National College of Ireland

Malware Classification using Km-SVM

MSc Internship
MSc in CyberSecurity

Ashish Ghorpade
Student ID: x18147461

School of Computing
National College of Ireland

Supervisor: Dr. Muhammad Iqbal

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Ashish Ghorpade
Student ID:	X18147461
Programme:	MSc in CyberSecurity **Year:** 2019-2020
Module:	Internship
Supervisor:	Dr. Muhammad Iqbal
Submission Due Date:	8th January 2020
Project Title:	Malware Classification using Km-SVm
Word Count:	4632 **Page Count:** 20

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.
I agree to an electronic copy of my thesis being made publicly available on NORMA the National College of Ireland's Institutional Repository for consultation.

Signature: 

Date: 7th January 2020

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Malware Classification using Km-SVM

Ashish Ghorpade
X18147461

Abstract

Malware identification and classification is a problem faced even in this decade. This is majorly due to the fact that advance malware are more sophisticated in nature and have state of the art abilities to remain hidden or change their code/behaviour more like a smart malware. Hence, old detection and classification techniques are no longer as effective. This resulted in pivoting towards machine learning for better detection and classification of such malware. This is the motive behind the topic of this research thesis. Through this thesis, effort has been made to better classify malware using a combination of supervised and unsupervised machine learning while keeping the accuracy at acceptable levels and reducing training time. This led to conducting study not only in malware classification but also in other fields such as speech recognition and medical research to identify different techniques which could possibly be used for successful malware classification.

1 Introduction

This research aims to find a better way to classify malware using a combination of existing machine learning techniques. Efforts were made to not only classify malware with accepTable 1ccuracy but also to reduce training time which could prove to be useful when dealing with huge malware datasets. This encouraged the creation of a tool that can be used for malware classification and outputs results from 3 different machine learning models for the purpose of evaluation. The primary goal of this project is to verify the hypothesis that Km-SVM (Kmean-SVM) can be used as a malware classifier with results as good as a supervised learning model (SVM) and a lower training time.

2 Related Work

A thorough literature review was conducted to better understand the background on research conducted in the area of Malware Identification and Classification based on various techniques. This was divided into 4 significant parts. Evolution of existing malware detection techniques helped to understand the trends and current status in the area of malware detection. A study was also conducted on how malware files are analysed and converted to a format usable for analysing the data. Further literature review was conducted to study various machine learning and prediction models used for analysis of malware and other areas such as medical science (cancer prediction) and voice recognition. This study was further extended to understanding how feature extraction and selection worked and its impact and significance on analysis and prediction. This review was crucial in hypothesising the basis of the research topic. Hence the hypothesis, Malware classification can be improved in accuracy over basic Kmeans(unsupervised) and reduce training time as compared to SVM(supervised) while

providing similar accuracy to supervised learning. This will be attempted by combining Kmeans and SVM using HMM as a complementing feature selection technique.

2.1 Evolution of Malware Detection Techniques

Accurate malware detection has been a major concern in the Cyber Security community. This has resulted in the evolution of detection techniques. The techniques described in [1] have evolved significantly from anomaly, specification and signature-based detection to sophisticated behaviour-based detection using various machine learning techniques such as SVM [2] and Deep learning [3]. This change in detection techniques was ineviTable 4ue to the development of sophisticated malware capable of polymorphism [4]. Modern malware has the capability to remain hidden, detect the host and Anti-Virus configuration, change behaviour and connect to a remote server for instructions [5],[6]. Although effective and accurate, machine learning techniques have short comings such as scarce availability of labelled data for supervised learning such as SVM [2]. This problem can be addressed though unsupervised machine learning such as K-means [7] but usually lack in accuracy as compared to supervised machine learning [8].

2.2 Dynamic Analysis of Code and MIST

Machine learning techniques require data in numerical format. However, malware files/executables contain code which is executed on a machine. This code is written in High level languages which is then converted into machine level and then as binary instructions [9]. The paper [10] proposes the use of online sandbox for execution of such code in a simulated environment and capturing the behaviour. This consists of sequences of API calls made by the malware file. Similar methodologies have been mentioned in [11] and [12]. The Dynamic code analysis technique mentioned in [10] was deemed to be effective based on the output and ease of use. CWSandbox was recommended for this analysis. A sample XML based output can be seen in Figure 1.

Figure 1

The literature in [13] and [14] further facilitated the use of CWSandbox and conversion Of XML based output to a Hexadecimal representation. This representation has been referred to as the Malware Instruction Set (MIST) in [14]. Figures B and C depict the XML representation and the MIST representation respectively. Figure 4 represents a generic view and interpretation of this format.

```
<load_dll filename="C:\WINDOWS\system32\kernel32.dll" successful="1"
 address="#7C800000" end_address="#7C908000" size="1081344"
 filename_hash="c88d57cc99f75cd928b47b6e444231f26670138f"/>
```

Figure 2

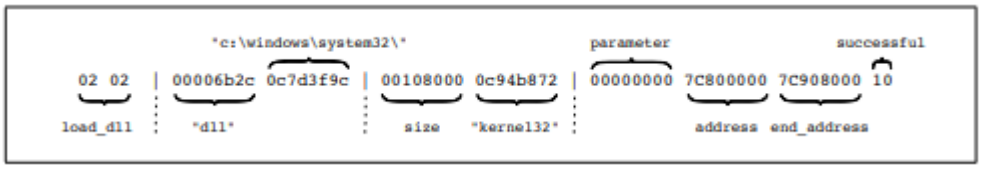

Figure 3

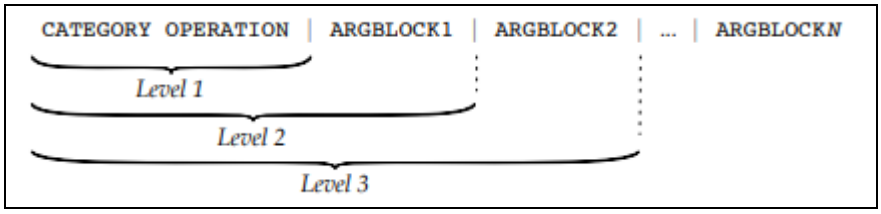

Figure 4

2.3 Km-SVM Algorithm

After referring to multiple papers on machine learning techniques for malware detection and classification and finding the optimum way to represent the API calls, the major focus was on hypothesizing an Algorithm that can utilize supervised and unsupervised learning techniques for malware analysis. This was encouraged due to the literature in [15] and [16]. Paper [16] suggested the use of semi-supervised learning for malware analysis which can be a potential solution for the problem of limited labelled data. It uses the LLGC (Learning with Local and Global Consistency) algorithm for classification. However, the accuracy obtained with 50% labelled data was about 80% which is lower that supervised learning accuracy. Paper [17] suggested use of the wKm-SVM (weighted Kmeans SVM) algorithm for cancer prediction. This facilitated further research in the use of Kmeans along with SVM. Papers [18], [19] and [20] have mentioned the use of Km-SVM algorithm. This was proposed to over come the problem of data being separated into subclasses. This is relatable to our problem of malware classification wherein, the malware can be further classified into sub-types based on their function, polymorphism and behaviour. These papers suggest that once the data has been clustered for each sub-class, this can be classified using SVM. This technique has an added advantage of low training time as compared to the former.

2.4 Feature Selection Techniques and HMM

Another crucial factor to consider while performing machine learning is the extraction and selection of features. These features form the basis of segregating/classifying the data. This is important for malware analysis as malware often have the tendency to

disguise as a benign file to trick Anti-Virus engines. Often, malware is not detected if they are dormant. While the API sequence calls discussed in [13] and [14] are useful, the features can be hidden. Papers [21], [22] and [23] discuss ways for extracting and selecting these features. Literature in [21] and [23] promotes the use of PorTable 5xecutable (PE) headers. A porTable 5xecuTable 6ile contains various headers in the form of a Key – Value pair. However, it was found that it is useful only for the identification problem and not for classification. This observation can be better understood from the further description of these papers. The papers suggest using few of these pairs as features to be analysed for identification of malware. Below are few of the headers suggested as features:

- **DebugSize –** The value shows the size of the debug directory. Debug directories tend to have a non-zero value in case of benign files and zero in case of malicious. Note that this can only be used for a possible identification of malware but not for classification of malware.

- **ExportSize –** This may also be a non-zero value for benign files as this directory may contain multiple DLLs. Malicious files do not tend to have many DLLs as their intent is limited.

- **NumberOfSections –** This feature may vary in both malicious and benign files.

Based on the above factors it can be confirmed that this method of feature extraction and selection **CANNOT** be used for malware classification but provides more insights into generic malware files and the importance of feature extraction and selection.

Further study led to [13] which provided a comparison of various feature extraction and selection techniques. The paper studies techniques such as unigram binary, unigram frequency, bigram binary, bigram frequency and **HMM (Hidden Markov Models).** The study concluded that HMM was the most optimum way to extract features. This observation was confirmed though studies mentioned in [24], [25] and [26].

HMM: Hidden Markov Model is a further derivation of the Markov Model. It is classified as a statistical model that can be trained on multiple observation sequences. This model can identify hidden states in the sequences and calculate the likelihood of a sequence occurring [27]. This model is used for solving the following three basic problems:

- **Problem 1 –** Calculate probability of an observation sequence based on a trained/fitted model.

- **Problem 2 –** Finding the best sequence/occurrence of hidden states for a trained model.

- **Problem 3 –** Model training to maximise the probability of the observation sequences.

These problems were found to be similar to the API call sequences being used for malware classification [28].

3 Research Methodology

The research methodology is based on the conclusions of the literature. The proposed approach is to utilise the MIST represented data available through the CWSandbox and encoding it for use in feature extraction and scoring. This is further used for performing clustering and finally SVM. These trained models are used for predicting the test data. The sub-sections below provide complete details on the methodology followed.

3.1 Data selection

Given the nature and possible damage, malware data and files are hard to obtain. As such, the malware dataset used in [13] and [14] is used. It is available at [29]. The dataset comprised of 29,661 malicious files belonging to various malware families. However, the number of files per malware family varied significantly. Hence, files belonging to only 9 families was considered. Table 1 shows the number of files per malware type:

MALWARE TYPE	FILE COUNT
Agent	748
Allaple	869
Patched	816
Autoit	1191
Nothingfound	6003
Swizzor	1037
Texel	5993
Virut	2092

Table 1

3.2 Data Pre-processing

Pre-processing of data was carried out to ensure that the data did not contain any discrepancies. This was achieved by balancing the files used per malware type and then encoding each file. The Table 2 shows the number of files per type that were used.

MALWARE TYPE	SET 1
Agent	200-300
Allaple	200-300
Patched	200-300
Autoit	200-300
Nothingfound	200-300
Swizzor	200-300
Texel	200-300
Virut	200-300

Table 2

The number of files used was decided based on the hardware limitations of the device used for training and testing and the literature mentioned in [24].

After selection of dataset, an encoding was performed on the MIST represented sequences to make the data compatible for Machine Learning. This was performed by referring to [13] and [14]. Table 3 shows the original and encoded values, below.

	Category	# syscalls		Category	# syscalls
01	Windows COM	4	*0B*	Windows Services	11
02	DLL Handling	3	*0C*	System	2
03	Filesystem	14	*0D*	Systeminfo	7
04	ICMP	1	*0E*	Thread	3
05	Inifile	5	*0F*	User	8
06	Internet Helper	5	*10*	Virtual Memory	5
07	Mutex	2	*11*	Window	5
08	Network	6	*12*	Winsock	13
09	Registry	9	*13*	Protected Storage	9
0A	Process	7	*14*	Windows Hooks	1

Table 3

Based on the reference Table 3, each unique call was encoded by a value ranging from 1-120. This was determined based on the 120 unique calls found in the MIST representation. This resulted in sequences of variable length with values ranging from 1 to 120. Each sequence represents the API call sequence for a single malicious file.

After the completion of encoding, the dataset was split into Train and Test sets. 10% of the dataset files were used for testing. This ratio was chosen based on the hardware limitations which in-turn limited the testing files used.

3.3 Feature Extraction using HMM

The Hidden Markov Model (HMM) was chosen for the purpose of feature extraction due to its compatibility with sequences of variable length. Problem 3 of the HMM was the basis of training 9 unique HMMs for the 9 unique malware types used [28]. The trained HMMs were used to calculate the log likelihood scores. These trained models were then used to score each file in the test dataset. This was done using the solution for Problem 1 of HMMs [28].

3.4 Training and Testing

K-Mean: The K-Mean clustering algorithm [7] was used to create clusters based on the HMM scores for the files. The number of clusters was based on the number of unique malware types. This part was conducted in batches of 4 or 5 malware types to overcome the hardware limitations faced during the execution of the complete code. Only the scores were used for the purpose of Kmeans to ensure that the data is unlabelled.

SVM: The SVM algorithm was used to classify the malware file by appending the scores generated from the malware files to the respective labels. This resulted in the generation of labelled data for performing SVM. However, this was done without using the cluster output from K-mean.

Km-SVM: Finally, a Km-SVM model was trained and used for testing. This was achieved by using the clustered output from K-Mean and performing SVM. This resulted in the creation of labels from the clusters and feeding the output to SVM for further classification.

4 Design Specification

This section depicts the complete design of the model created. This acts as a complete tool for malware classification. This tool is capable of mining data from separate malware MIST files, encoding, saving cache, splitting the dataset into train and test parts, performing HMM, Kmean, SVM and Km-SVM on the train and test data. Figure 5 shows the complete process flow and the design of the tool. As seen from the Figure 5, malicious exe and dll files are placed in the CWSandbox and a MIST represented output is obtained. This output is encoded for usability. The encoded data is then stored in a cache file for possible future use. This cache file is then split into train and test data. The train data is used to fit the HMM model on set of observation sequences. The number of models trained is equal to the number of unique malicious file types. The train files are scored and these scores are used for Kmeans clustering. The same scores are labelled and used for fitting an SVM model. Separately, the clustered output from Kmeans is used for SVM to generate the output of a Km-SVM model. The output contains a Graph, Model Accuracy and Training time. This process is again repeated for the test data but on the trained models obtained from the previous iteration. This tool was designed with the intent of comparing the output results from three separate models – Kmeans, SVM and Km-SVM.

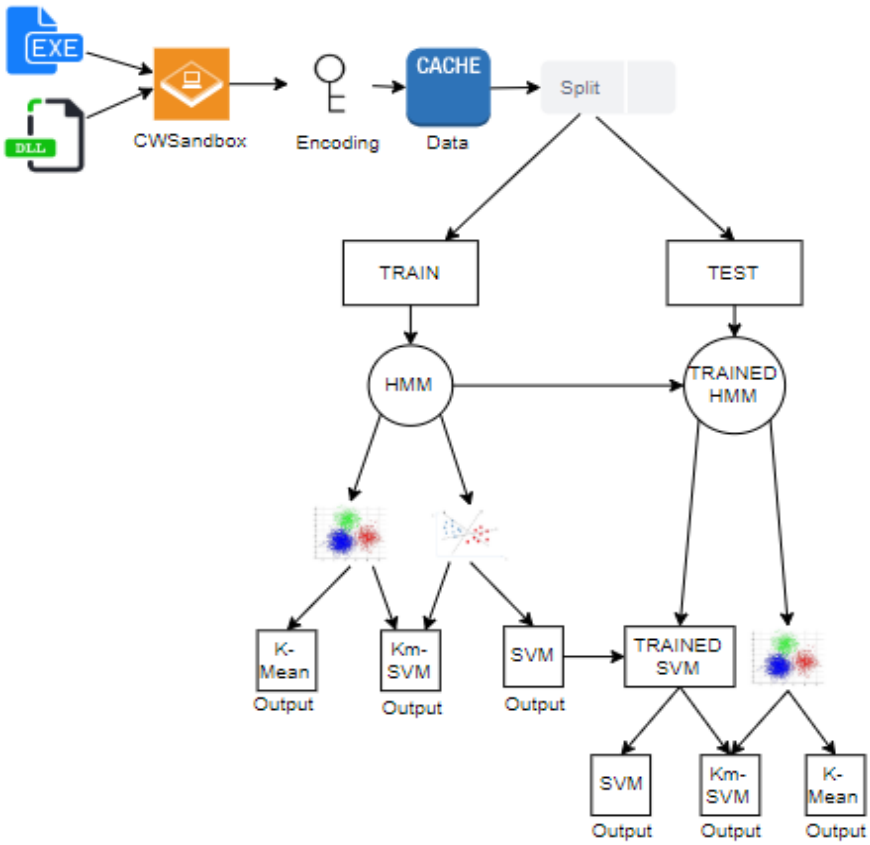

Figure 5

5 Implementation

This part explains the complete implementation of the developed tool. Information on the technologies used and creation of an environment is explained further.

Hardware Specifications: For the purpose of this project, a 4th Gen i7 Processor Laptop with 8 GB Ram was used. Due to these limitations, the training and testing was limited to 200-300 files per type of malware. Additionally, to further reduce the load on the hardware, analysis was performed in batches of 4-5 malware file types.

Software Specifications: The host operation system is Windows 8.1 Professional 64-Bit. Below is the list of software used for the purpose of this project.

- **Python 3.7.4 –** Python is a robust language which is easy to learn and use. This language is supported by a number of IDEs. Moreover, it has a vast community support and packages available for machine learning and other data analytic models. This encouraged the use of Python for the creation of the proposed tool.

- **Visual Studio Code –** This is an IDE that requires minimum resources to function. Additionally, it supports the use of Python which was the basis of selection of this software.

- **Python Packages –** Multiple python packages were used for performing operations such as data mining, cleaning, encoding and machine learning. A list of these packages and their respective versions is made available in the Configuration Manual.

Code Tuning and Setup: It is recommended that Visual Studio Code be used for setting up the tool. This tool has 3 major components:

- Dataset.py – This file contains the code to mine, encode and save data in cache file.

- Hmmmodels.py – This script contains the complete code for training and testing the data through HMM, Kmeans, SVM and Km-SVM.

- Dataset – This folder contains the raw files belonging to different malware families. The number of files can be varied based on the System Configuration of the Machine used to run the tool.

Output: The output consists of Graphs, Accuracy out of 1 and Training/Testing time for Kmeans, SVM and Km-SVM on the Train and Test data. This is useful for evaluation purpose of the models.

6 Evaluation

For the purpose of this report, only the output values for the test set are considered. Three separate case studies are performed with varying set of malware types and number of files per malware type limited to 200-300.

6.1 Batch 1 – AutoIT, Allaple, NothingFound, Swizzor

Classification results of the test on Batch one can be seen in Figures F,G and H. Model Accuracy can be seen in Figure 9 and Training time for SVM and Km-SVM can be seen in Figure 10.

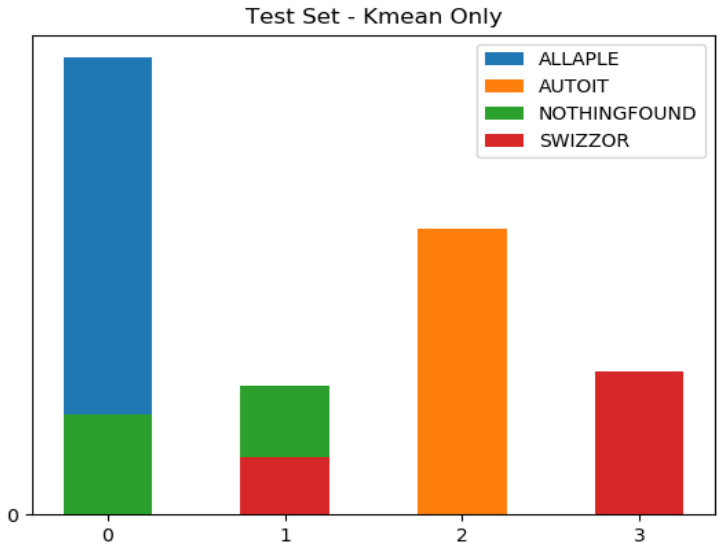

Figure 6

It can be seen that AutoIt dominates cluster 2 while Allaple and Swizzor dominate cluster 0 and 3 respectively. Cluster 1 is shared between NothingFound and Swizzor. Traces of NothingFound are found in Cluster 0 as well.

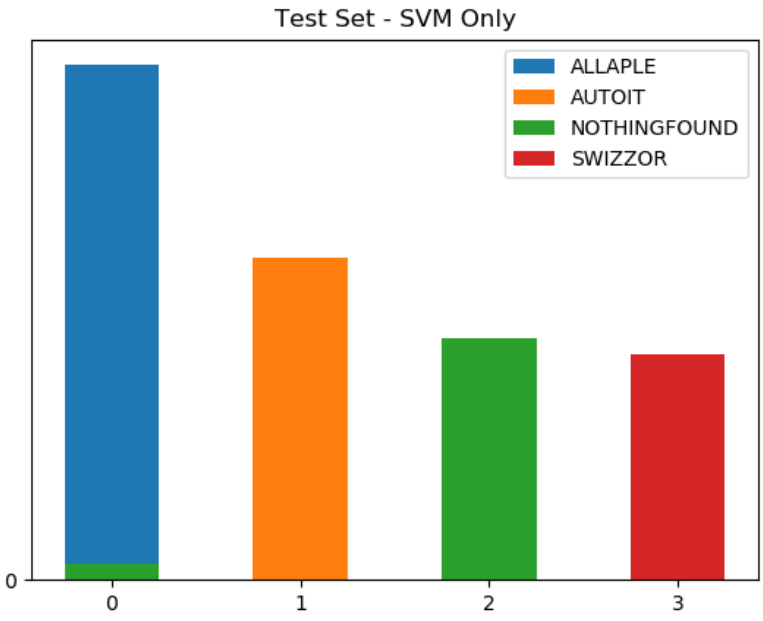

Figure 7

Figure 7 depicts 4 distinct graphs for the 4 malware families based on SVM model. NothingFound was the only one that shared Graph 0 and Graph 2.

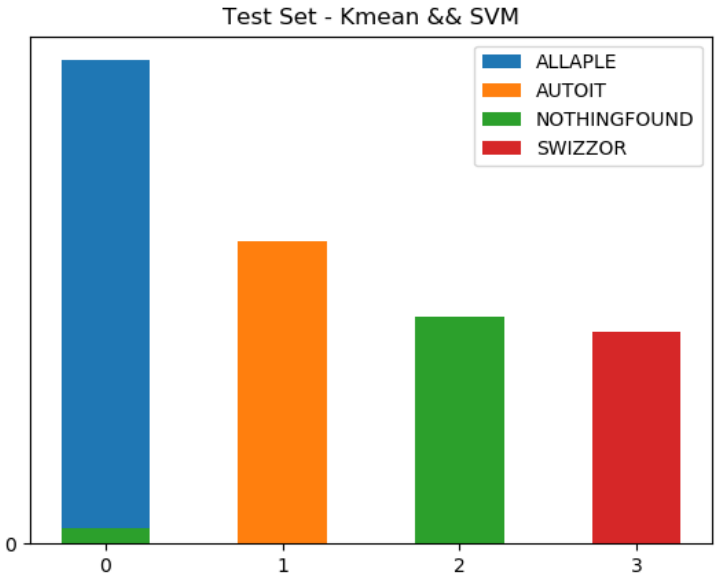

Figure 8

Figure 8 shows results similar to Figure 7. This implies that the output for SVM and Km-SVM was similar. This can be confirmed from the test accuracy Figure 13entioned in Figure 9.

```
C:\Users\Ashish\Desktop\HMM.v.5>C:/Users/Ashish/Ap
.5/hmmmodels.py
 Load dataset from cache file. OK
     Loading Model: models/Model.pkl
Train Set: 810 samples
Test Set: 90 samples
Accuracy for Test:
     Kmean - SVM: 0.9
     SVM only: 0.9
     Kmean only: 0.4666666666666667
```

Figure 9

The accuracy for Km-SVM was similar to SVM. However, accuracy for Kmean was only half of the other two models.

```
Training Time for SVM Only (720 samples): 0.06663036346435547
          Cluster with kmean: 4 clusters
          Find  Neighbor set
               Create Distance matrix: 720 x 720
               Iterate for each sample in misC:
          NT set (Train set for Kmean+SVM): 718 samples
          Fit SVM with Gaussian kernel with NT set
Training Time for Kmean+SVM (718 samples): 0.00702357292175293
```

Figure 10

The training time for Km-SVM was found to be significantly lower than SVM. This can be seen from the first and last lines in Figure 10.

6.2 Batch 2 – Allaple, AutoIt, Patched, Virut

Classification results of the test on Batch one can be seen in Figures K, L and M. Model Accuracy can be seen in Figure 14 and Training time for SVM and Km-SVM can be seen in Figure 15.

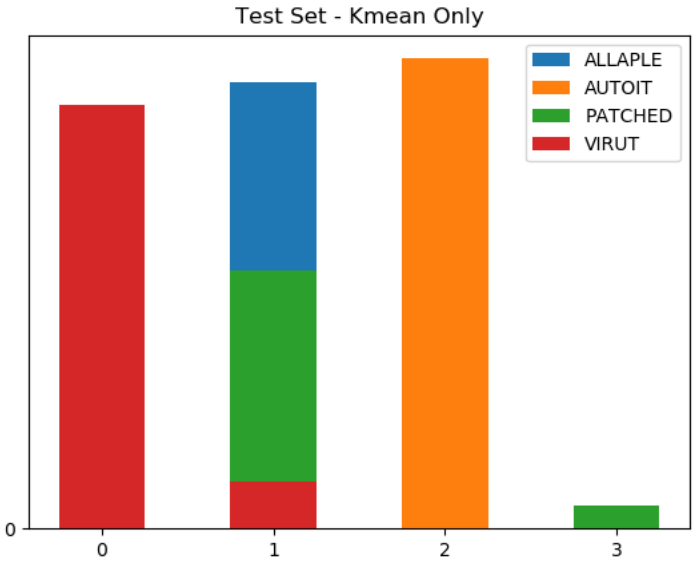

Figure 11

Cluster 0 and 2 are occupied by Virut and AutoIt respectively. Cluster 3 has a very small number of Patched files whereas, cluster 1 is shared majorly by Pathced and Allaple and a small number of Virut files.

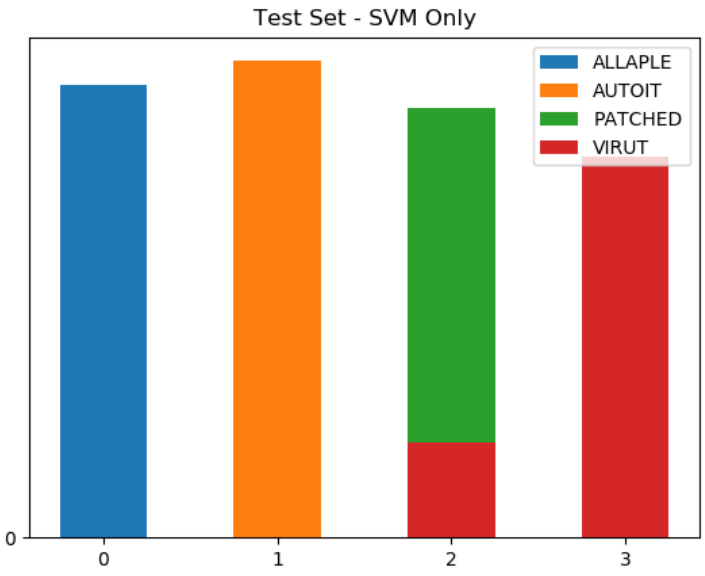

Figure 12

SVM classification is observed to be significantly better as compared to Kmeans. Only the graph 2 is shared by majority of Patched files and a small amount of Virut files.

Figure 13

Figure 13 was once again found to be similar to Figure 12 which implies that the results for Km-SVM were similar to SVM.

```
C:\Users\Ashish\Desktop\HMM.v.5>C:/Users/Ashish/A
.5/hmmmodels.py
 Load dataset from cache file. OK
     Loading Model: models/Model.pkl
Train Set: 720 samples
Test Set: 80 samples
Accuracy for Test:
     Kmean - SVM: 0.9125
     SVM only: 0.9125
     Kmean only: 0.0125
```

Figure 14

The accuracy for Km-SVM and SVM was 0.91 out of 1. However, accuracy for Kmean was only very low in comparison to the other two models.

```
Training Time for SVM Only (720 samples): 0.005003690719604492
          Cluster with kmean: 4 clusters
          Find  Neighbor set
               Create Distance matrix: 720 x 720
               Iterate for each sample in misC:
          NT set (Train set for Kmean+SVM): 720 samples
          Fit SVM with Gaussian kernel with NT set
Training Time for Kmean+SVM (720 samples): 0.006022214889526367
     Save Model: models/Model.pkl
```

Figure 15

The training time for Km-SVM was found to be almost similar to SVM. Not much significant difference could be found since the number of training files were low.

6.3 Batch 3 – Agent, Allaple, AutoIt, Patched, Swizzor

Classification results of the test on Batch one can be seen in Figures P,Q and R.
Model Accuracy can be seen in Figure 19 and Training time for SVM and Km-SVM can be seen in Figure 20.

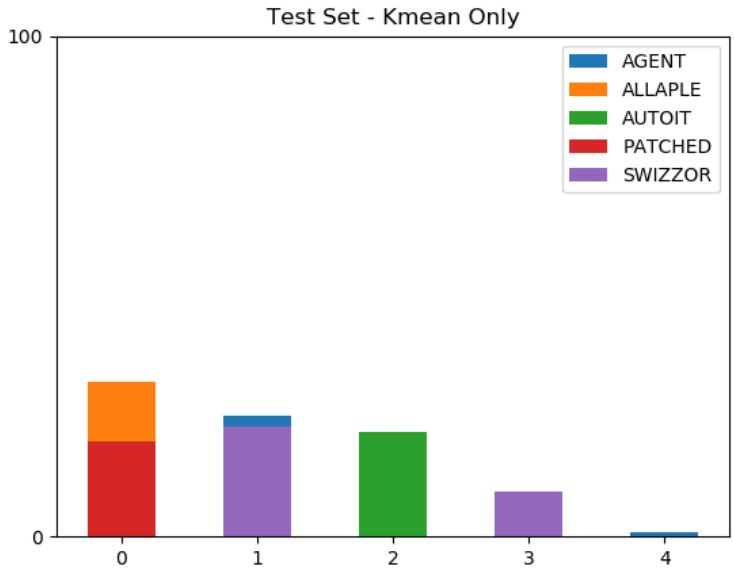

Figure 16

AutoIt and Swizzor were found to the the dominant families in clusters 2 and 1 respectively. The clusters appear to be unbalanced.

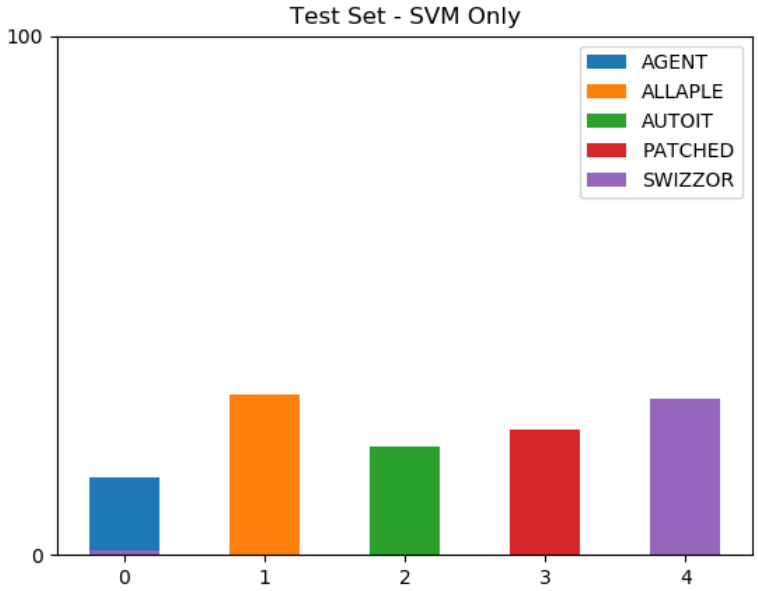

Figure 17

The graph in Figure 17 implies that SVM classification was significantly better than Kmeans. All the malware families are adequately classified.

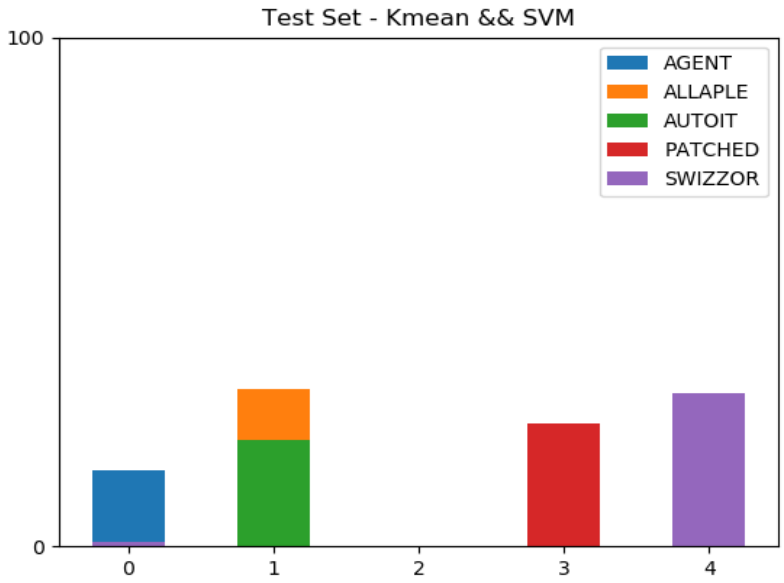

Figure 18

The results for Km-SVM were different than SVM for this batch of malware files.

```
C:\Users\Ashish\Desktop\HMM.v.5>C:/Users/Ashish/Ap
.5/hmmmodels.py
 Load dataset from cache file. OK
     Loading Model: models/Model.pkl
Train Set: 1257 samples
Test Set: 140 samples
Accuracy for Test:
     Kmean - SVM: 0.7142857142857143
     SVM only: 0.8642857142857143
     Kmean only: 0.17142857142857143
```

Figure 19

The accuracy for Km-SVM was 71% which is lower than SVM at 86%. However, accuracy for Kmean was even lower than the other two models.

```
Training Time for SVM Only (1257 samples): 0.02500152587890625
          Cluster with kmean: 5 clusters
          Find  Neighbor set
               Create Distance matrix: 1257 x 1257
               Iterate for each sample in misC:
          NT set (Train set for Kmean+SVM): 1081 samples
          Fit SVM with Gaussian kernel with NT set
Training Time for Kmean+SVM (1081 samples): 0.027019023895263672
     Save Model: models/Model.pkl
```

Figure 20

The training time for both the models was almost similar as observed for the batch 2.

6.4 Discussion

Observations from the 3 different tests conducted on the separate sets of malware families show that the results obtained from Km-SVM were significantly higher than Kmeans for all three tests and equal to SVM in two tests and 16% lower in test 3. The training time for Km-SVM was lower than SVM in test 1 and equal to SVM in test 2 and test 3. These results are subject to the configurations used in the tool and the family types. The malware families of Allaple, AutoIt and Swizzor were found to be significantly different and dominant than the other families over all 3 tests performed.

7 Conclusion and Future Work

The results obtained from the tests show strong affirmation towards the stated hypothesis for use of Km-SVM for malware classification. However, due to time constraints and hardware limitations, a more extensive assessment could not be carried out. However, based on the literature review and the test results for this research, it can be stated that the results were satisfactory. These shortcomings could be addressed in as part of the future work. Additionally, appending the LLGC (Learning with Local and Global Consistency) algorithm will compliment the existing tool as the test results can be used to retrain the model further in-turn improving the overall test accuracy for further iterations. Future work may also include the implementation of KFold Cross Validation or Leave one out Cross Validation for achieving more reliable results.

References

[1] Cerias.purdue.edu. (2007). CERIAS - Center for Education and Research in Information Assurance and Security. [online] Available at: https://www.cerias.purdue.edu/apps/reports_and_papers/view/4328/ [Accessed 11 Sep. 2019].

[2] M. Kruczkowski and E. N. Szynkiewicz, "Support Vector Machine for Malware Analysis and Classification," 2014 IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT), Warsaw, 2014, pp. 415-420. [online] Available at: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6927654&isnumber=6927590

[3] KULSHRESTHA, R. (2019). Malware Detection Using Deep Learning. [online] Medium. Available at: https://towardsdatascience.com/malware-detection-using-deep-learning-6c95dd235432 [Accessed 7 Oct. 2019].

[4] Lord, N. (2018). What is Polymorphic Malware? A Definition and Best Practices for Defending Against Polymorphic Malware. [online] Digital Guardian. Available at: https://digitalguardian.com/blog/what-polymorphic-malware-definition-and-best-practices-defending-against-polymorphic-malware [Accessed 9 Oct. 2019].

[5] Lord, N. (2018). What is Advanced Malware?. [online] Digital Guardian. Available at: https://digitalguardian.com/blog/what-advanced-malware [Accessed 13 Oct. 2019].

[6] Granneman, J. (2013). Antivirus evasion techniques show ease in avoiding antivirus detection. [online] SearchSecurity. Available at: https://searchsecurity.techtarget.com/feature/Antivirus-evasion-techniques-show-ease-in-avoiding-antivirus-detection [Accessed 8 Oct. 2019].

[7] En.wikipedia.org. (2020). K-means clustering. [online] Available at: https://en.wikipedia.org/wiki/K-means_clustering [Accessed 11 Nov. 2019].

[8] Guru99.com. (2020). Supervised vs Unsupervised Learning: Key Differences. [online] Available at: https://www.guru99.com/supervised-vs-unsupervised-learning.html [Accessed 16 Oct. 2019].

[9] En.wikipedia.org. (2020). High-level programming language. [online] Available at: https://en.wikipedia.org/wiki/High-level_programming_language [Accessed 20 Oct. 2019].

[10] Willems, C., Holz, T. and Freiling, F. (2007). Toward Automated Dynamic Malware Analysis Using CWSandbox. IEEE Security and Privacy Magazine, [online] 5(2), pp.32-39. Available at: https://www.syssec.ruhr-uni-bochum.de/media/emma/veroeffentlichungen/2012/12/14/CWSandbox-IEEESP2007.pdf [Accessed 11 Nov. 2019].

[11] Bayer, U., Moser, A., Kruegel, C. and Kirda, E. (2006). Dynamic Analysis of Malicious Code. Journal in Computer Virology, [online] 2(1), pp.67-77. Available at: https://sites.cs.ucsb.edu/~chris/research/doc/virology06_dynamic.pdf [Accessed 5 Nov. 2019].

[12] Song D. et al. (2008) BitBlaze: A New Approach to Computer Security via Binary Analysis. In: Sekar R., Pujari A.K. (eds) Information Systems Security. ICISS 2008. Lecture Notes in Computer Science, vol 5352. Springer, Berlin, Heidelberg. [online] Available at: https://www.comp.nus.edu.sg/~liangzk/papers/iciss08.pdf

[13] IMRAN, M., AFZAL, M. and QADIR, M. (2017). A comparison of feature extraction techniques for malware analysis. TURKISH JOURNAL OF ELECTRICAL ENGINEERING & COMPUTER SCIENCES, [online] 25, pp.1173-1183. Available at: https://pdfs.semanticscholar.org/d6ba/bbdd779de6c75abae95a54b4f8dd706d70ef.pdf [Accessed 2 Nov. 2019].

[14] Trinius, Philipp & Willems, Carsten & Holz, Thorsten & Rieck, Konrad. (2010). A Malware Instruction Set for Behavior-Based Analysis.. 205-216. [online] Available at: http://www.mlsec.org/malheur/docs/mist-tr.pdf

[15] Santos I., Nieves J., Bringas P.G. (2011) Semi-supervised Learning for Unknown Malware Detection. In: Abraham A., Corchado J.M., González S.R., De Paz Santana J.F. (eds) International Symposium on Distributed Computing and Artificial

Intelligence. Advances in Intelligent and Soft Computing, vol 91. Springer, Berlin, Heidelberg. [online] Available at: http://paginaspersonales.deusto.es/isantos/papers/2011/2011-Santos-SemiByteNgramas.pdf

[16] P. M. Comar, L. Liu, S. Saha, P. Tan and A. Nucci, "Combining supervised and unsupervised learning for zero-day malware detection," 2013 Proceedings IEEE INFOCOM, Turin, 2013, pp. 2022-2030. doi: 10.1109/INFCOM.2013.6567003. [online] Available at: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6567003

[17] Kim, S. (2016). Weighted K-means support vector machine for cancer prediction. [online] Available at: https://link.springer.com/article/10.1186%2Fs40064-016-2677-4 [Accessed 17 Nov. 2019].

[18] Bang, S. and Jhun, M. (2014). Weighted Support Vector Machine Using k-Means Clustering. [online] Taylor & Francis. Available at: https://www.tandfonline.com/doi/full/10.1080/03610918.2012.762388?scroll=top&needAccess=true [Accessed 19 Oct. 2019].

[19] T. Siriteerakul and V. Boonjing, "Support Vector Machine accuracy improvement with k-means clustering," 2013 International Computer Science and Engineering Conference (ICSEC), Nakorn Pathom, 2013, pp. 218-221. doi: 10.1109/ICSEC.2013.6694782 [online] Available at: https://ieeexplore.ieee.org/document/6694782

[20] Yao, Y., Liu, Y., Yu, Y., Xu, H., Lv, W., Li, Z. and Chen, X. (2013). K-SVM: An Effective SVM Algorithm Based on K-means Clustering. Journal of Computers, [online] 8(10). Available at: http://www.jcomputers.us/vol8/jcp0810-25.pdf [Accessed 4 Dec. 2019].

[21] Raman, K. (2012). Selecting Features to Classify Malware. [online] Available at: https://www.covert.io/research-papers/security/Selecting%20Features%20to%20Classify%20Malware.pdf

[22] C. Cepeda, D. L. C. Tien and P. Ordóñez, "Feature Selection and Improving Classification Performance for Malware Detection," 2016 IEEE International Conferences on Big Data and Cloud Computing (BDCloud), Social Computing and Networking (SocialCom), Sustainable Computing and Communications (SustainCom) (BDCloud-SocialCom-SustainCom), Atlanta, GA, 2016, pp. 560-566.doi: 10.1109/BDCloud-SocialCom-SustainCom.2016.87 [online] Available at: https://ieeexplore.ieee.org/document/7723741

[23] Chhabra, Dalbir Kaur R., "Feature selection and clustering for malicious and benign software characterization" (2014). University of New Orleans Theses and Dissertations. 1864. [online] Available at: https://scholarworks.uno.edu/td/1864

[24] Pai, Swathi, "A Comparison of Clustering Techniques for Malware Analysis" (2015). Master's Projects. 404. DOI: https://doi.org/10.31979/etd.nu7n-2cjh. [online] Available at: https://scholarworks.sjsu.edu/etd_projects/404

[25] Annachhatre, C., Austin, T. and Stamp, M. (2014). Hidden Markov models for malware classification. Journal of Computer Virology and Hacking Techniques, [online] 11(2), pp.59-73. Available at: https://www.researchgate.net/publication/272017073_Hidden_Markov_models_for_malware_classification [Accessed 6 Dec. 2019].

[26] Rabiner, L. (1989). A tutorial on hidden Markov models and selected applications in speech recognition. Proceedings of the IEEE, [online] 77(2), pp.257-286. Available at: https://www.ece.ucsb.edu/Faculty/Rabiner/ece259/Reprints/tutorial%20on%20hmm%20and%20applications.pdf [Accessed 16 Dec. 2019].

[27] En.wikipedia.org. (2020). Hidden Markov model. [online] Available at: https://en.wikipedia.org/wiki/Hidden_Markov_model [Accessed 7 Dec. 2019].

[28] Phon.ox.ac.uk. (2020). The three basic problems for HMMs. [online] Available at: http://www.phon.ox.ac.uk/jcoleman/new_SLP/Lecture_2/HMM_problems.htm [Accessed 18 Dec. 2019]