Spring 2018

Boosted Hidden Markov Models for Malware Detection

Aditya Raghavan
San Jose State University

Boosted Hidden Markov Models for Malware Detection

A Project

Presented to

The Faculty of the Department of Computer Science

San Jose State University

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

by

Aditya Raghavan

May 2018

The Designated Project Committee Approves the Project Titled

Boosted Hidden Markov Models for Malware Detection

by

Aditya Raghavan

APPROVED FOR THE DEPARTMENTS OF COMPUTER SCIENCE

SAN JOSE STATE UNIVERSITY

May 2018

Dr. Mark Stamp	Department of Computer Science
Dr. Katerina Potika	Department of Computer Science
Mr. Fabio Di Troia	Department of Computer Science

ABSTRACT

Boosted Hidden Markov Models for Malware Detection

by Aditya Raghavan

Digital security is an important issue today, and efficient malware detection is at the forefront of research into building secure digital systems. As with many other fields, malware detection research has seen a dramatic increase in the application of machine learning algorithms. One machine learning technique that has found widespread application in the field of pattern matching and malware detection is hidden Markov models (HMMs). Since HMM training is a hill climb technique, we can often significantly improve a model by training multiple times with different initial values. In this research, we compare boosted HMMs (using AdaBoost) to HMMs trained with multiple random restarts, in the context of malware detection. These techniques are applied to a variety of challenging malware datasets and we analyze the results in terms of effectiveness and efficiency.

ACKNOWLEDGMENTS

I would like to thank Dr. Mark Stamp for his support and guidance throughout the project. I would also like to thank the committee members, Mr. Fabio Di Troia for his advice whenever in need and Dr. Katerina Potika for her valuable time.

TABLE OF CONTENTS

CHAPTER

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

Introduction

A recent study by International Telecommunications Union stated that, as of 2017, around 54% of households worldwide had access to the Internet [4]. In terms of pure numbers, the count of Internet users has increased from around 1 billion in 2005 to almost 3.6 billion in 2017 [5]. This trend of digitalization is expected to continue over the next few years and soon the entire world will be connected by Internet. These statistics indicate that both developed and developing nations are heavily reliant on their computers and Internet.

This proliferation of computers and Internet have resulted in digitalization of almost all services throughout the world. Not only businesses, even essential services like power grids, water dams, traffic lights etc. are all controlled by large scale computer systems. We also have an increasing number of Internet-of-Things (IoT) devices, which connect every aspect of an individual's life with the Internet. There are around 23 billion IoT devices all over the world as of today [6]. Although, digitalization and automation is the right way forward, they bring their own challenges and issues. These systems are vulnerable if not protected and maintained efficiently.

Bad actors exploit this over reliance on Internet for financial gain. There are reports of cyber-warfare being the foremost mode of attack in future conflicts world-wide [7]. Malware is the driving force behind such attacks. Malware, short for, malicious software, are programs which are designed to damage and/or affect the functioning of devices [8]. The scale and the mode of attack vary in each case as they range from a targeted attack on an individual to a worldwide ransomware attack.

From Figure 1 we can see that, every third computer in the world is affected by malware and this number is rapidly increasing. Such malware attacks, result in huge financial losses to those affected. Reports suggest that, there was a 62% increase in average cost of a cybercrime over five years in 2017 [9]. The cost of such financial losses are further expected to reach $6 trillion by 2021. Hence, combating malware is of crucial importance in the cyber world today and this increasing cost has also resulted in an increase in expenditure by the governments and businesses for protecting their systems which is expected to exceed $1 trillion in 2021 [10].

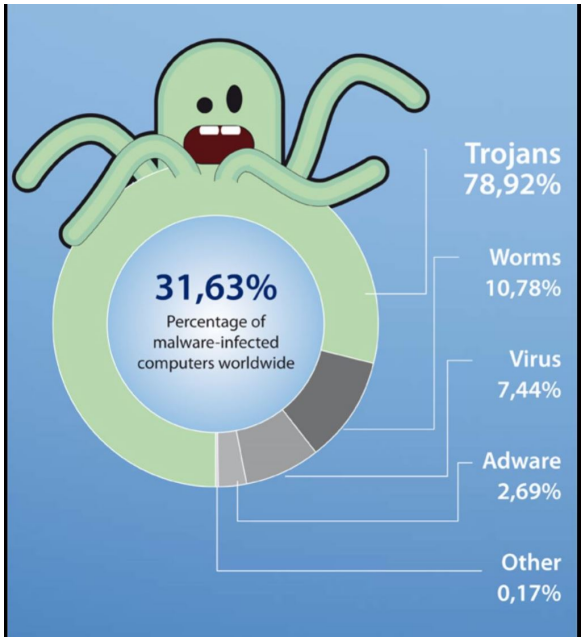

Figure 1: Computer Infections Worldwide [2]

There are different types of malware which affect the smooth functioning of computer systems in different ways. Adware, spyware, trojans, viruses, worms, ransomware etc. are some of the most common types of malware [11].

- `Adware` — Advertising supported malware display advertisements automatically using pop-ups on certain software which use adware as a source of revenue.

- `Spyware` — This malware, tracks the user activities on the device on which they are residing and are generally used in combination with other form of malware.
- `Trojans` — Malware that disguise themselves as verified and legitimate software are called as trojans. These generally install spyware to monitor the target user's activities by installing keyloggers.
- `Ransomware` — Malware that on infecting the target machines, steal, restrict access to important files for the users and ask for ransom in exchange for returning the files are termed as Ransomware. There have a couple of high profile instances of worldwide attacks using this form of malware in 2017.
- `Virus` — Malware that spread through a network of computers by replicating and possibly even mutating by piggybacking on other software are known to be viruses.
- `Worms` — Worm is one of the oldest forms of malware, that self replicate unlike viruses which require human intervention to spread.

Several techniques are employed to counter these ever-increasing malware threats. Anti-virus software, firewalls, intrusion detection/prevention systems are used in tandem to keep systems secure. Anti-virus software generally relies primarily on signature detection (i.e., pattern matching) to detect malware. However, there are many advanced forms of malware that can evade signature-based detection [12].

Machine learning techniques may be used to improve on signature detection [13]. Hidden Markov models (HMMs) are one popular machine learning technique that has been successfully applied to the malware detection problem [14]. In this research, we compare the effectiveness of malware detection using HMMs with multiple random restarts to combining HMMs using AdaBoost [15].

The report is organized as follows. In Chapter 2 we provide relevant background information, including an introduction to hidden Markov models and AdaBoost, and we also discuss problems related to malware detection. In Chapter 3, we look at boosted hidden Markov model and its implementation considered in this paper. Chapter 4 describes our experimental setup and results, and we provide some discussion and analysis of our results. Finally, in Chapter 5, we provide our conclusion and discuss future work.

CHAPTER 2

Background

Systems make adjustment on its own based on some intelligence and these enforced changes are termed as machine learning [16]. Several complex problems like character recognition and voice recognition have been solved using various machine learning techniques. Hidden Markov models is one such technique which has found widespread applications in solving pattern matching problems in an efficient manner [17, 18]. In this chapter, we introduce the concepts of hidden Markov models and AdaBoost. We also briefly discuss, the issues with machine learning applied in the context of malware detection and classification.

2.1 Hidden Markov Model

Hidden Markov models are based on a Markov process [14]. In a Markov process, the probabilities for a random process depends on a sequence of previous event(s). If the Markov process is of order one, it indicates that the probability of the next event depends only on the current event. Hidden Markov models make use of this feature of a Markov process to predict/determine probabilities of an unknown/hidden state as depicted in Figure 2.

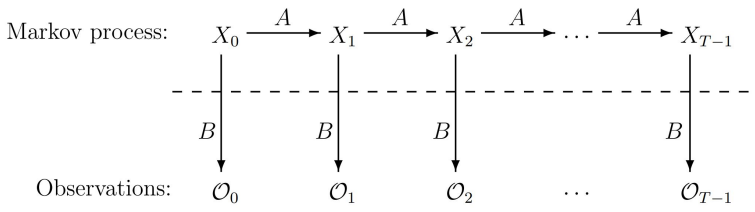

Figure 2: Hidden Markov Model

As mentioned in the Figure 3, the matrix denoted by A, holds probabilities of transition from one state to the other [14]. Another matrix, termed as the observation probability matrix, contains the values of emission probabilities. Both, state transition matrix and observation probability matrix are row stochastic. Finally, there is an initial state distribution matrix, also row stochastic, which can be set to any starting value deemed appropriate to have several starting points of the model. The state transition matrix is of size $N \times N$, while observation probability matrix is of size $N \times M$. The size of the initial distribution matrix is $1 \times N$.

T = length of the observation sequence
N = number of states in the model
M = number of observation symbols
Q = $\{q_0, q_1, \ldots, q_{N-1}\}$ = distinct states of the Markov process
V = $\{0, 1, \ldots, M-1\}$ = set of possible observations
A = state transition probabilities
B = observation probability matrix
π = initial state distribution
$\mathcal{O}$ = $(\mathcal{O}_0, \mathcal{O}_1, \ldots, \mathcal{O}_{T-1})$ = observation sequence.

Figure 3: Hidden Markov Model Notations

In previous research done using hidden Markov models, it is observed to perform well in solving pattern matching problems. Hidden Markov models based on Markov processes of order one and two have been used to detect handwritten letters [18]. Using a selected set of features, this research demonstrates that, hidden Markov model using the Viterbi algorithm produces a model with 93% accuracy rate [19].

Voice recognition is another such research topic, where hidden Markov models are known to have performed quite well [20]. The algorithm for voice recognition is based on HMM and uses features extracted from the vocals available for training the model. Every HMM has two different stages, training and testing, available data is split for each stage. During training, the model is still naive and requires lot of iterations of training to be able to develop a model good enough to test on with good accuracy.

Malware detection, like the problems of voice recognition and character recognition, are basically pattern matching problems. Every malware with a known signature or pattern can be used to train these models. The signature extracted from a malware is fed into this model for it to be able to identify similar files. Once trained, files are tested on this model and scores are computed. Threshold values are set, and then files are classified as malware or benign depending on which side of the threshold they fall [13].

The features chosen as input for the hidden Markov models may also vary in a malware detection problem. There are two basic types of features that can be chosen:

- `Static` — Features extracted from a file not in execution
- `Dynamic` — Features extracted from a file in execution (i.e., runtime)

Opcode sequences, function calls, file entropy, and the size of the file are some examples of such features. It is tougher to extract dynamic features as compared to the static features of any file. However, existing research indicate that hidden Markov models provide better results with dynamic features [21].

2.2 AdaBoost

Boosting is the process of combining many weak classifiers which perform better than a coin flip to create a strong classifier. AdaBoost is a boosting algorithm used to build a strong classifier by combining several weak classifiers [22]. However, this technique requires a sufficient number of classifiers to be effective and the stronger model generated performs better only by a certain amount. AdaBoost is one of the most widely used boosting technique. Adaptive Boosting (AdaBoost) builds subsequent classifiers based on the classifier determined at the current stage.

AdaBoost is a greedy and an iterative approach which identifies the weaknesses of each classifier and chooses the best classifier to mitigate that weakness. As there is no guarantee that the performance of the classifiers always improves, we stop when the results are no longer better than the previous stage. In the Figure 4, we see that every data sample has a classification in every classifier. The strong classifier is created by weighing the weak classifiers to correctly classify every test sample. However, AdaBoost, tends to snowball errors as it is highly sensitive to noise.

Data	Label	Classifiers			
		c_1	c_2	$\cdots$	c_L
X_1	z_1	-1	$+1$	$\cdots$	$+1$
X_2	z_2	$+1$	-1	$\cdots$	-1
X_3	z_3	-1	-1	$\cdots$	$+1$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
X_n	z_n	-1	$+1$	$\cdots$	-1

Figure 4: AdaBoost

There are several applications of such boosting algorithms. One such example is using AdaBoost to combine several classifiers generated using Gaussian Mixture Models to build a strong classifier for Network Intrusion Detection [23]. Another implementation of AdaBoost can be found for improving the selection of features for a vision based application [24].

2.3 Malware detection using Machine Learning

Most of the anti-virus software nowadays use machine learning models to improve their performance. To gauge the performance of any such model, we plot a Receiver Operating Characteristic curve (ROC) curve and calculate the area under the curve [25]. They use their scan engine to check for any files with a pattern matching any of the

malware definitions present in the software. It has been observed that, anti-virus software vendors dedicate great emphasis on reducing the false positive rate of their software. It has been widely reported that, high false positive rates affect the brand reputation of an anti-virus vendor [26]. The matrix in Figure 5 gives the structure of the confusion matrix.

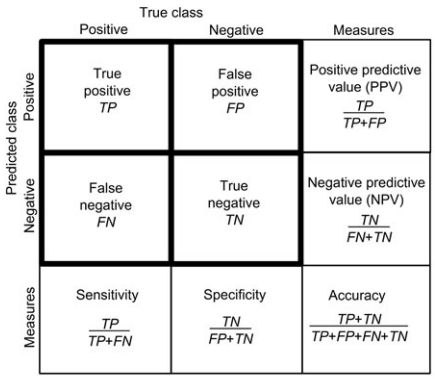

Figure 5: Confusion Matrix [3]

To ensure low false positive rates, the anti-virus software vendors prefer to lower their true positive rate as well. By lowering the true positive rate or the sensitivity of the model, the false negative rate increases thereby also reducing the accuracy of the model. Hence, if we can reduce the false positive rate of any such model using machine learning then, the performance of anti-virus software would improve.

We also address the cold start problem in malware detection, caused due to limited availability of malware samples. Many a malware families, have very few samples of malware available for analysis. This increases the difficulty of extracting a signature from the malware and training a model with sufficient data to identify the malware. Improving the classification of such malware with limited samples, will help mitigate the vulnerability of a system to lesser known malware.

CHAPTER 3

Implementation

Combining classifiers to achieve better results has been found to work very well in several pattern matching problems [15, 27]. We employ a similar technique of boosting a set of weak classifiers of hidden Markov models using AdaBoost to improve malware detection.

3.1 Hidden Markov Models with Random Restarts

As mentioned in Section 2.1, we train an HMM on static and/or dynamic features extracted from a set of sample data and classify files based on a score generated using such models. The traditional method used for such pattern recognition problems using HMM involves random restarts. The re-estimated values of the model, during training, plateau after a while. Training the model further after this does not help improve the model.

Random restart specifies repeating the training stage for HMM model using different initial values for initial state distribution and state transition matrix. We train a model, score the files after training and then repeat the process with different initial values to the model as the previous model. We compare the performance of all models obtained using random restarts and choose the one with the best performance. However, the rest of the models trained and used for scoring are not considered any further. Although, random restarts help us find the best classifier over a range of initial values, the training effort for all models except the best performing model chosen for classification is ignored. We look to take advantage of these discarded models for improving the classification as compared to the random restart technique.

3.2 Combining Classifiers

Considering a set of classifiers for any generic pattern matching problem, the observed mismatch in the set of misclassified patterns does not always overlap [27]. This indicates that, the mismatch caused in each classifier might be different for different set of patterns. Hence, there is an opportunity to combine these classifiers and improve the classification using machine learning techniques like boosting.

3.3 Boosted Hidden Markov Models

Boosted hidden Markov models are created by combining multiple hidden Markov models using AdaBoost. This technique has been previously found to have produced promising results for creating an intrusion detection system [15]. A boosted hidden Markov model is built for improving the performance of an intrusion detection system. A variant of the HMM called incremental HMM (iHMM) is used to reduce training time before combining these multiple classifiers with AdaBoost [15]. A sequence of events for every user is recorded and the iHMM uses this information to train itself.

All these models are trained individually and independently of each other. These models vary due to random restarts and number of iterations of training they have been subject to. These models are then individually asked to classify the various scenarios to detect the intrusion of an unknown user into the system [15]. The models on their own do not perform as well as when they are combined using AdaBoost. This follows the methodology of combining several weak classifiers to build a strong classifier. This is achieved using AdaBoost and the several incremental hidden Markov models used are the weak classifiers.

A similar setup is followed in this research for improving malware detection. A dataset containing sequence of opcodes for different malware families are used to build models [1]. Multiple hidden Markov models are trained on a sequence of opcodes. These HMM are then used to score the test files and these scores are fed to the AdaBoost boosting algorithm to generate an improved classification.

On every iteration of the boosting algorithm, a weak classifier is identified and added to build a stronger classifier. Depending on the classification of the malware, these weak classifiers are assigned weights and combined appropriately to obtain better classification from the resultant classifier. Such boosting techniques can be used to resolve the false positive rate and cold start issue in the field of malware detection.

CHAPTER 4

Experiments and Results

To compare the malware detection techniques using the traditional method and boosting techniques, we performed a set of experiments on the Malicia dataset [1]. Table 1 lists the different malware families in the Malicia dataset.

Table 1: Malicia Dataset [1]

Malware Family Name	Malware Type	File Count
Cridex	Trojan	74
Harebot	Backdoor	53
Security Shield	Spyware	58
SmartHDD	Trojan	68
Zbot	Trojan	2316
ZeroAccess	Trojan	1305

We use the dataset to perform classification using both traditional and boosting techniques. We create multiple hidden Markov models using random restarts, train each model and score the models. We consider the area under the curve of a ROC as a parameter to gauge the performance of the model [25].

Boosting technique is used to classify the malware by combining all the models to obtain a final classification from the boosted model. We perform a 5-fold cross validation on the results obtained using traditional technique and measure the robustness of both the techniques by morphing the training data by 10%, 50% and 100%. We later consider the application of this technique to address the cold start problem in malware detection, by limiting the amount of data available for training the model.

4.1 Initial Experiments

Each of the malware family in the dataset is split into two sections of training data and scoring data. We randomly pick 60% of the total number of samples for each family and use it for training the models. The remaining 40% is used for scoring purposes. These files contain a stream of opcodes extracted for the original malicious files. We generate a long sequence of opcodes from all the chosen training files to be given as observation sequence for our multiple hidden Markov models.

We modify this string of opcodes before feeding it to the model. Only the top thirty opcodes from this sequence are considered. Rest all opcodes are simply considered as other opcode and is replaced by OTHER in the sequence. For all malware families, the top thirty opcodes amount for over 90% of the total training data as seen in Table 2. Hence, it helps us optimize the training phase by also reducing the dimensions of the observation probability matrix i.e. $N = 2$, $M = 31$ for every model.

Table 2: Top Opcodes Frequency

Malware Family Name	Top 30 Opcodes Percentage
Cridex	94.8%
Harebot	94.3%
Security Shield	96.3%
SmartHDD	99.99%
Zbot	93.7%
ZeroAccess	96.1%

For the boosted hidden Markov models, we use the models generated by random restarts and combine to possibly generate a stronger classifier. The classification for each of the models are given as inputs for the AdaBoost boosting algorithm. It considers the threshold value for each model and the scores for each tested file. This

algorithm, then produces a score for each of these test files and that classification is considered to be the final classification using boosted hidden Markov models. The benign files used for every experiment are the same and are also obtained from the Malicia dataset [1]. We compare the results of both techniques by plotting a ROC curve and measuring its area under the curve.

As shown in Figure 6, for the Cridex malware family, using the traditional random restarts technique, the best classification we obtain has an area under the curve (AUC) of its ROC curve of 0.583. However, using the boosting techniques on these models the classification improves the AUC by 6%. In case of Harebot malware, there is no difference in the performance of boosted technique versus the random restart technique. For SmartHDD malware family, a clear separation can be seen between the scores of malware files and benign files. This results in an AUC of 1 as shown in Figure 7 and gives us the perfect classification using the traditional model.

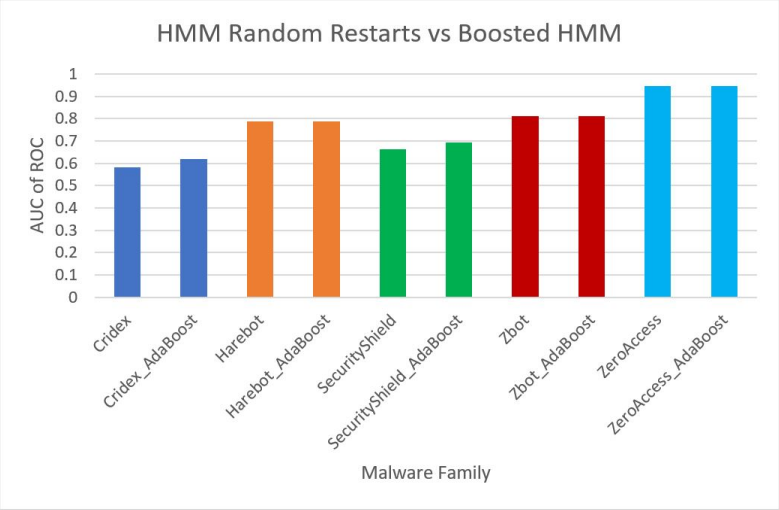

Figure 6: AUC of ROC for malware families

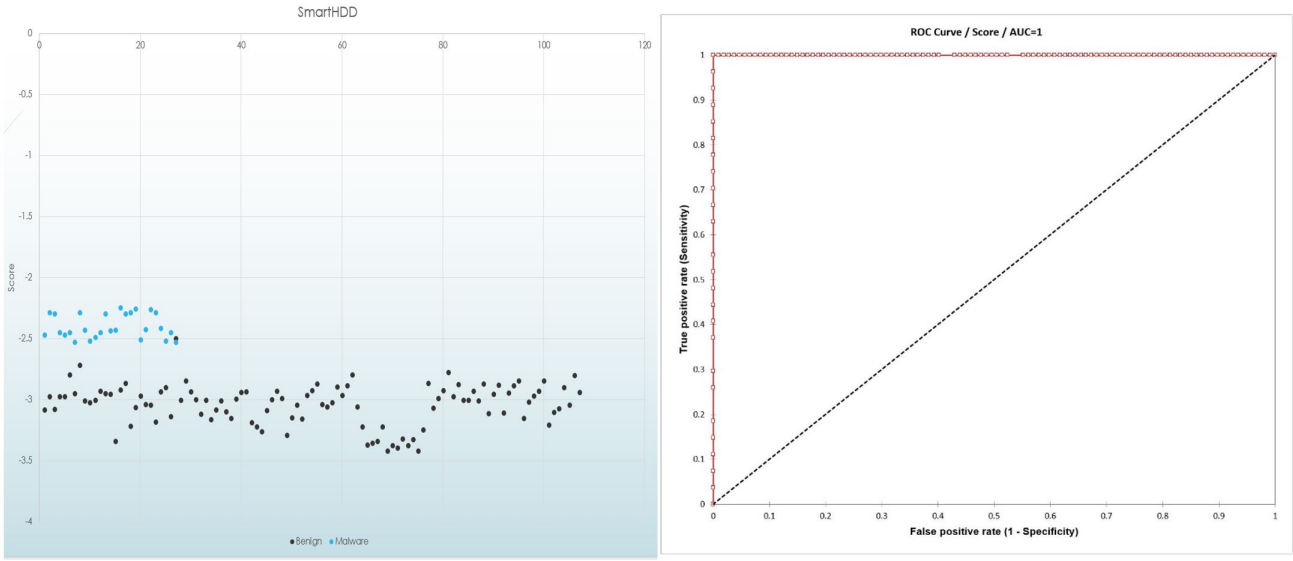

Figure 7: SmartHDD Classification

We see an improvement of 5% in AUC for the classification using the boosting technique for Security Shield. Zbot is one of the biggest malware families in this dataset. The performance of boosting technique for this malware does not provide any improvement and a similar case is seen for ZeroAccess malware as well.

From this set of experiments, we see that although boosting does not improve the classification for all the malware families, it does perform better than the traditional technique of random restarts for some of families. We also observe that, for the families where boosting does not improve the classification, we already have a strong classifier using the random restarts technique. Boosting is applied to weak classifiers and the resultant classifier performs better than the traditional technique. This indicates that for any malware family which has weak classifier using the traditional technique, we can improve the classification using boosting.

4.1.1 Cross Validation

Some of the malware families included in the dataset [1] are not large in number. This limited availability of data samples might result in inclusion of bias due to improper partition of data. In such cases, the availability of more data helps remove or reduce these biases. This is achieved using cross validation [14].

We split the malware data set into equal subsets and train the model using all but one subset which is used for testing purposes. We repeat this and ensure each subset is used as scoring data set once. This provides us multiple models which can then be combined using boosting techniques. Such cross-fold validation smooths out any bias that may be present in the match data (malware data). We perform 5-fold cross validation, thereby breaking down every match set (each malware family) into 5 equal subsets and check the performance of the boosted hidden Markov models.

Splitting the Cridex dataset into 5 separate subsets, we do a 5-fold cross validation which gives an AUC of ROC curve of 0.554 as shown in Figure 8. The AUC decreases when compared with the initial experiment with no morphing, however, it is considered a more accurate classification as it removes the biases introduced when choosing the training data.

On 5-fold cross validation of Harebot malware family, we get an AUC of ROC curve of 0.518 as shown in Figure 9 which is not much better than the probability of a coin toss. Hence, this indicates that the model obtained is very poor and not reliable for classification of Harebot malware with the data available. The model obtained for Security Shield malware performs very poorly as we get an AUC of ROC curve of 0.509 as shown in Figure 10. For the SmartHDD malware dataset, we almost get a clear separation for the model with an AUC of 0.998 as shown in Figure 11.

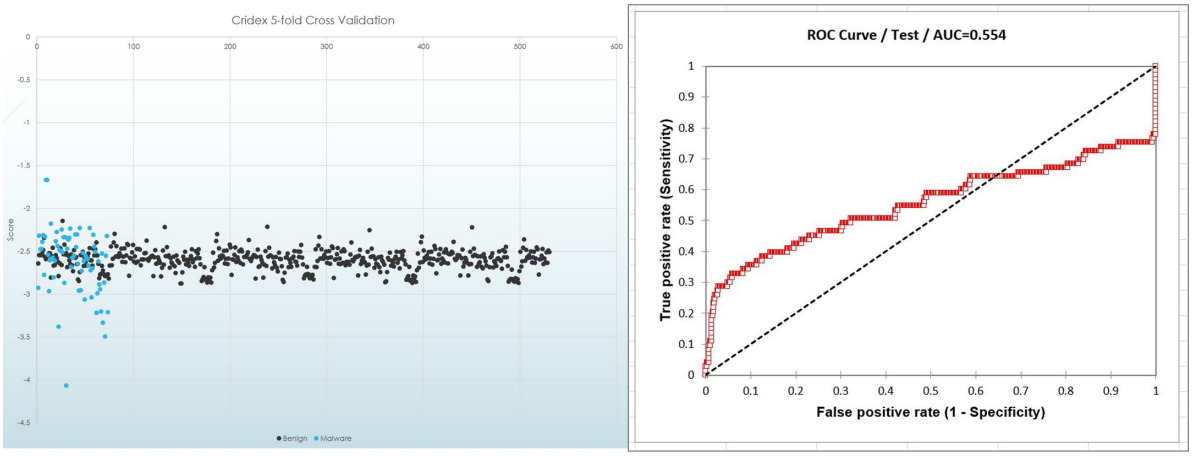

Figure 8: Cridex 5-fold Cross Validation

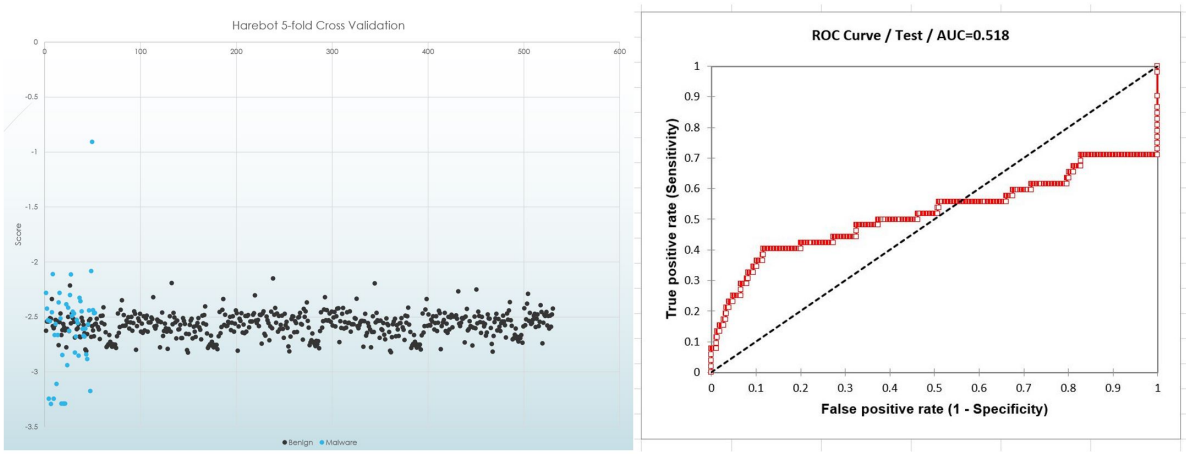

Figure 9: Harebot 5-fold Cross Validation

For Zbot malware family, on 5-fold cross validation we get an AUC for ROC curve of 0.770 as shown in Figure 12 and on cross validation of ZeroAccess malware, we get an AUC of ROC curve of 0.90 as shown in Figure 13. There is a correction in the AUC values of ROC curve of each malware family on 5-fold cross validation. This ensures that any bias that could've been included because of random selection of training and scoring data would be removed. It is imperative to use such techniques wherever applicable to get a more accurate classification.

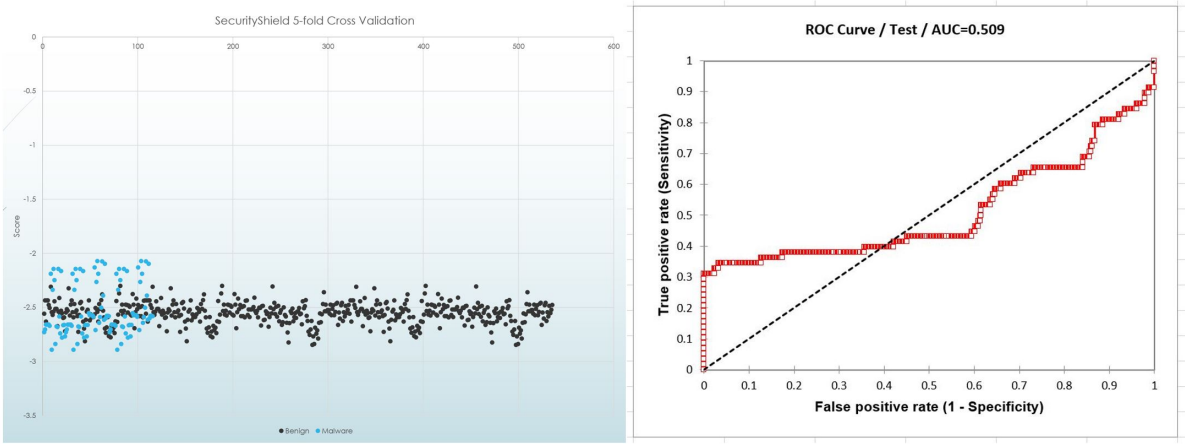

Figure 10: Security Shield 5-fold Cross Validation

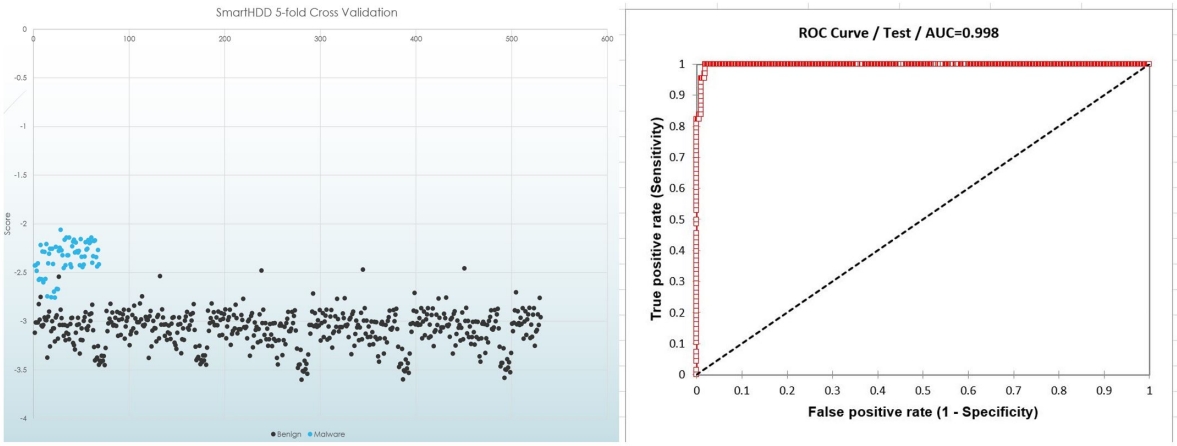

Figure 11: SmartHDD 5-fold Cross Validation

4.2 Morphing

To further compare and verify the performance of boosted hidden Markov models versus the traditional technique we morph the data up to different degrees and measure the performance of both techniques. Morphing provides an indication of the robustness of the techniques employed. The benign opcode sequence is concatenated at the end of the original training data sequence. Experiments were performed for three different degrees of morphing: 10%, 50% and 100%. This degree indicates that for the first set

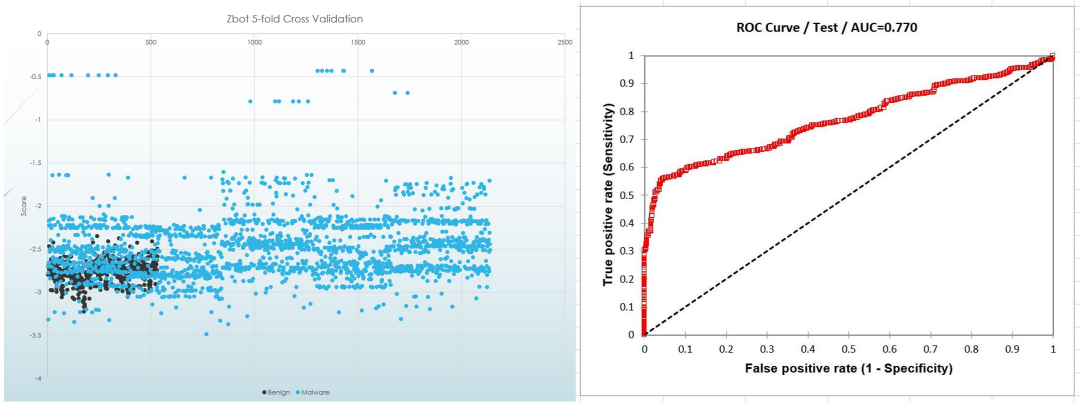

Figure 12: Zbot 5-fold Cross Validation

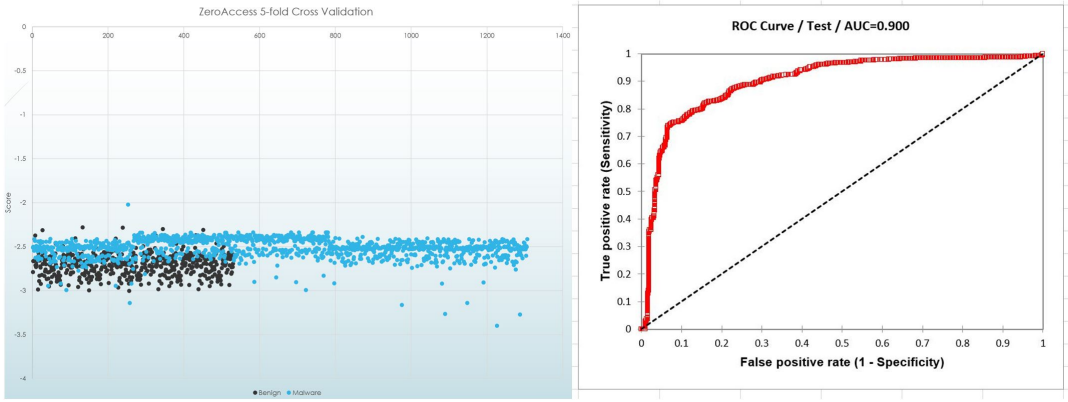

Figure 13: ZeroAccess 5-fold Cross Validation

of experiments, the training data was morphed by adding benign opcode sequence of the size equivalent to 10% of original training data size.

Morphing is one of the most common techniques employed by malware developers to defeat such signature detection techniques. We measure the performance of boosting technique against the traditional techniques when such tactics are implemented to avoid detection. For every malware family, we categorize the files in training data and scoring data. The split is same as that in the previous experiments - 60:40. We add benign files to the training dataset such that appropriate percentage of morphing is enforced.

4.2.1 10% Morphing

For this experiment, we concatenate benign code at the end of the training data which is equal to 10% of its actual size. Generally, higher degree of morphing is applied with more sophisticated techniques. However, we use this simple experiment to measure the performance of the techniques over a range of degree of morphing as seen in Figure 14.

For Cridex malware, the boosting technique out performs the traditional technique. There is an 8% increase in the AUC using the boosting technique. With 10% morphing, boosting provides a better classification than the random restart technique for Harebot malware. The improvement in classification between the two techniques also increases by 4,5%.

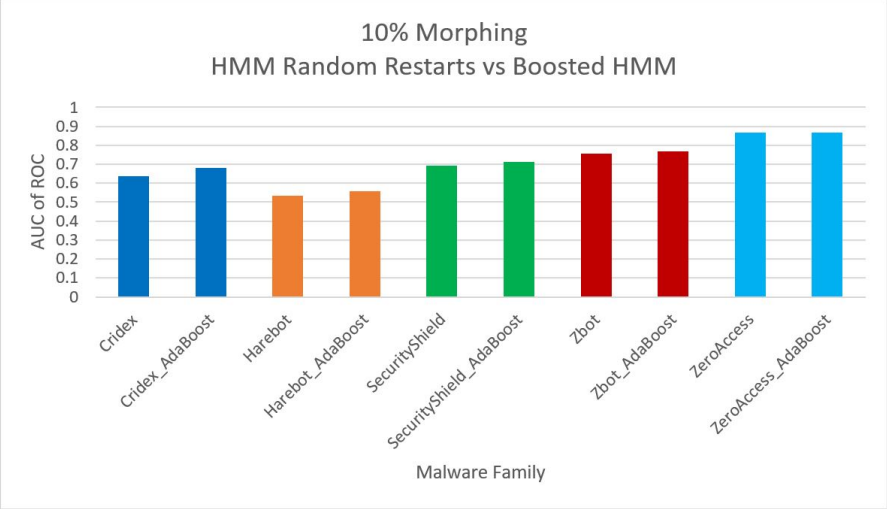

Figure 14: Malware Families with 10% Morphing

Just like in the previous malware families, the boosting technique out performs traditional technique for Security Shield with an 2.5% improvement on the AUC. Zbot malware classification sees a 1.5% improvement in the AUC using boosting compared to random restarts technique. For ZeroAccess malware, as seen in the initial

experiment, boosting does not improve the classification provided by the traditional technique. We can attribute this to the fact that, with this minimal morphing, the classifier obtained from the traditional techniques are still strong classifiers. Hence, boosting does not really perform well when any of the classifiers are strong.

4.2.2 50% Morphing

For the next set of experiments, we increase the amount of morphed training data to 50%. A long sequence of benign opcodes, half the size of the actual training data, is appended at the end of the training data. This increased morphing degree, helps us verify if using techniques like adding dead code can help defeat the techniques being analyzed 15. For malware families of Cridex, Security Shield and Zbot there is no improvement in the AUC Boosting technique as compared to random restarts technique. An improvement of 5.8% and 16% AUC of ROC is seen for Harebot and ZeroAccess malware respectively.

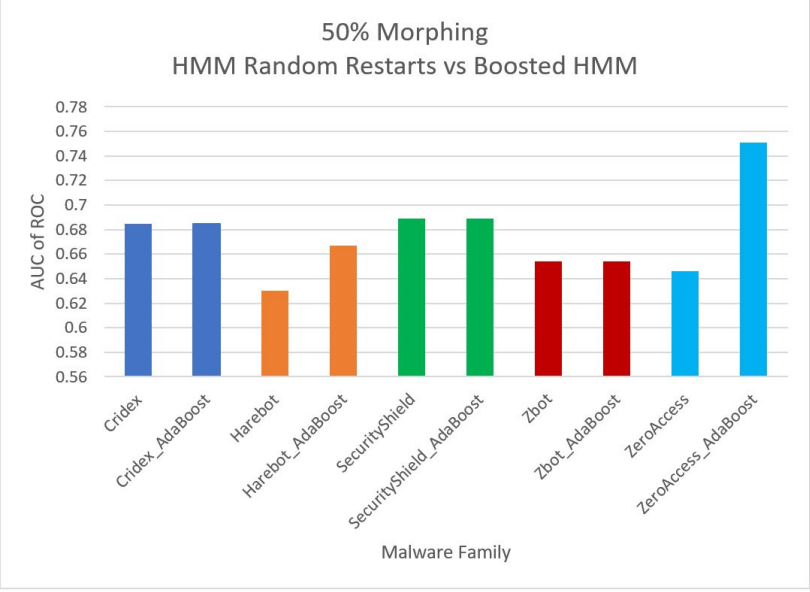

Figure 15: Malware families with 50% Morphing

4.2.3 100% Morphing

To test the robustness of suggested techniques, we ramp up the degree of morphing to 100%. The benign opcode sequence which is attached at the end of the original training data, is equal to the size of the actual training data. This doubles the size of the training data and makes it much more difficult to detect signatures in the test files. The classification does not improve for Cridex and Security Shield. There is however, an improvement seen in the classification for other malware families as shown in Figure 16.

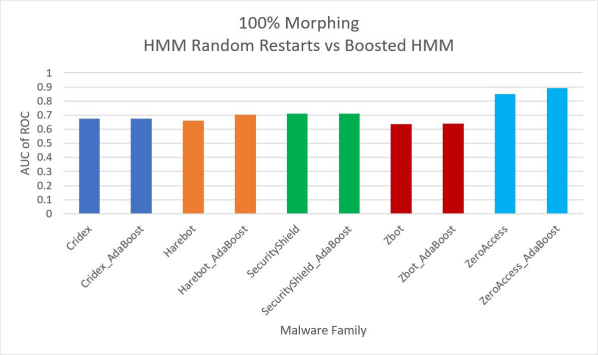

Figure 16: Malware families with 100% Morphing

4.2.4 Summary

The classification is best for most families with no morphing of the training data. Morphing worsens the performance of classifiers based on signature detection for some malware. However, using boosting the performance of these models is still better than the models using traditional techniques. Although, for models with strong classifiers obtained using random restarts, the boosting techniques do not perform as well. In most cases, the AUC of ROC curve is highest for models using boosting with no morphing. Even the slightest amount of morphing (10%) affects the performance of the classifiers almost as much as training data with 50% or 100% morphing.

4.3 Cold Start Problem

Even with a huge volume of malware in the digital world, the availability of information on every malware family is not possible. If there is a lack of malware samples for any type of malware, it is difficult to build effective models to train and identify the malware. Machine learning techniques like HMM, need a significant input for training to build a good model. The cold start problem results in generation of poor classifiers. In general terms, lack of sufficient training data is a major issue for generating signature detection models.

We may however, resolve this problem using a boosting technique which will help improve the classification of weak classifiers generated using the limited training data. To test the boosted HMM in this context, we consider a set of experiments on the malware dataset by restricting the amount of training data over a range of values from five to twenty-five. We generate many models using random restarts and train them on the limited training data. We then score the test files and applying boosting to these classifiers to try and improve the classification with limited training.

4.3.1 Varying training data size

Experiments were performed choosing a range of files randomly to train and build a thousand weak classifiers for each of the malware family. These models generated using random restarts were then subject to thousand iterations of boosting to build a strong classifier. The files chosen for both set of experiments did not have any overlap. We see that, for every malware family, there is an improvement in the classification using boosting. The best available classifier is chosen at every iteration of boosting to reduce misclassifications. We can choose the best performing model from thousand iterations as sometimes the performance of the model worsens on further boosting

depending on the weak classifier being combined with it. We plot a graph containing the accuracy values for each weak classifier and compare it with the accuracy of the strong classifier generated after each iteration of boosting.

Models were trained initially only on five files for each malware family. Similar experiments were conducted for ten, fifteen, twenty and twenty-five files each. The results were summarized by plotting bar graph for each malware family comparing HMM with random restarts versus boosted HMM. We also compare the difference in AUC of ROC for these techniques.

In Figure 17, for Cridex we see an improvement in accuracy of 0.73% using just 5 files for training the model. Harebot on the other hand sees an improvement of 1.6% in accuracy as show in Figure 18. An improvement of similar magnitude is seen for Security shield in Figure 19.

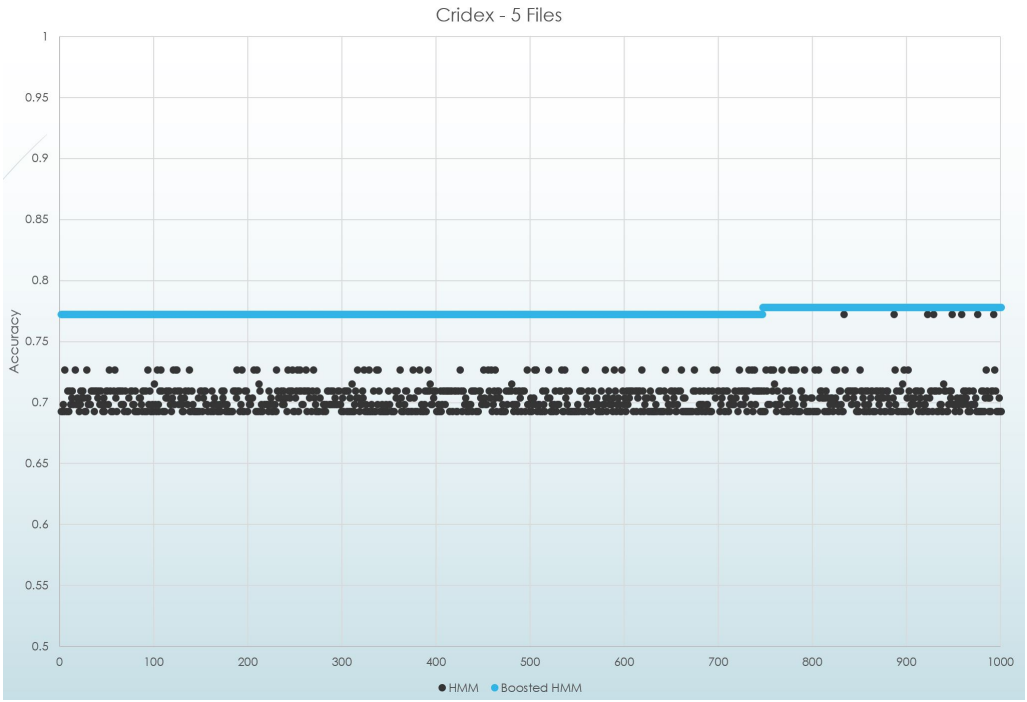

Figure 17: Cridex Cold Start - 5 Files

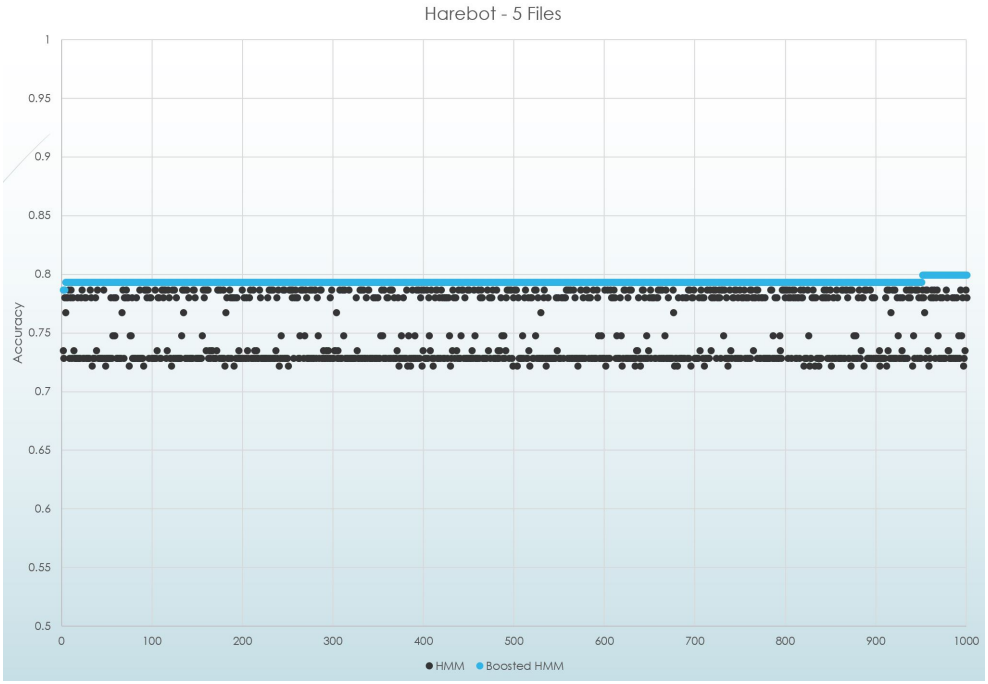

Figure 18: Harebot Cold Start - 5 Files

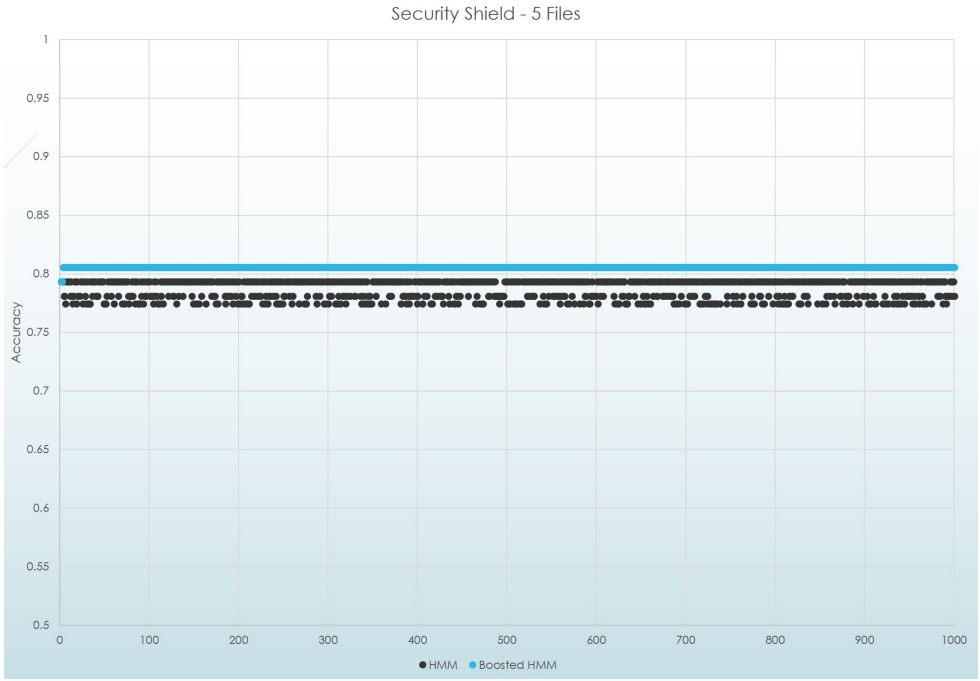

Figure 19: Security Shield Cold Start - 5 Files

As only 107 benign files are used throughout the project, using an imbalanced dataset affects the accuracy and the AUC of the ROC curve of the models for that family. Only two hundred files were considered for each of Zbot and ZeroAccess to avoid this imbalance in dataset. Zbot shows the biggest improvement of 3.7% in its accuracy using boosted HMM as seen in Figure 20. However, no improvement is seen in the accuracy for ZeroAccess as shown in Figure 21.

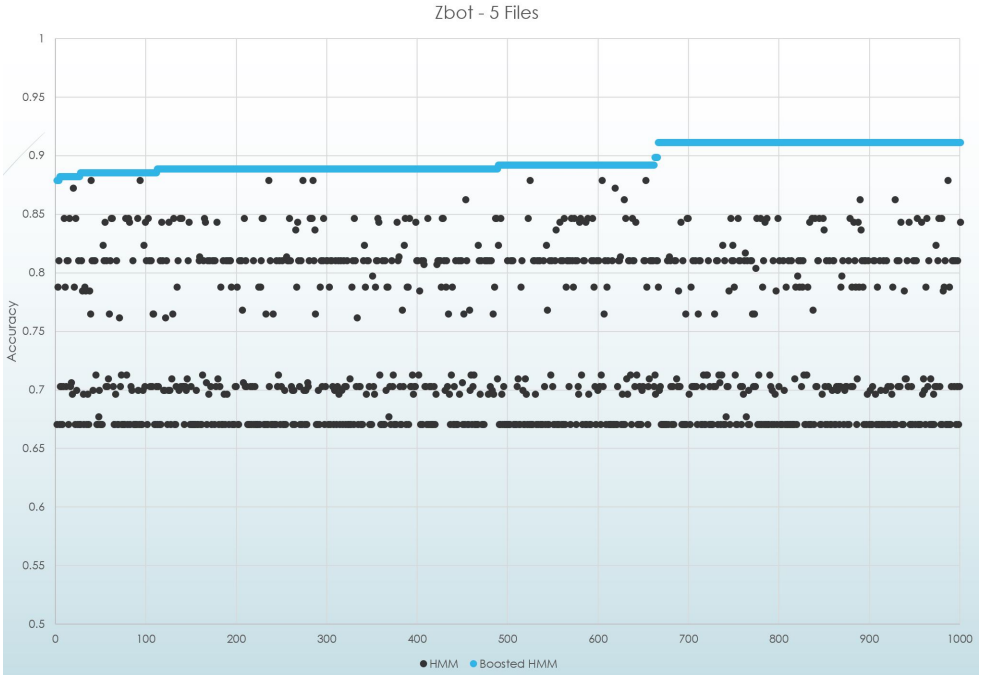

Figure 20: Zbot Cold Start - 5 Files

We can see that from Figure 22, the accuracy for all but one of the malware family improves by varying percentages on using boosted HMM when only five files are available for training the HMM with random restarts. Also, on comparing the AUC for the ROC curve of the models generated using these techniques, an improvement is seen in all malware family whose accuracy improved as shown in Figure 23.

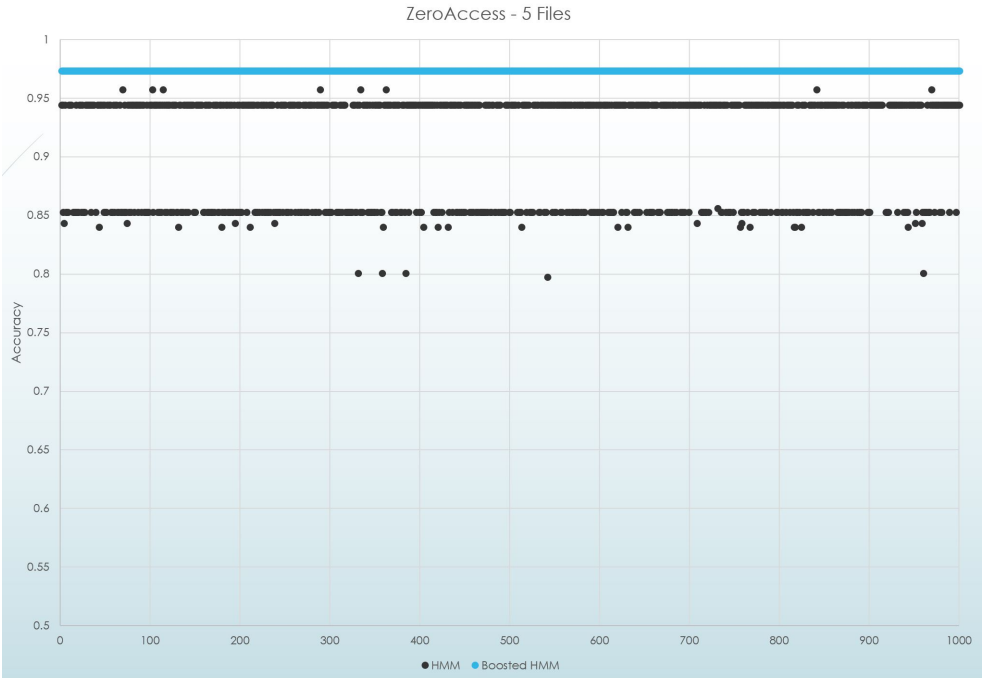

Figure 21: ZeroAccess Cold Start - 5 Files

Figure 22: Cold Start - 5 Files Summary

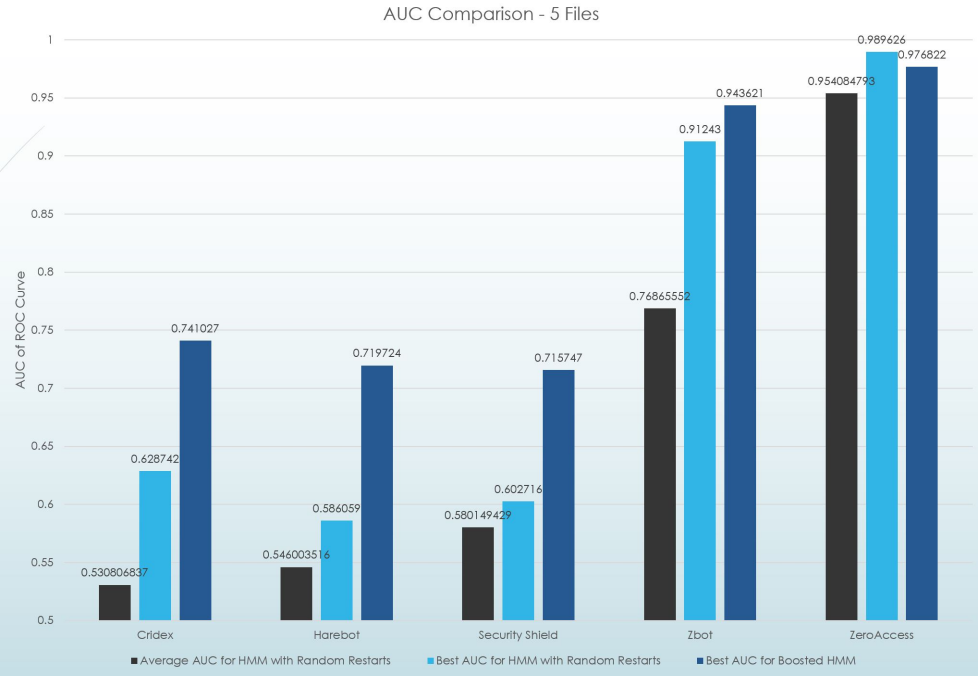

Figure 23: Cold Start - 5 Files AUC for ROC comparison

In Figure 24, for Cridex we see an improvement in accuracy of 4.28% considering just 10 files for training. Boosting also improves the classification for Harebot by 0.7% as shown in Figure 25. However, there is no improvement seen for Security Shield as seen in Figure 26.

Similar to the previous experiment for Zbot and ZeroAccess, only 200 files are considered for scoring to avoid an imbalanced dataset. A significant improvement of 3.4% is seen in accuracy for Zbot as shown in Figure 27. However, no improvement is seen in the accuracy for ZeroAccess family as seen in Figure 28.

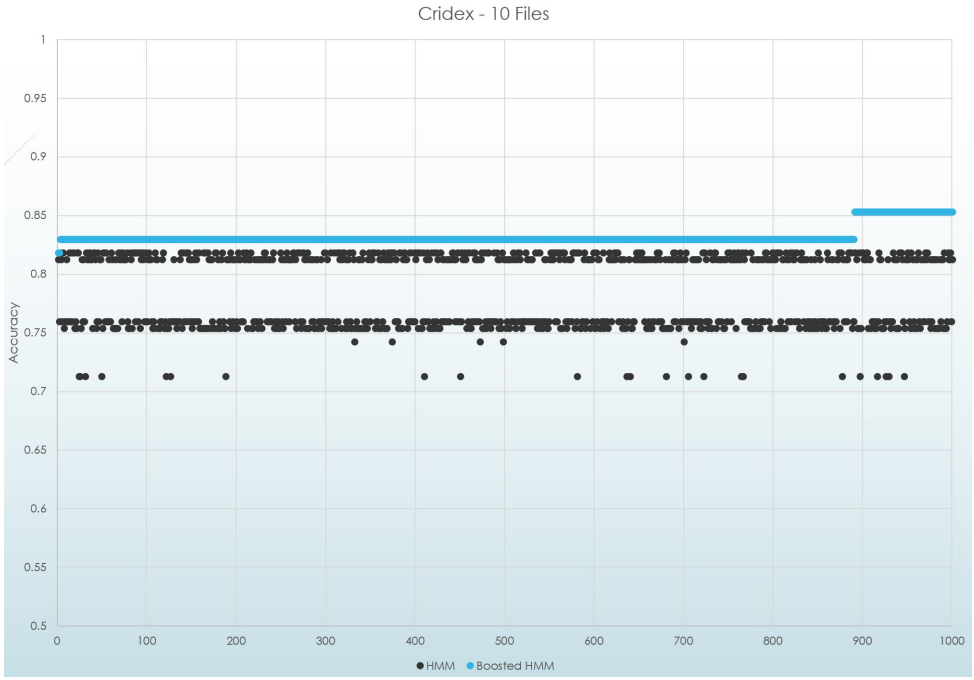

Figure 24: Cridex Cold Start - 10 Files

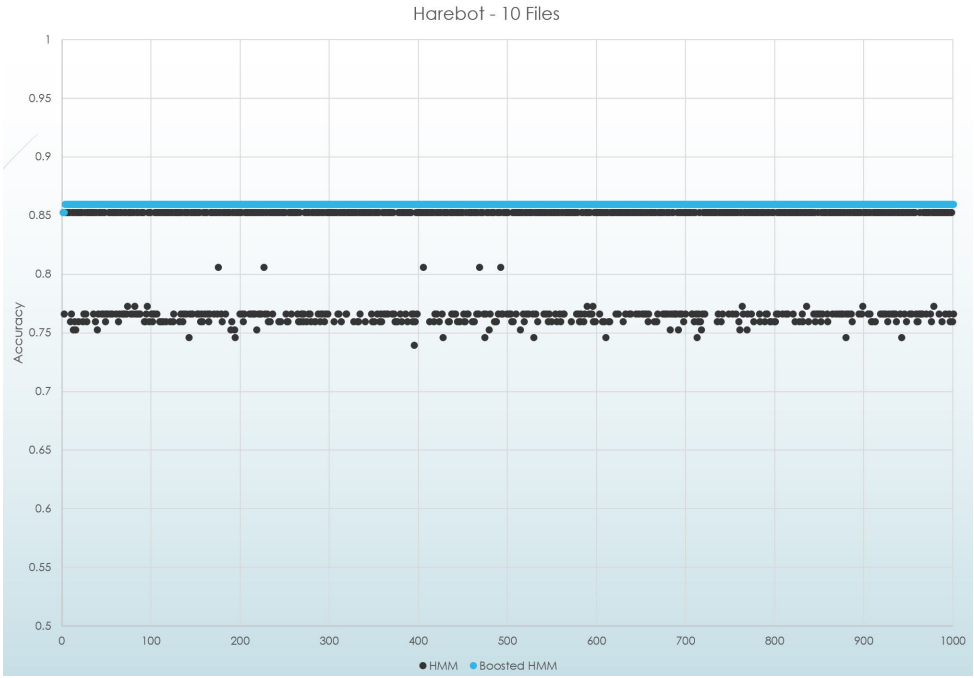

Figure 25: Harebot Cold Start - 10 Files

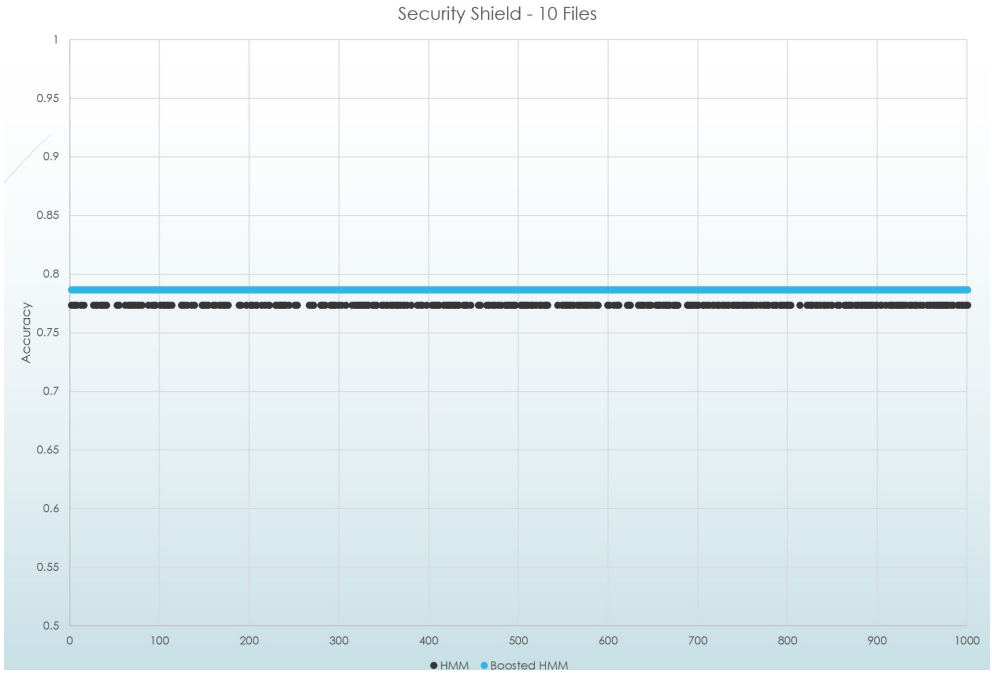

Figure 26: Security Shield Cold Start - 10 Files

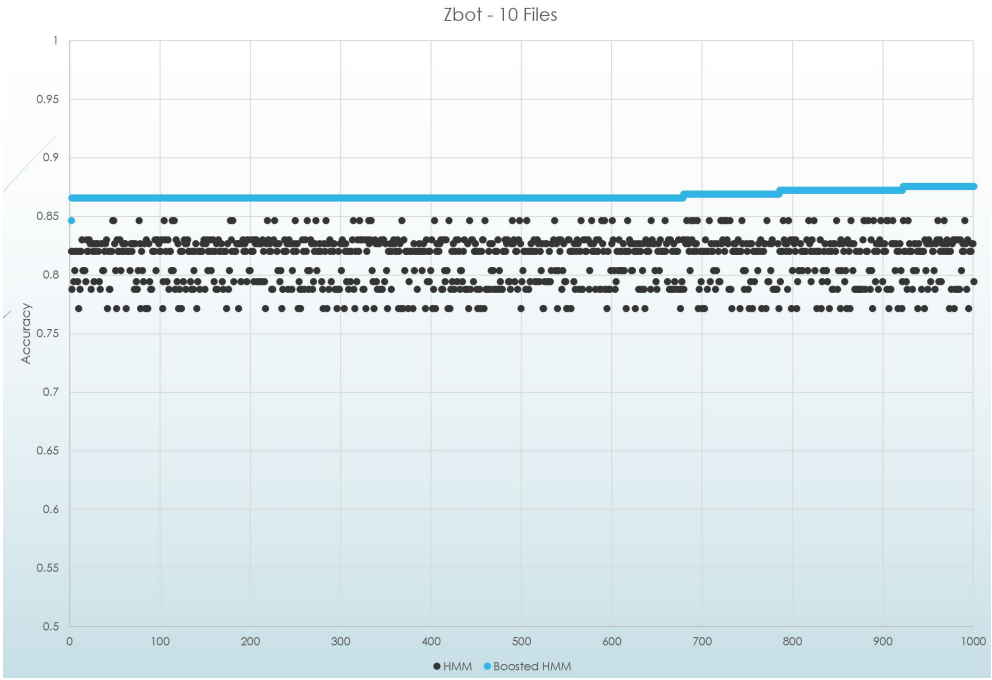

Figure 27: Zbot Cold Start - 10 Files

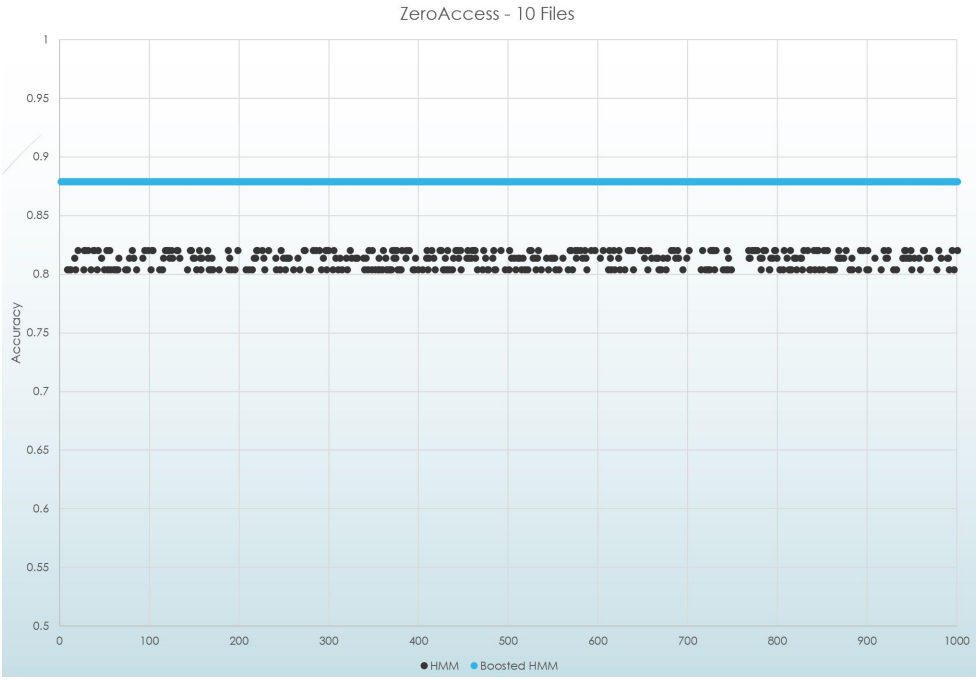

Figure 28: ZeroAccess Cold Start - 10 Files

The Figure 29 provides a comparison of the performance of both techniques for each of the malware family using only 10 files for training. On comparing the AUC of the ROC curve for both the techniques, we see that for all malware families except ZeroAccess, there is a significant improvement in the AUC using boosted HMM. In Figure 30, we see that the best case of boosted HMM outperforms both the average case and the best case for HMM with random restarts.

Using 15 files for training, we see an improvement in accuracy for every malware family except Security Shield as seen in Figure 33. There is a 4.5% improvement in accuracy for Cridex as seen in Figure 31 and a 0.8% improvement for Harebot as shown in Figure 32.

Figure 29: Cold Start - 10 Files Summary

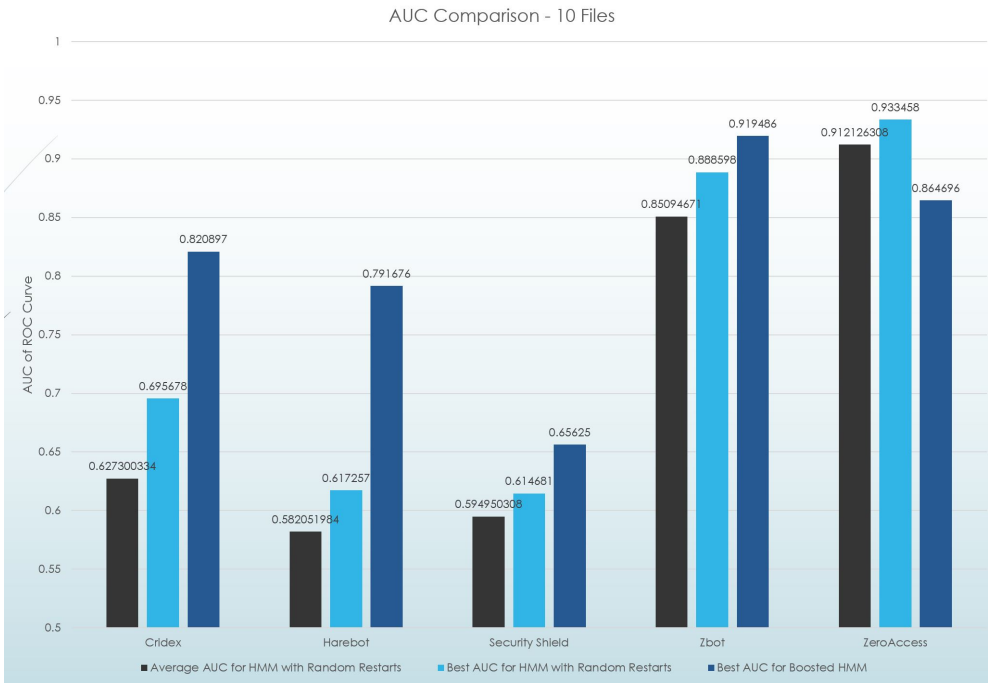

Figure 30: Cold Start - 10 Files AUC for ROC comparison

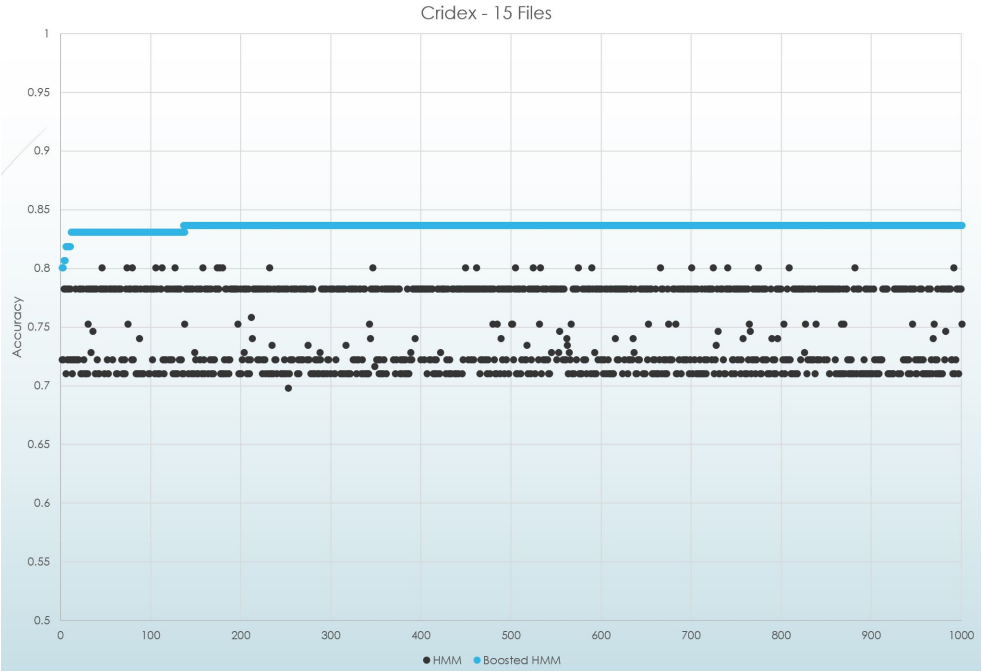

Figure 31: Cridex Cold Start - 15 Files

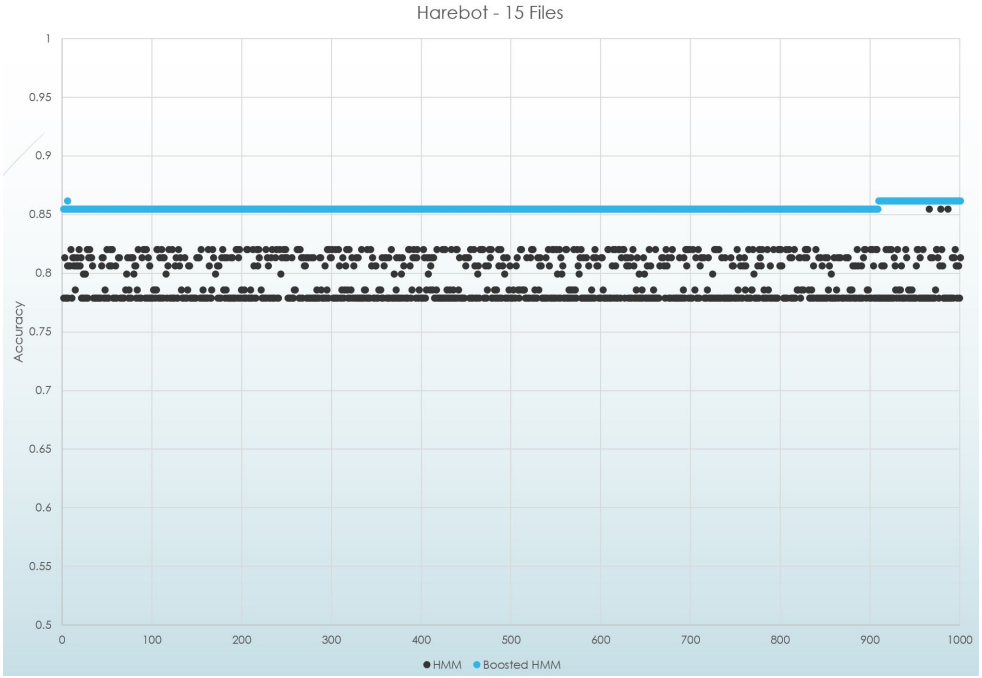

Figure 32: Harebot Cold Start - 15 Files

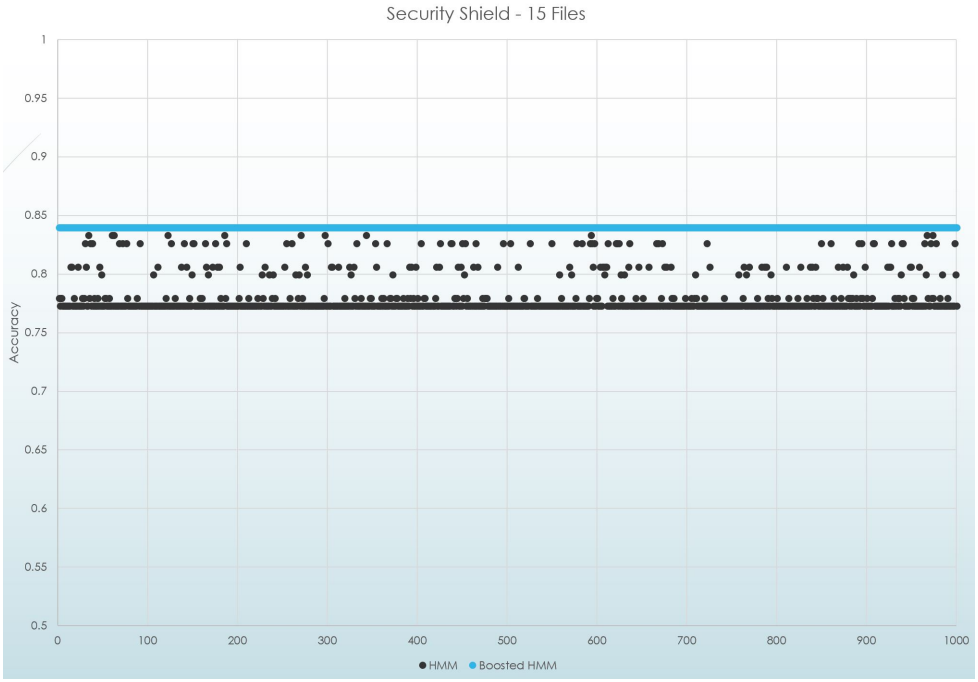

Figure 33: Security Shield Cold Start - 15 Files

Considering only 200 files for scoring Zbot and ZeroAccess, to avoid an imbalanced dataset, we see a significant improvement of 5.6% in accuracy for Zbot as shown in Figure 34. However, the accuracy for ZeroAccess family improves only by 0.33% as seen in Figure 35.

The Figure 36 provides a comparison of the performance of both techniques for each of the malware family using only 15 files for training. On comparing the AUC of the ROC curve for both the techniques, we see that for all malware families except ZeroAccess, there is a significant improvement in the AUC using boosted HMM which is similar to what was observed in previous experiments. In Figure 37, we see that the best case of boosted HMM again outperforms both the average case and the best case for HMM with random restarts.

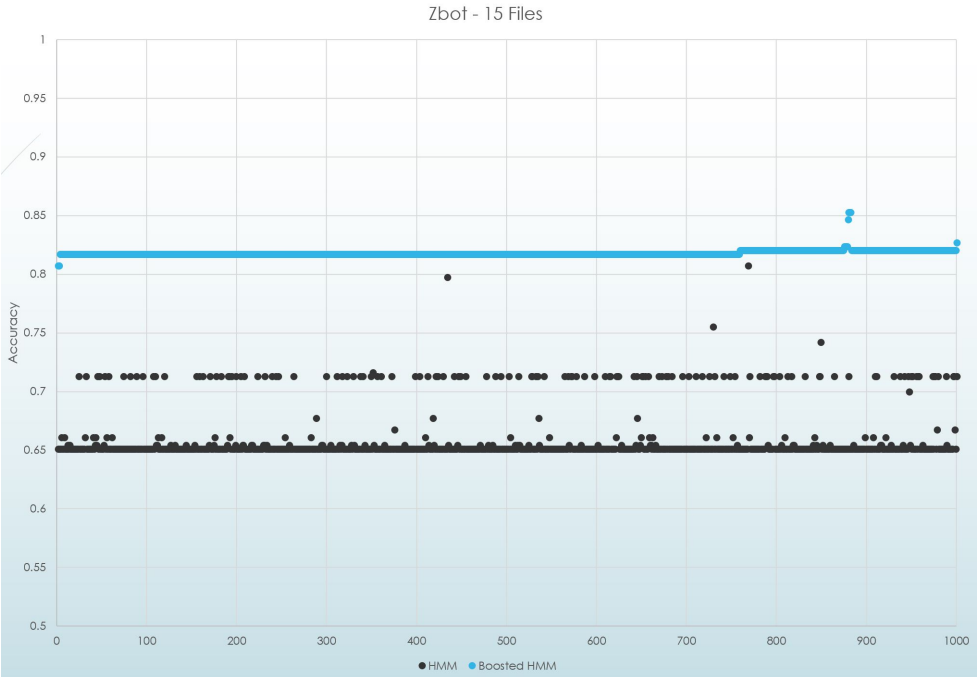

Figure 34: Zbot Cold Start - 15 Files

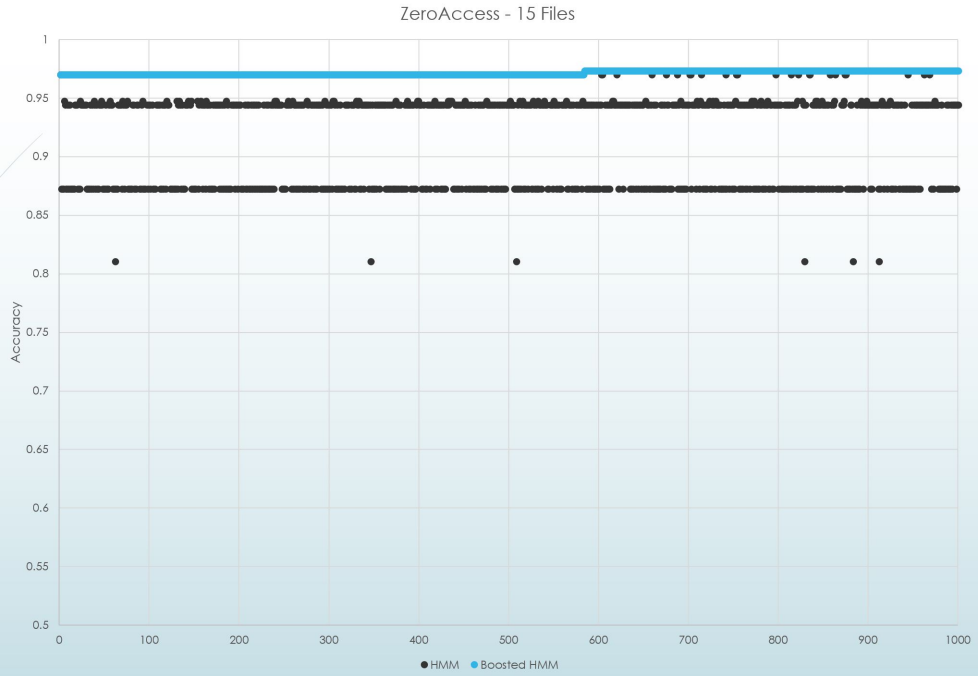

Figure 35: ZeroAccess Cold Start - 15 Files

Figure 36: Cold Start - 15 Files Summary

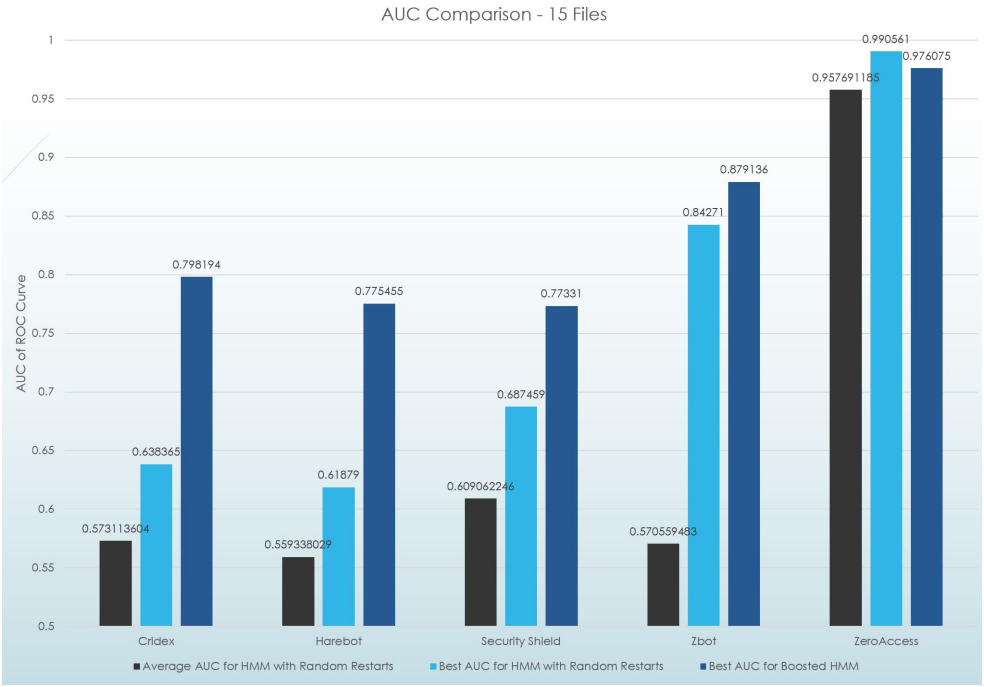

Figure 37: Cold Start - 15 Files AUC for ROC comparison

For next set of experiments, we use 20 files for training the models using both HMM with random restarts and then using those models to boost the scores and generate a stronger classifier. For the first time, we see an improvement in accuracy with each malware family. In Figure 38, we see an improvement in accuracy for of 0.73% and an improvement of 1.6% for Harebot as seen in Figure 39. There is a 0.79% improvement in accuracy for Security Shield as seen in Figure 40.

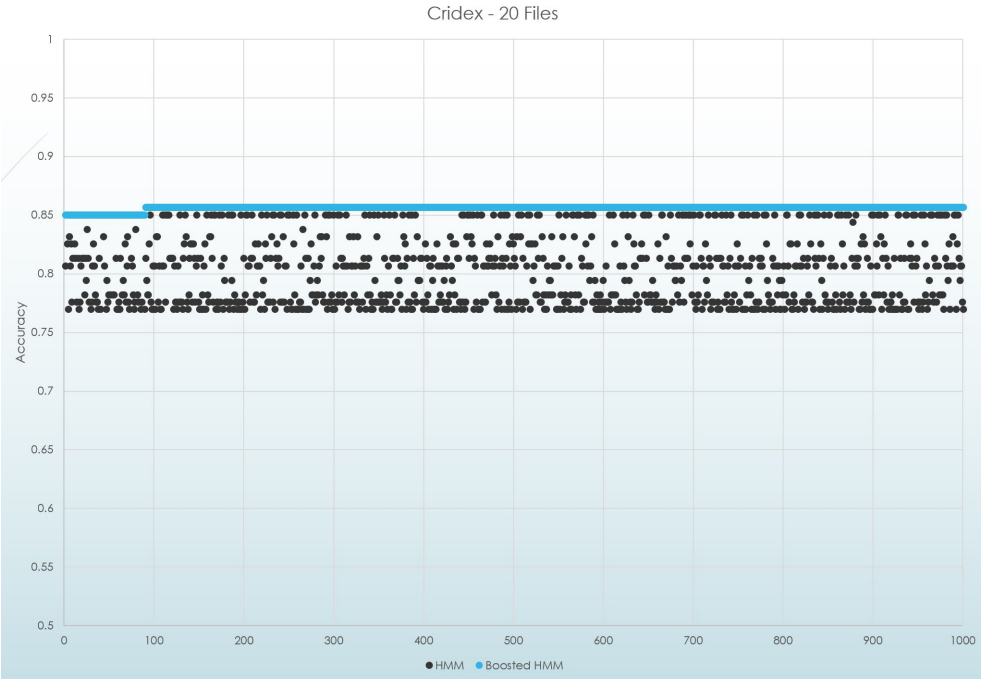

Figure 38: Cridex Cold Start - 20 Files

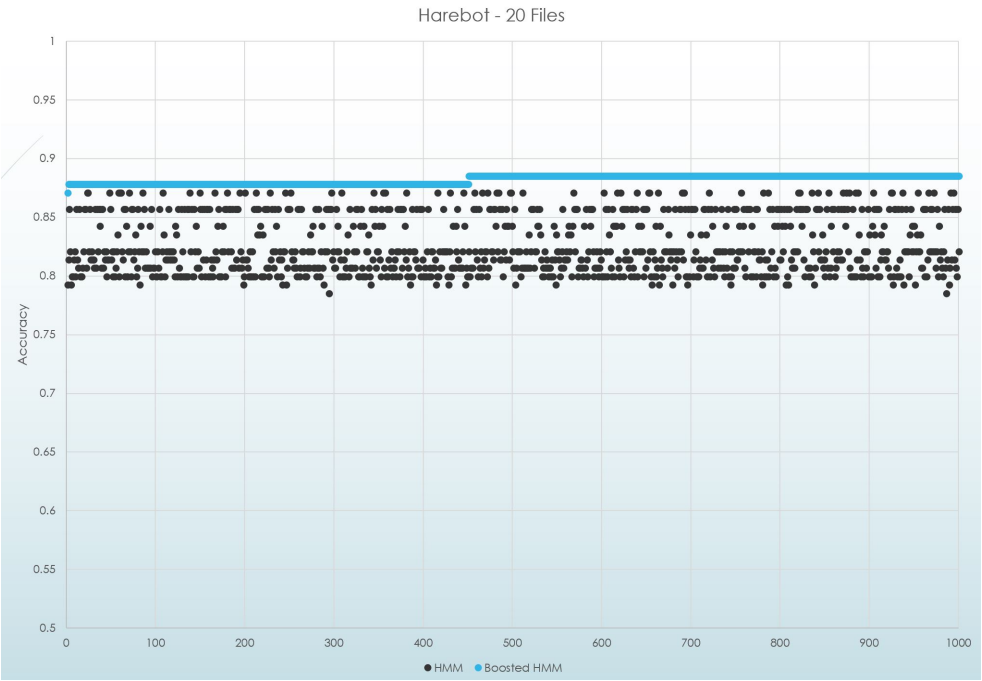

Figure 39: Harebot Cold Start - 20 Files

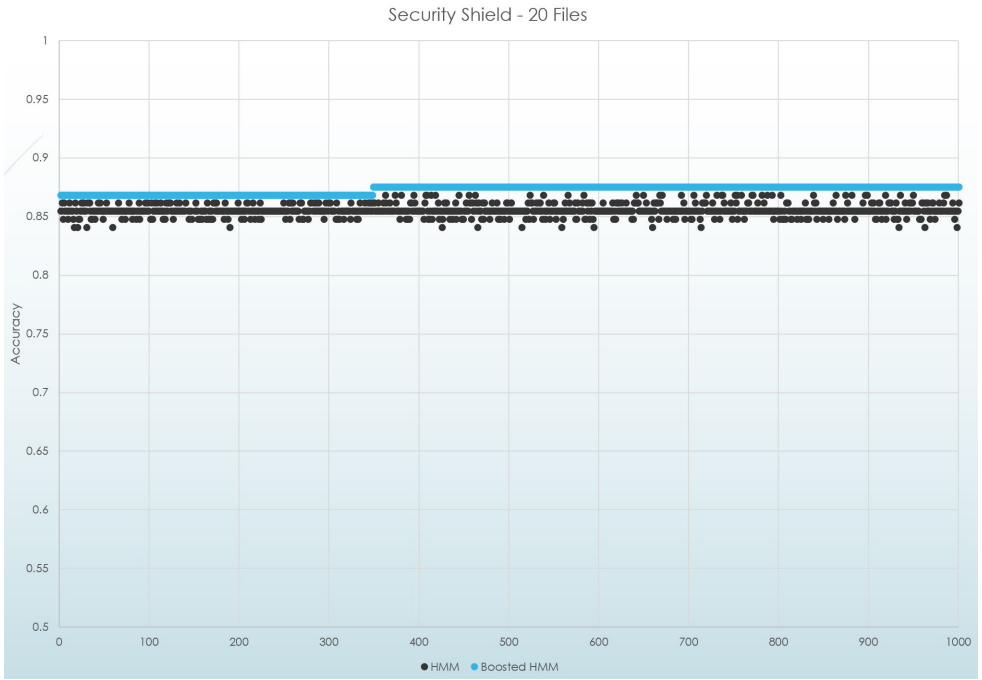

Figure 40: Security Shield Cold Start - 20 Files

We see an improvement of 3.24% in accuracy for Zbot as shown in Figure 41 and the accuracy for ZeroAccess family improves by 0.68% as seen in Figure 42. For these two families, only 200 files are considered for scoring to avoid an imbalanced dataset.

The Figure 43 provides a comparison of the performance of both techniques for each of the malware family using only 20 files for training. On comparing the AUC of the ROC curve for both the techniques, we see that for all malware families except Security Shield and ZeroAccess, there is a significant improvement in the AUC using boosted HMM which is similar to what was observed in previous experiments. In Figure 44, we see that the best case of boosted HMM again outperforms the average case for HMM with random restarts.

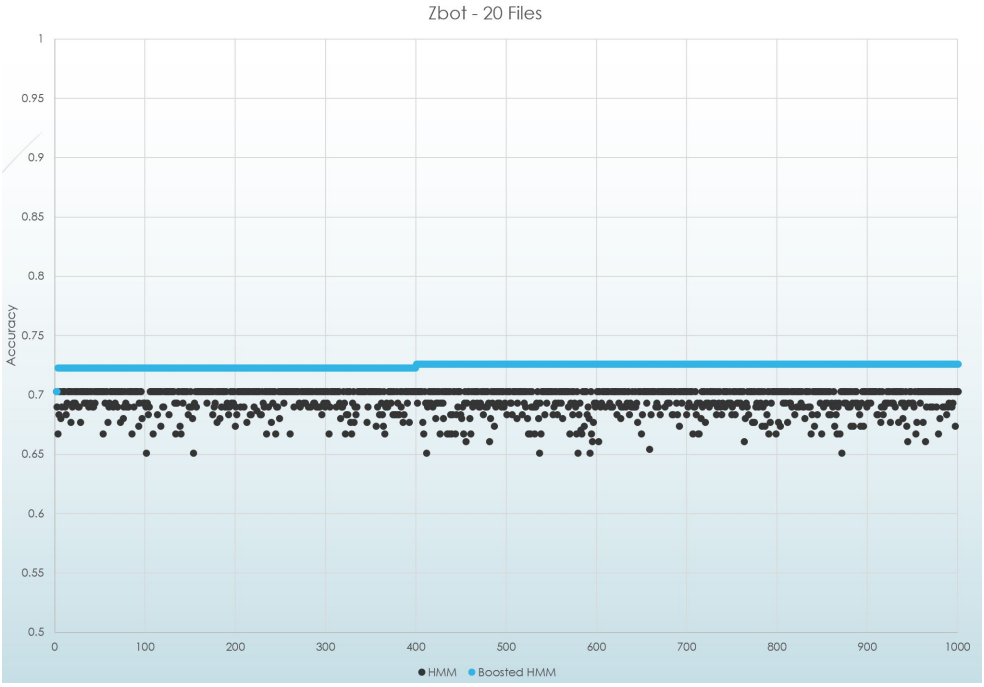

Figure 41: Zbot Cold Start - 20 Files

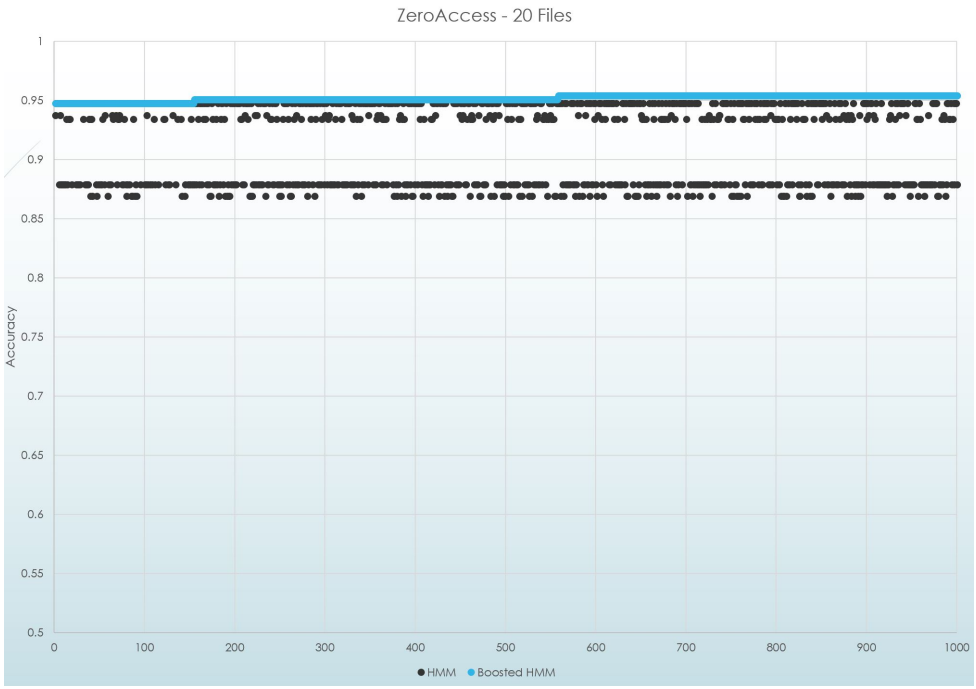

Figure 42: ZeroAccess Cold Start - 20 Files

Figure 43: Cold Start - 20 Files Summary

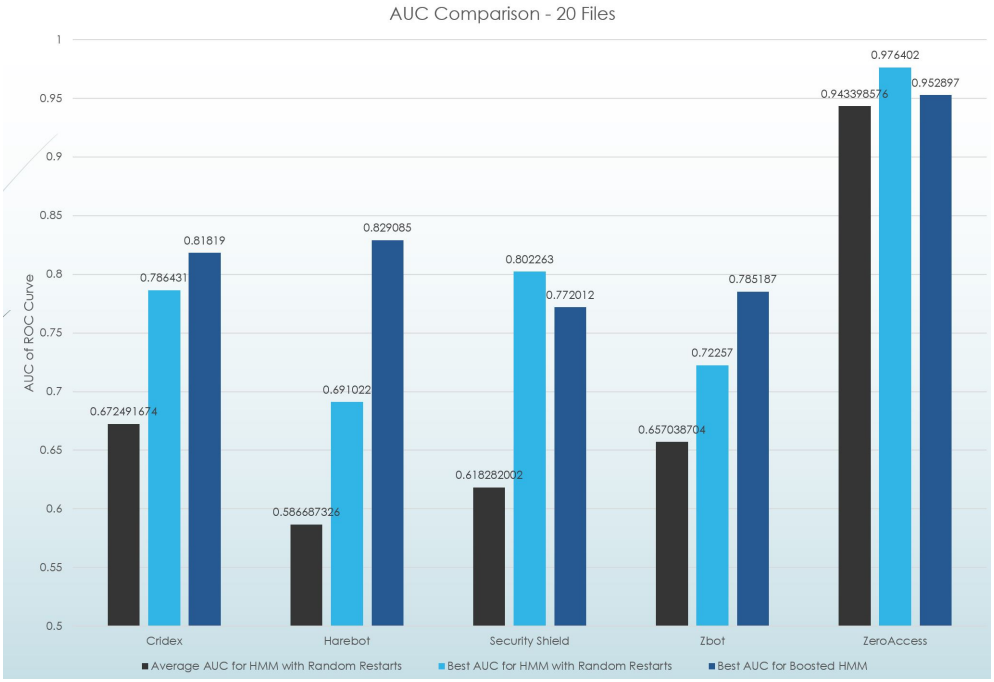

Figure 44: Cold Start - 20 Files AUC for ROC comparison

Finally, we use 25 files for training the models for each family and again an improvement in accuracy is seen for each malware family. In Figure 45, we see an improvement in accuracy for of 2.8% and an improvement of 0.84% for Harebot as seen in Figure 46. A similar improvement of 0.82% is seen in accuracy for Security Shield as shown in Figure 47.

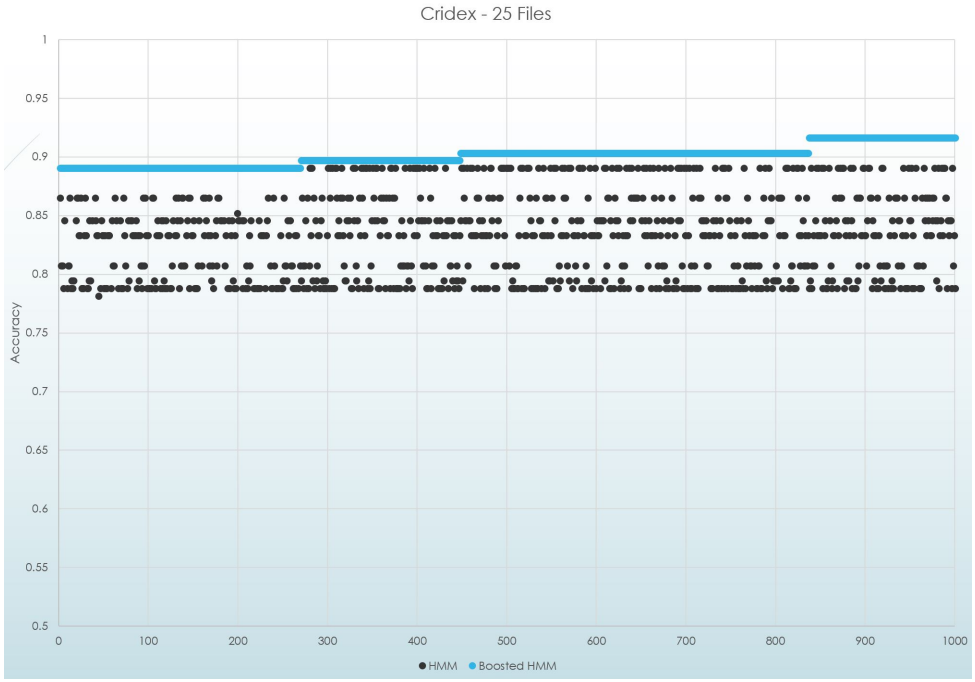

Figure 45: Cridex Cold Start - 25 Files

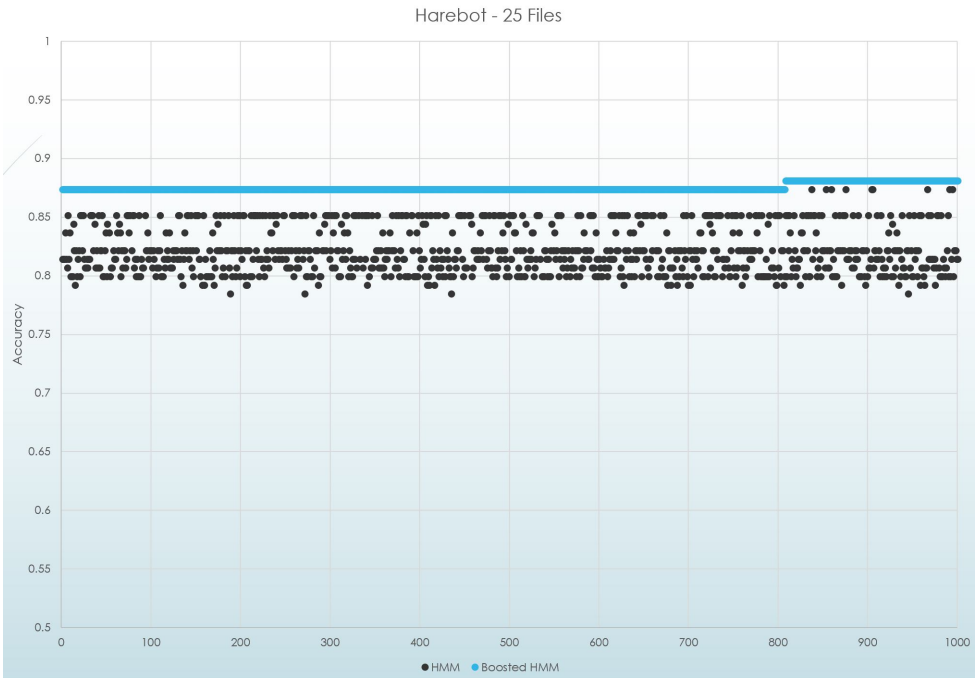

Figure 46: Harebot Cold Start - 25 Files

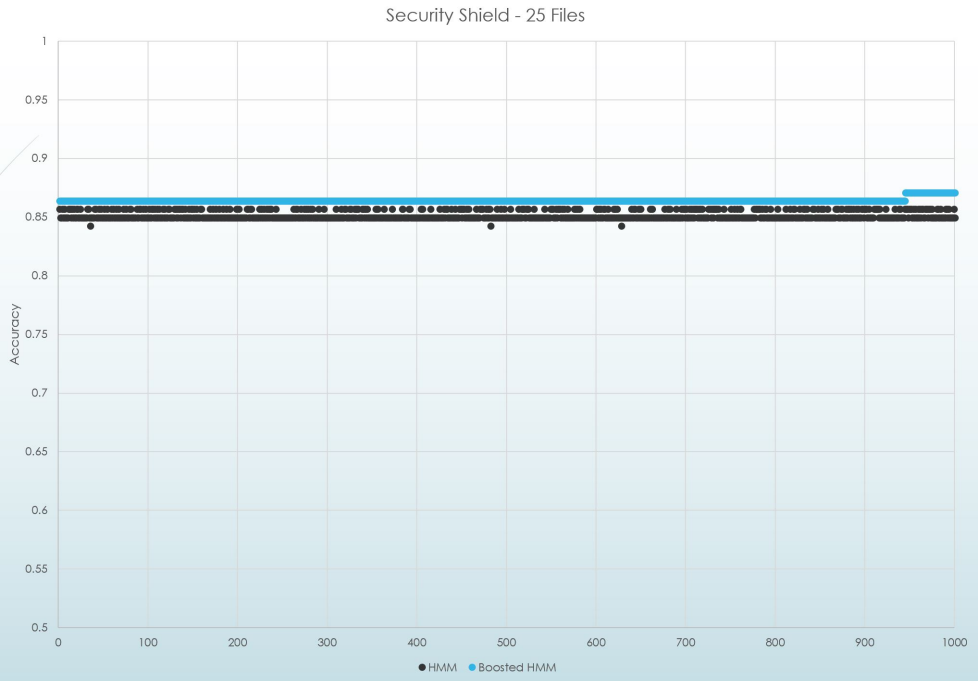

Figure 47: Security Shield Cold Start - 25 Files

For Zbot and ZeroAccess, only 200 files are considered for scoring to avoid an imbalanced dataset. A significant improvement of 4.86% in accuracy is observed for Zbot as shown in Figure 48 and the accuracy for ZeroAccess family improves by 0.67% as seen in Figure 49.

The Figure 50 provides a comparison of the performance of both techniques for each of the malware family using only 20 files for training. On comparing the AUC of the ROC curve for both the techniques, we see that for all malware families except ZeroAccess, there is a significant improvement in the AUC using boosted HMM which is similar to what was observed in previous experiments. In Figure 51, we see that the best case of boosted HMM consistently outperforms the average case and the best case for HMM with random restarts in all but the best case for ZeroAccess family.

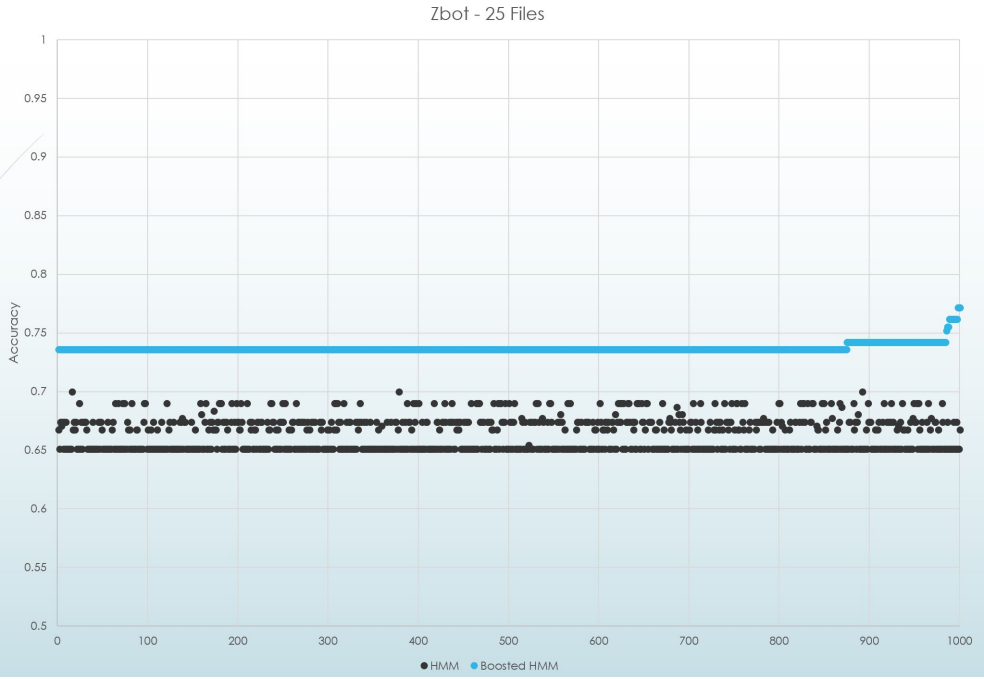

Figure 48: Zbot Cold Start - 25 Files

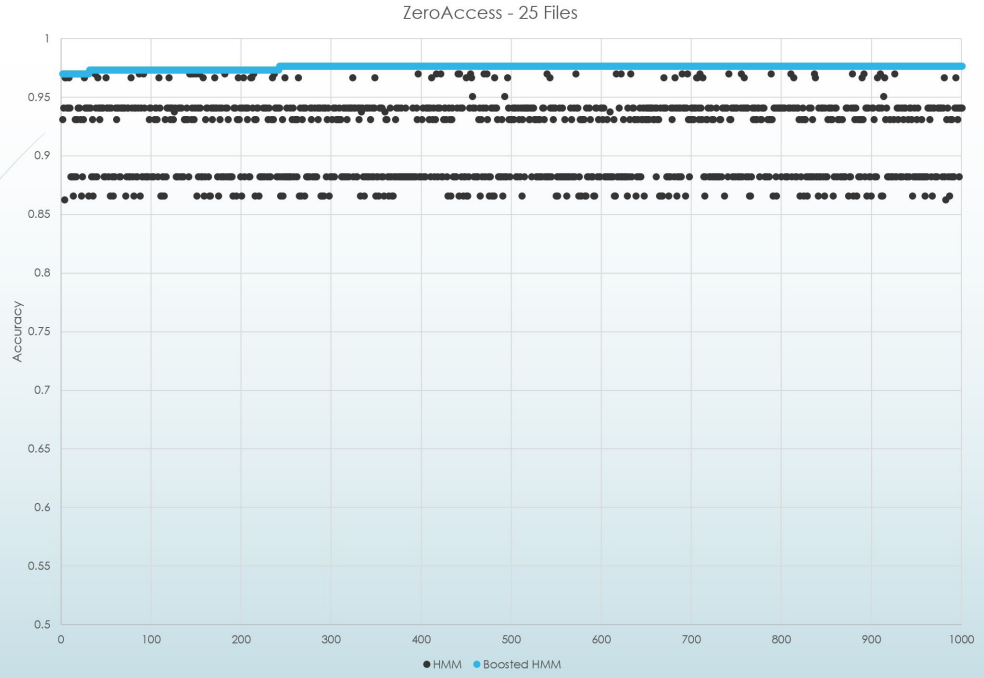

Figure 49: ZeroAccess Cold Start - 25 Files

Figure 50: Cold Start - 25 Files Summary

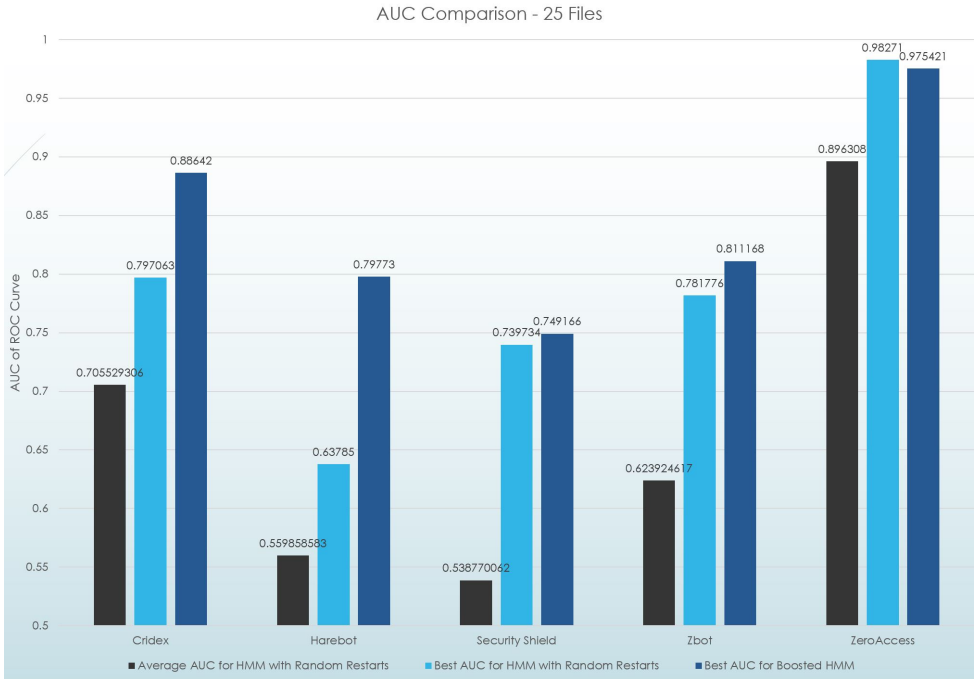

Figure 51: Cold Start - 25 Files AUC for ROC comparison

4.3.2 Summary

We can see that, from Figure 52, there is an improvement in accuracy using boosted HMM over the traditional random restarts technique for almost all cases except for a couple of experiments. Also, no trend is observed in the improvement depending on the number of files chosen for training. Depending on the malware family, we may choose the appropriate number of files used for training the models to maximize the improvement in accuracy.

Figure 52: Cold Start - Accuracy Improvement Percentage

CHAPTER 5

Conclusion and Future Work

5.1 Conclusion

With an increasing threat of malware attacks and proliferation of Internet, it is important to have optimal and efficient malware detection techniques. Combining classifiers for malware detection using hidden Markov models and boosting techniques like AdaBoost provide better results than the traditional random restart technique and help improve malware classification. Morphing is one of the common techniques used for defeating classifiers based on signature detection. We see that, using the traditional techniques, even the slightest amount of morphing by just concatenating benign code at the end of training data, decreases the accuracy of the classifier.

Sophisticated techniques of adding dead code make it easier to defeat such traditional signature detection methodologies. The classifiers obtained using traditional techniques would be weaker. We can then apply boosting techniques on these weak classifiers to produce better results. A similar approach is found to have improved the performance of classifiers for intrusion detection systems [15]. Boosting can be applied to malware detection and various similar pattern recognition and signature detection problems, to improve the classification. Boosting proves to be very useful for combating techniques used by malware developers to avoid detection. We should always employ boosting algorithm on the classifications obtained using multiple models and combine them to obtain a better improved classification with minimal extra effort.

5.2 Future Work

New samples of malware collected, are seen to employ highly sophisticated techniques to avoid detection. Some malware are encrypted while some are metamorphic which constantly evolve to avoid such signature detection. We can use boosted techniques combined with hidden Markov models to identify and detect encrypted text. Using this, we would be able to identify encrypted malware. Metamorphic malware on the other hand, constantly evolve. However, they are still not very different from the original form of malware. Hence, boosting will help identify those odd samples of malware which might have only slightly avoided detection. We have limited information and sample data on most of the newer malware. We can use multiple models with limited training which act as weak classifiers and combine those using boosting to have a stronger classifier. This will help us quickly improve our detection ability for malware which are uncommon. We have a variety of such scenarios in which boosting would help us improve our ability to detect malware.

LIST OF REFERENCES

[1] ‘‘Malicia Project,’’ http://malicia-project.com/, 2015.

[2] ‘‘Panda Security: Cyber crime insights,’’ https://www.pandasecurity.com/mediacenter/malware/cyber-crime-insights/, November 2012.

[3] M. Vihinen, ‘‘Measures and their interpretation in variation effect analysis,’’ https://www.researchgate.net/figure/Contingency-matrix-and-measures-calculated-based-on-it-2x2-contigency-table-for_fig4_230614354, June 2012.

[4] ‘‘International telecommunications union (ICT) facts and figures 2017,’’ https://www.itu.int/en/ITU-D/Statistics/Documents/facts/ICTFactsFigures2017.pdf, July 2017.

[5] ‘‘Statista: Internet users stats --- number of internet users,’’ https://www.statista.com/statistics/273018/number-of-internet-users-worldwide, 2018.

[6] ‘‘Statista: Internet of Things --- Number of connected devices worldwide,’’ https://www.statista.com/statistics/471264/iot-number-of-connected-devices-worldwide/, 2018.

[7] ‘‘Rand Corporation: Cyber Warfare,’’ https://www.rand.org/topics/cyber-warfare.html.

[8] ‘‘TechTerms: Malware definition,’’ https://techterms.com/definition/malware.

[9] ‘‘Accenture: Cost of cyber crime study,’’ https://www.accenture.com/t20170926T072837Z__w__/us-en/_acnmedia/PDF-61/Accenture-2017-CostCyberCrimeStudy.pdf, 2017.

[10] S. Morgan, ‘‘Top 5 cybersecurity facts, figures and statistics for 2018,’’ https://www.csoonline.com/article/3153707/security/top-5-cybersecurity-facts-figures-and-statistics.html, January 2018.

[11] N. DuPaul, ‘‘Common malware types: Cybersecurity 101,’’ https://www.veracode.com/blog/2012/10/common-malware-types-cybersecurity-101, October 2012.

[12] J. Aycock, *Computer Viruses and Malware*, ser. Advances in Information Security. Springer US, 2006. [Online]. Available: https://books.google.com/books?id=xnW-qvk1gzkC

[13] T. H. Austin, E. Filiol, S. Josse, and M. Stamp, ‘‘Exploring hidden markov models for virus analysis: A semantic approach,’’ in *Proceedings of the 2013 46th Hawaii International Conference on System Sciences*, ser. HICSS ’13. Washington, DC, USA: IEEE Computer Society, 2013, pp. 5039--5048.

[14] M. Stamp, *Introduction to Machine Learning with Applications in Information Security*. Chapman and Hall/CRC, 2017.

[15] Y.-S. Chen and Y.-M. Chen, ‘‘Combining incremental hidden markov model and adaboost algorithm for anomaly intrusion detection,’’ in *Proceedings of the ACM SIGKDD Workshop on CyberSecurity and Intelligence Informatics*, ser. CSI-KDD ’09. New York, NY, USA: ACM, 2009, pp. 3--9.

[16] N. J. Nilsson, ‘‘Introduction to machine learning,’’ http://robotics.stanford.edu/people/nilsson/MLBOOK.pdf, November 1998.

[17] H. Bourlard, Y. Kamp, and C. Wellekens, ‘‘Speaker dependent connected speech recognition via phonetic markov models,’’ in *ICASSP ’85. IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 10, Apr 1985, pp. 1213--1216.

[18] A. Kundu, Y. He, and P. Bahl, ‘‘Recognition of handwritten word: first and second order hidden markov model based approach,’’ in *Computer Vision and Pattern Recognition, 1988. Proceedings CVPR ’88., Computer Society Conference on*, Jun 1988, pp. 457--462.

[19] G. D. Forney, ‘‘The viterbi algorithm,’’ *Proceedings of the IEEE*, vol. 61, no. 3, pp. 268--278, March 1973.

[20] S. S. Jarng, ‘‘HMM voice recognition algorithm coding,’’ in *2011 International Conference on Information Science and Applications*, April 2011, pp. 1--7.

[21] D. Carlin, A. Cowan, P. O’Kane, and S. Sezer, ‘‘The effects of traditional anti-virus labels on malware detection using dynamic runtime opcodes,’’ *IEEE Access*, vol. 5, pp. 17 742--17 752, 2017.

[22] M. Stamp, ‘‘Boost your knowledge of adaboost,’’ https://www.cs.sjsu.edu/~stamp/ML/files/ada.pdf, December 2017.

[23] W. Hu, J. Gao, Y. Wang, O. Wu, and S. Maybank, ‘‘Online AdaBoost-based parameterized methods for dynamic distributed network intrusion detection,’’ *IEEE Transactions on Cybernetics*, vol. 44, no. 1, pp. 66--82, Jan 2014.

[24] H. Grabner and H. Bischof, ‘‘On-line boosting and vision,’’ in *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR’06)*, vol. 1, June 2006, pp. 260--267.

[25] T. G. Tape, ‘‘The area under an ROC curve,’’ http://gim.unmc.edu/dxtests/ROC3.htm.

[26] E. Ban, ‘‘Insights into antivirus technology,’’ http://oemhub.bitdefender.com/importance-of-false-positives-for-antimalware-engine-quality, June 2015.

[27] J. Kittler, M. Hatef, R. P. W. Duin, and J. Matas, ‘‘On combining classifiers,’’ *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no. 3, pp. 226--239, Mar 1998.