. Bachelor's. Thesis. South-East Finland University of Applied Sciences, Kymenlaakso, Finland.
23. Zhang, H.; Xiao, X.; Mercaldo, F.; Ni, S.; Martinelli, F.; and Sangaiah, A.K. (2018). Classification of ransomware families with machine learning based on N-gram of opcodes. *Future Generation Computer Systems.* 90, 211-221.
24. Ghezelbigloo, Z.; and Vafaeijahan, M. (2014). Role-opcode vs. opcode: The new method in computer malware detection. *First International Congress on Technology, Communication and Knowledge (ICTCK)*, Mashhad Branch, Islamic Azad University, Mashhad, Iran.
25. Sornil, O.; and Liangboonprakong, C. (2013). Malware Classification Using N-grams Sequential Pattern Features. *International Journal of Information Processing and Management (IJIPM)*, 4(5), 59-67.
26. Nativ, Y. (2014). theZoo. Retrieved November 5, 2018, from https://github.com/ytisf/theZoo.

27. Microsoft. Azure Machine Learning Studio. Retrieved November 17, 2018 from https://studio.azureml.net.
28. Wold, S.; Esbensen, K.; and Geladi, P. (1987). Principal Component Analysis. *Chemometrics and Intelligent Laboratory Systems*, 2(1), 37-52.
29. Jollife, I.T. (2002). Principal Component Analysis, *Springer Science & Business Media*, 2, 487.
30. Wang, J.; and Zheng. N. (2016). Measures of Correlation for Multiple Variables, *arXiv*
31. Ugoni, A.; and Walker, B. (1995). The Chi square test: an introduction. *COMSIG review / COMSIG, Chiropractors and Osteopaths Musculo-Skeletal Interest Group*. 4. 61-4.
32. Huang, S.H. (2015). Supervised feature selection : A tutorial. *Artificial Intelligence Research,* 4(2), 22-37.
33. McNee, S.M.; Riedi, J.; and Konstan, J.A. (2006). Accurate is not always good: How Accuracy Metrics have hurt Recommender Systems. *Proceedings of the 2006 Conference on Human Factors in Computing Systems*, Montréal, Canada, 1097-1101.
34. Provost, F.; Fawcett, T.; and Kohavi, R. (1998). The case against accuracy estimation for comparing induction algorithms. *Proceeding of the 15th International Conference on Machine Learning,* San Francisco, CA, 445-453.
35. Powers, D.M.W (2015). What the F-measure doesn't measure: Features, Flaws, Fallacies and Fixes, *arXiv*.
36. Yedidia, A. (2016). Against the F-score.
37. Zweig, M.H.; and Campbell, G. (1993). Receiver-operating characteristic (ROC) plots: A fundamental evaluation tool in clinical medicine. *Clinical Chemistry*. 39(4), 561-577.
38. Hanley, J.A.; and McNeil, B.J. (1982). The Meaning and Use of the Area under a Receiver Operating Characteristic (ROC) Curve. *Radiology*, 143(1), 29-36.
39. Ostrand, T.; and Weyuker, E. (2007). How to measure success of fault prediction models. *Fourth international workshop on Software quality assurance: in conjunction with the 6th ESEC/FSE joint meeting,* 25-30.
40. Davis, J.; and Goadrich, M. (2006). The Relationship Between Precision-Recall and ROC Curves. *Proceedings of the 23rd international conference on machine learning*. ACM, 233–240.
41. Manning, C.D.; and Schutze, H. (1999). Foundations of Statistical Natural Language Processing. *MIT Press*, Cambridge, MA.
42. Raghavan, V.; Jung, G.S.; and Bollmann, P. (1989). A Critical Investigation of Recall and Precision as Measures of Retrieval System Performance. *ACM Transactions on Information Systems*, 7, 205-229.