University of North Carolina at Charlotte

Department of Computing & Informatics

ITCS-6156 Machine Learning

Final Report

Fall 2019

SEMANTIC MALWARE DETECTION USING DEEP LEARNING

Key Terms: *Static Analysis, Dynamic Analysis, Code Obfuscation, malware, bytecode, RNN, CNN, DanQ, LSTM, Micro Model*

I. Introduction

A malware is any software that is intentionally designed to cause damage to a computer, server, client, or computer network. The industry has come a long way from the days of signature-based approach for identifying malwares. Detecting malwares using machine learning has been employed widely. We have made use of deep learning framework

Problem Statement

The biggest challenge that anti-malware faces today is the vast amounts of data and files which need to be evaluated for potential malicious intent. One of the main

reasons for these high volumes of different files is the fact that, in order to evade detection, malware authors introduce polymorphism to the malicious components. This means that malicious files belonging to the same malware "family", with the same forms of malicious behavior, are constantly modified and/or obfuscated using various tactics, such that they look like many different files.

In order to deal with the new malware types generated every day, new methodologies are needed to prevent any damage that may be caused by them. Malware files belonging within a specific malware family have a similarity embedded within them. We exploit this to enable our models to learn this intrinsic embedded feature to effectively classify a file as malware or not.

Motivation and Challenges

The growth of malwares has significant expansion over the years. Finding a solution for the various malware attacks was challenging, as they were constantly varying. Prediction of the malware pattern plays an important role to provide a better solution for malware attacks. It also helps to understand deep learning's importance in Information Security.

There are millions of new malicious malwares registered every day. According to [1], just in 2019, there are a total of 987 million new malwares registered. Last month alone, 15.56 million malwares have been identified and registered. Such rapid increase in malwares makes it very challenging for the anti-malwares to predict them and fight against them. The idea behind picking this particular area for our work was to come with an intelligent way to deal with the ever-evolving malwares that are produced in millions.

The biggest challenge we faced, as a team, was obtaining the file samples for training our model. Since malwares have malicious intent, it's dangerous if the files end up in the wrong hands. Thus, every dataset had the malware files encoded in a way where it cannot be misused anymore. While this is good from an end user standpoint, as academic researchers ourselves, we found it hard as it added another layer of complexity before we can start working on the samples.

The second challenge we had to overcome was the computational complexity of the resource intensive processes required for file decoding, training multiple machine learning models and deep learning. Our models were trained on a local machine with modest hardware.

Open Questions in the domain

1. How to implement DanQ in real time applications to predict the malware sequences or patterns?
2. How to extend the approach to predict more types of malwares within the malware family?

Concise Summary of Proposed Approach

Deep learning algorithms have attractive solutions for such problems because they are scalable with large datasets and are effective in identifying complex patterns from feature-rich datasets. They are able to do so because deep learning algorithms utilize large training data and specialized hardware to efficiently train deep neural networks (DNNs) that learn high levels of abstractions from multiple layers of non-linear transformations. Currently, DanQ approach has been used for 'Predicting the DNA sequences'. Malwares have a unique code structure when compared to normal programs. They behave differently when exposed to the virtual environment, in terms of its behavior, system interaction, and the effects on the host machine. From our research, we see similarities in predicting the DNA sequences with the malware file type prediction. We try to expand this DanQ approach for predicting the malware sequence, since it is similar to the prediction of DNA sequence. Like DNA have some specific patterns, malwares also have some specific patterns, which can be identified and used to foretell the malware type or family and prevent the malware attack.

Data Description

Here we have a set of known malware files representing a mix of 9 different families. Each malware file has an **ID**, a **20-character hash value** uniquely identifying the file, and a **Class label**, an integer representing one of 9 family names to which the malware may belong to **Ramnit, Lollipop, Kelihos_ver3, Vundo, Simda, Tracur, Kelihos_ver1, Obfuscator.ACY, Gatak.**

II. Literature Survey

[2] In the first stage the executables file is classified into malware using Static and Dynamic analysis and in second stage the malware executables file is categorized into corresponding malware family. In order to evaluate the performance of various classical machine learning classifiers, they have used the following methodologies, Adam optimizer and binary cross entropy are used as loss function. Batch normalization is used

to prevent overfitting. Sigmoid is used in output layer which gives 0 or 1 as output where 0 is benign and 1 is malware. [3] Transform a malware/benign file into a grayscale image (Once the dataset is ready, we should convert each file into a 256x256 grayscale image). CNN approach is used since it uses input layer (convolutional layer), one or more hidden layers and output layer where the early layers are used to detect the low level features like edges, length, height, width, etc.of the images and finally all the layers are combined at the final stage to produce a valuable result. CNN will automatically extract features & classify data into various classes (Feature learning). [4] Static analysis is perceived as vulnerable to obfuscation and evasion attack. Here comes the emergence of dynamic analysis. One problem with dynamic analysis is that the time it takes to perform the analysis is high. Also, if the malware data increases then simultaneously, dynamic analysis time also increases. Hence to improve the performance of the malware detection, behavior analysis is used. It was concluded that incremental technique in behavior-based analysis gives better performance in time and memory requirement than regular clustering. [5] Gray images are generated from the malware and benign files. For feature extraction and malware image classification DL approach (Convolutional Neural networks) is used. Using this approach, they have got an accuracy of 96%. [6] Malware detection is done through static malware analysis through opcode sequence pattern of malware. They have used disassembly tool IDA Pro to obtain opcode sequence of malware. In the last 2 stages LSTM approach is used. This proposed method has got the accuracy of 0.99 average AUC. [7] They are using dynamic analysis approach, through behavioral analysis of the malware. In order to detect the malware in its early stage through dynamic analysis, they are taking snapshots of the malware behavior in the first 20 seconds of the malware execution. Since the data (snapshots) they are collecting are based on time series sequence, they are using RNN methodology. [9] One interesting strategy that we came across included representing a malware as a grayscale image and then computing texture features. The images are then classified using KNN algorithm. The major drawback of the approach was they used very shallow learning techniques which are not very scalable with the growing number of malwares. In order to tackle these problems, we are making use of deep learning technique. The reason behind developing a deep learning method is that the machine learns the patterns in the images that it is presented with rather than needing human input to define the patterns that machine should look for in the image.

Based on our research, we will be using static analysis on the files as Dynamic analysis requires controlled environment to execute the files. This also requires deep domain knowledge to analyze the API calls made by the malicious file to the underlying operating system. Additionally, certain malwares are capable of evading the model during runtime.

III. Methods

Feature Extraction:

The data obtained from Microsoft contains .bytes representation of the infected files. This representation is then converted into features through unigram bag of words to obtain the feature set from the given data. Since bytecode contains hex code from 00 to FF (256 values) we end with 256-dimensional data frame.

We also add the size of the files as a feature as it was found useful for classification (observed through the EDA phase). This data is then normalized to bring the values between 0 and 1. Further malware samples belonging to classes other than the above-mentioned ones have been used. These files are converted to their respective object code through a custom python script.

For performing Deep learning, we had to convert the files to respective jpg images which were then used for malware classification. The hexdump of each file is taken as the input, which is then converted to integer. This results in a value that ranges from 0 to 255 (256 values in total). We encode the values to color gradients from white to black where 0 represents white and 255 represents black. This is encoded into a numpy matrix which is saved to local machine. This is the basis for our classification approach using Deep learning.

Classification Procedure:

Since we have 9 classes of malwares present, we are faced with a multi class classification problem. Once the files are vectorized and train-test split, we model classifiers through KNN, Logistic Regression, Random Forest, Xgboost. We start the modeling from a random model which is set as the baseline for all the other models to be measured against. The models are created as micro models **[8],** which helps dispose and retrain the model for a specific classifier alone as opposed to retraining the whole model, which we see fit in the ever-changing malware domain.

Performance Measure:

The performance measure we have used to evaluate the models is multi class log loss to evaluate the models. This metric quantifies the accuracy of a classifier by penalizing false classifications. Minimizing the log loss is equivalent to maximizing the accuracy of the classifier. For instance, if the model predicts a file to belong to a specific class with probability 1 the loss will be 0 and vice versa.

DAN-Q

DanQ is a novel hybrid convolutional and bi-directional long short-term memory recurrent neural network framework. In the DanQ model, the convolution layer captures regulatory motifs, while the recurrent layer captures long-term dependencies between the motifs in order to learn a regulatory 'grammar' to improve predictions. It improves considerably upon other models across several metrics. For some regulatory markers, it can achieve over a 50% relative improvement in the area under the precision-recall curve metric compared to related models.

The various layers of the DanQ model are as follows:

- **CNN** - It is used as the first layer of feature extraction from the images. Feature extraction includes grasping the patterns of the malware and this will be further utilized by the RNN layer for better prediction of the malware sequence patterns.
- **Pooling Layer** - This layer is used for dimensionality reduction in the malware images.
- **RNN** - Long Short-Term Memory (LSTM) - This layer makes use of the previous layers outputs as its input and will predict the next occurrence of the malware sequence. This will picturize the long-term relationship between the malware signature patterns.

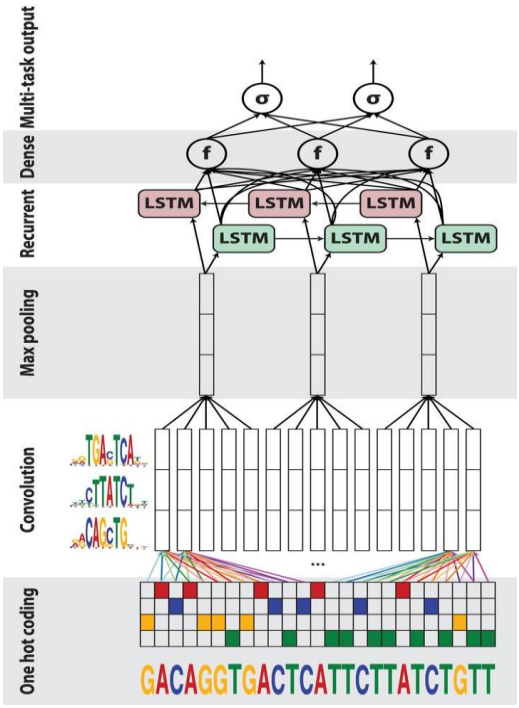
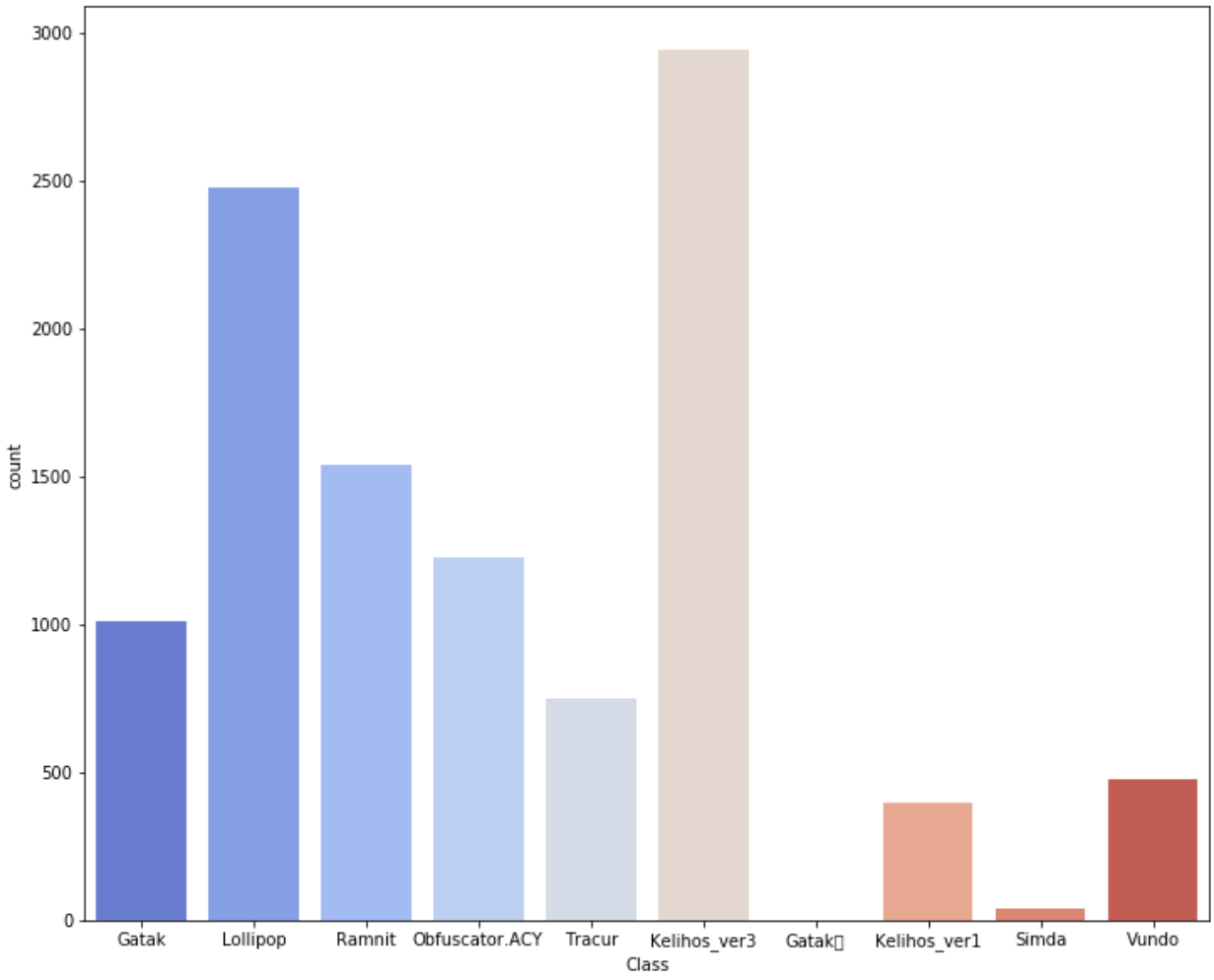


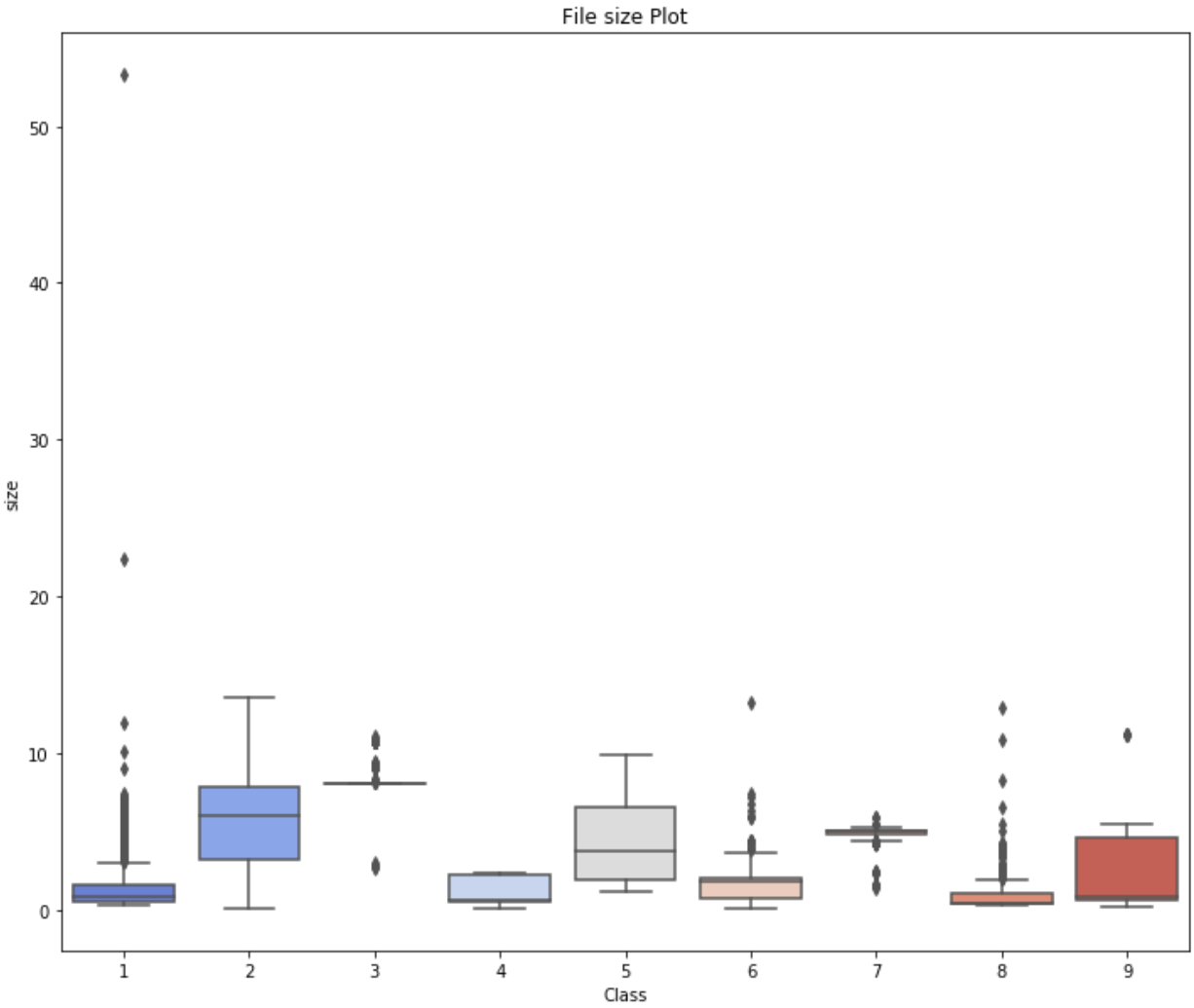
Fig-1: DanQ model working procedure

IV. Experiments

We start our experiments by analyzing the data we have. Files distributed by class represented in a count plot. We see that the files represented by the classes Simda, Gatak occur less frequently than others.



Also, we infer from [9] that the file size does play a major part in determining malware class. So, when we plot the files on size, we see a clear distinctive feature that can be used to classify the files.



Baselining with a random model:

We start by setting a baseline for classification models through a random model. This random model helps us better understand how good the other models are in comparison to what the random model could achieve.

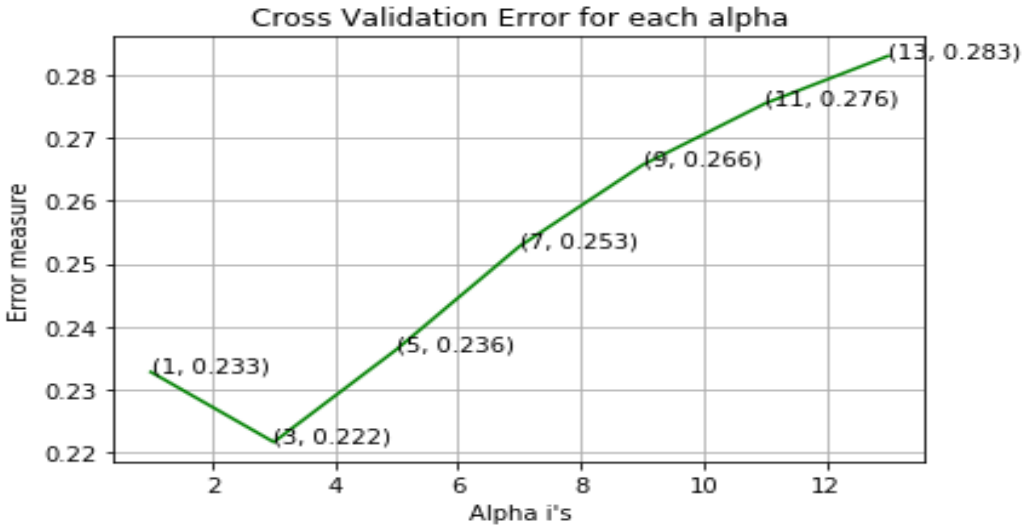
a) **Log Loss (Error estimate) for Random Model**
   Log loss on Cross Validation Data using Random Model: **2.45168427089047**
   Log loss on Test Data using Random Model: **2.490667522770678**
   Number of misclassified points: **89.28242870285189**

Classification Accuracy for different Machine Learning Algorithms based on the Log Loss metric:

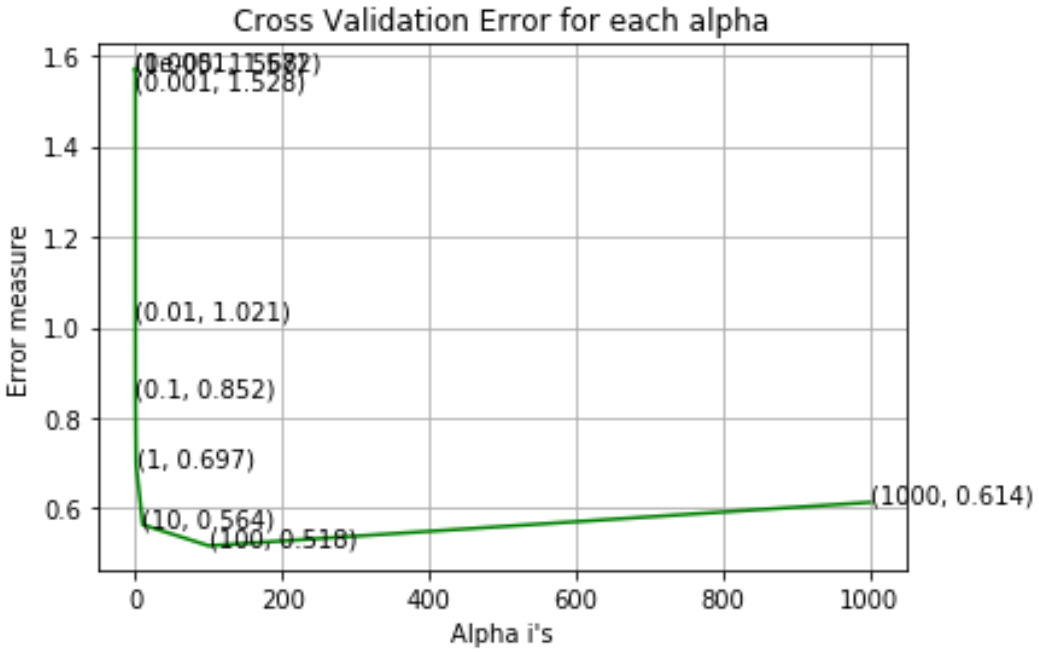
Machine learning algorithms	Classification Accuracies
Log Loss for K Nearest Neighbors (KNN)	The train log loss is: **0.1238851827467459** The cross validation log loss is: **0.22156196790637459** The test log loss is: **0.2363362424041084** Number of misclassified points **5.795768169273229**
Log Loss for Logistic Regression	log loss for train data : **0.5039599156189928** log loss for cv data : **0.5177373055381459** log loss for test data : **0.5369493206066701** Number of misclassified points : **11.86752529898804**
Log Loss for Random Forest	The train log loss is: **0.026426259944792196** The cross validation log loss is: **0.06626379228838201** The test log loss is: **0.09675510057278004** Number of misclassified points **2.3459061637534497**
Log Loss for XGBoost	The train log loss is: **0.023713446906898694** The cross validation log loss is: **0.05347666161312362** The test log loss is: **0.07584521137468213** Number of misclassified points **1.4719411223551058**

Visualization for Cross validation error for each alpha

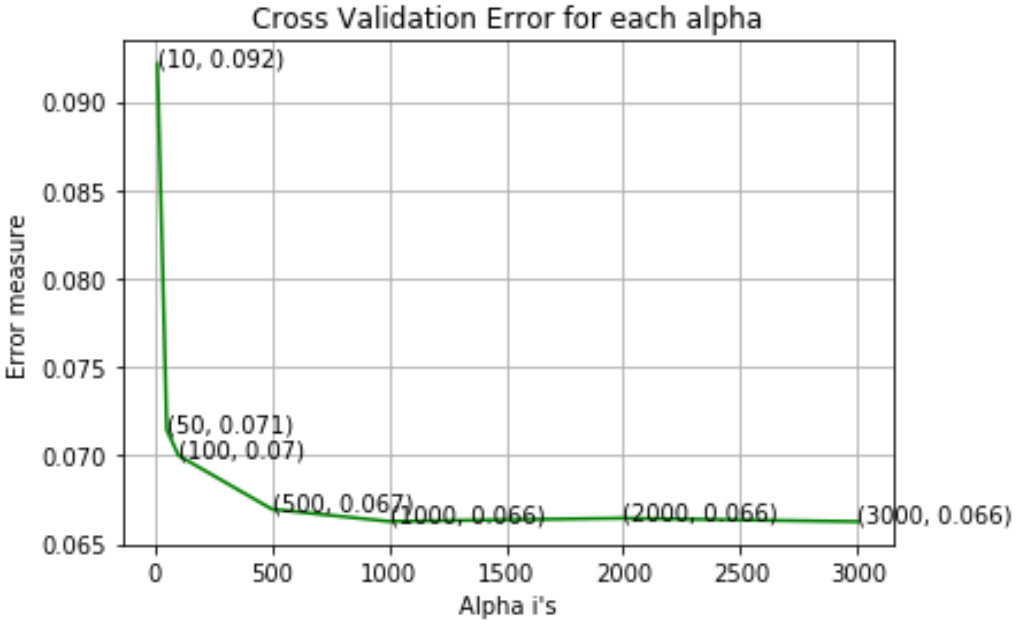
1. **KNN**



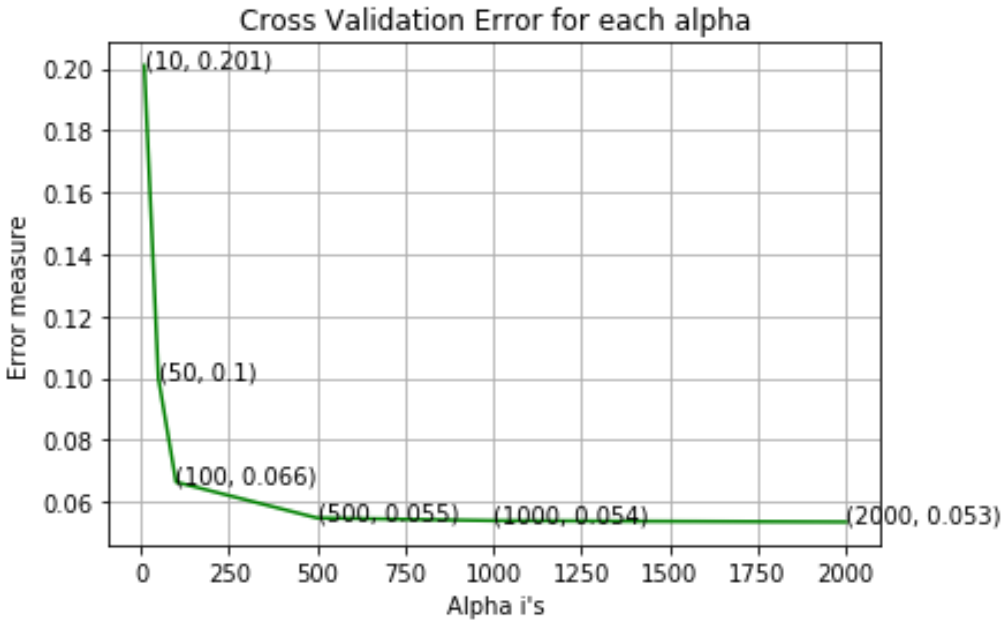
2. Logistic Regression



3. Random Forest



4. XGB Classifier



We were successfully able to bring down the log loss from 2.49 to a mere log loss of **0.07584521137468213** in classifying the malware files using the machine learning models. [9] Machine learning models are shallow learning methods which are not very scalable and needs to be feature engineered manually. Deep learning helps overcome this.

For DAN-Q, we start by down sampling the input image samples to a 64*64-byte image. This is then converted to a NumPy array. This enables us to featurize the images. From here, the first 320 values are considered for further processing.

V. Conclusion

We have learnt to work efficiently with large datasets (Over half TB). Got an overall knowledge of Machine Learning in Information Security. Not only was it helpful to learn image recognition through pixel matrix representation of files but gave us an insight to where such concepts can be applied. Additionally, we learnt how an industry problem can be solved to a reasonable extent with limited domain knowledge. None of our team members are specialists in Information Security in any measurable sense, but we learnt a lot along the journey, both in Information Security practices and in applied Machine Learning/Deep learning domains.

VI. References and Citations:

1. AV-Test “Malware statistics”, Retrieved from https://www.av-test.org/en/statistics/malware/
2. R.Vinaya kumar et al., “Robust Intelligent Malware Detection Using Deep Learning” in IEEE, 2019.
3. Ria Kulshrestha “Malware Detection Using Deep Learning (Malware Detection Using Convolutional Neural Networks)”, Retrieved from https://towardsdatascience.com/malware-detection-using-deep-learning-6c95dd235432
4. K. Rieck, P. Trinius, C. Willems, and T. Holz, “Automatic analysis of malware behavior using machine learning,” Journal of Computer Security, 2011.
5. Sunoh Choi Information Security Division ETRI Daejeon, South Korea ; Sungwook Jang ; Youngsoo Kim ; Jonghyun Kim, “Malware Detection using Malware images and Deep Learning,” in IEEE, 2017.
6. Renjie Lu, “Malware Detection with LSTM using Opcode Language”, arXiv:1906.04593v1 [cs.CR] 10 Jun 2019.
7. Mamoun Alazab, Sitalakshmi Venkataraman, Paul Watters, Moutaz Alazab, “Zero-day Malware Detection based on Supervised Learning Algorithms of API call Signatures” in Proceedings of the 9th Australian Data Mining Conference.
8. Matilda Rhode, Pete Burnap and Kevin Jones, “Early-stage malware prediction using recurrent neural networks”, ELSEVIER, 2018.
9. L. Nataraj, S. Karthikeyan, G. Jacob, and B. Manjunath, “Malware images: visualization and automatic classification,” in Proceedings of the 8th international symposium on visualization for cyber security. ACM, 2011, p. 4
10. “Understanding and detecting malware threats”, Retrieved from https://d3gpjj9d20n0p3.cloudfront.net/fortiguard/research/DetectingMalwareThreats.pdf
11. “Deep learning rises: New methods for detecting malicious PowerShell”, Retrieved from https://www.microsoft.com/security/blog/2019/09/03/deep-learning-rises-new-methods-for-detecting-malicious-powershell/