Empirical Study on Microsoft Malware Classification

Rohit Chivukula[1], Mohan Vamsi Sajja[2], T. Jaya Lakshmi[3], Muddana Harini[4]
University of Huddersfield, Huddersfield, United Kingdom[1]
Department of Computer Science and Engineering, SRM University, Andhra Pradesh, India[2, 3, 4]

***Abstract*—A malware is a computer program which causes harm to software. Cybercriminals use malware to gain access to sensitive information that will be exchanged via software infected by it. The important task of protecting a computer system from a malware attack is to identify whether given software is a malware. Tech giants like Microsoft are engaged in developing anti-malware products. Microsoft's anti-malware products are installed on over 160M computers worldwide and examine over 700M computers monthly. This generates huge amount of data points that can be analyzed as potential malware. Microsoft has launched a challenge on coding competition platform Kaggle.com, to predict the probability of a computer system, installed with windows operating system getting affected by a malware, given features of the windows machine. The dataset provided by Microsoft consists of 10,868 instances with 81 features, classified into nine classes. These features correspond to files of type asm (data with assembly language code) as well as binary format. In this work, we build a multi class classification model to classify which class a malware belongs to. We use K-Nearest Neighbors, Logistic Regression, Random Forest Algorithm and XgBoost in a multi class environment. As some of the features are categorical, we use hot encoding to make them suitable to the classifiers. The prediction performance is evaluated using log loss. We analyze the accuracy using only asm features, binary features and finally both. xGBoost provide a better log-loss value of 0.078 when only asm features are considered, a value of 0.048 when only binary features are used and a final log loss of 0.03 when all features are used, over other classifiers.**

Keywords*—*Multi-class classification; malware detection; XGBoost

I. Introduction

There are several kinds of malware that can infect a computer system. The number of malwares exceeds 800M in 2019 [1]. Detecting a given file as malware is one of the interesting research problems. Malware detection is challenging because the cybercriminals continuously change the way of attacking the computer systems, resulting in change in the features of malware software. There is a long-lasting confrontation between cyber security experts and malware creators. Machine learning algorithms can be efficiently used to identify whether a given file is malware or not. These algorithms require features/attributes of malwares. Malware files exist either in the form of byte files or assembly language files. Features can be successfully extracted from these files.

Microsoft is one of the major companies that develop anti-malware products. Microsoft has launched a challenge to detect malwares on Kaggle.com [2]. Microsoft has provided nearly half a tera byte of data consisting of malware files. The dataset given in [2] consists of 10,868 instances with 81 features, classified into nine classes.

Several works are available in the literature on malware classification. Ahmadi et al and Drew et al work on textual feature extraction from the challenge dataset [3,4]. The dataset is of huge size and it is difficult to work on a computer with moderate configuration. Hu et al. address scalability of the dataset [5]. Scofield et al. utilize an entity resolution strategy that merges syntactically dissimilar features [6]. Deep learning techniques are used in [7] and [8] to classify malwares based on the textual features. Narayanan et al. use the classifications like SVM, k-Nearest Neighbours and Artificial Neural Networks in their work [9]. More recent works can be found in [10].

In this work, we apply various multi class classification algorithms to predict the class of a given malware. The organization of this paper is as follows: Section 2 describes the research problem, dataset details, feature extraction and evaluation measures. Section 3 explains proposed approach to solve the problem. Section 4 details the experimental setup. Results are given in Section 5 along with some discussion. Conclusions are given at the end.

II. Problem Description

A. Problem Statement

Microsoft has classified malware into 9 classes. Microsoft malware classification is the problem of determining in which class of malware, a given file belongs to. This is a multi-class classification problem. To problem can be elaborated as follows: Given a file, the problem is to estimate the probability of the file belonging to each type of nine classes of malware. In multi-class classification problems, the algorithm predicts the class with maximum probability as the target class. But this kind of approach is not probable for malware classification because, estimation of the probabilities that belong to each class is valuable. For example, the probability of a file belonging to class 3 is 0.5 and class 4 is 0.4. If the problem is modelled such that the file belongs to class 3 considering the maximum probability, we will lose the information of the file may also be affected by class 4 with slight margin. Therefore, our approach computes probability of a given malware belonging to each of the 9 classes. The structure of the solution followed in this work is given in Fig. 1.

B. Dataset Description

The dataset available at Microsoft malware classification challenge webpage [1] has been used in this work. The organizers of this challenge have provided the training and test datasets separately. There are two kinds of files in this dataset. (1): .asm file and (2): .bytes file. Total train dataset consists of 200GB of data, out of which 50GB is .bytes files and 150GB is .asm files. There is a total of 10,868 .bytes files and 10,868 asm files, comprising 21,736 files in total, with nine possible class labels denoting 9 types of malwares. The number of files in each kind of class is given in Table I.

Fig. 2 shows the distribution of instances among nine classes of malware in the given dataset. It is understood from Fig. 2 that the problem is highly imbalanced with 27% of instances belonging to class 3 and 0.4% of instances in class 5. Classes 4, 5 and 7 occur very infrequently whereas, classes 1, 2 and 3 are the malwares that occur frequently.

Box plot on asm file size is given in Fig. 3. This indicates that class 2 and 5 have some similarity. But from class distribution plot in Fig. 2 implies that class 2 is frequently occurring, and class 5 is the least occurring class. This signifies that file size is useful in predicting class labels.

Predicted Probability	0.5	0	0	0	0.1	0.4	0	0	0
Class Label	1	2	3	4	5	6	7	8	9

Fig. 1. Structure of Solution.

TABLE I. DATASET DESCRIPTION

Class ID	Family name	#files	Type
1	Ramnit	1541	Worm
2	Lollipop	2478	Adware
3	Kelihos_ver3	2942	Backdoor
4	Vundo	475	Trojan
5	Simda	42	Backdoor
6	Tracur	751	TrojanDownloader
7	Kelihos_ver1	398	Backdoor
8	Obfuscator.ACY	1228	Any kind of obfuscated malware
9	Gatak	1013	Backdoor

A sample data points in both files are given in Table II.

TABLE II. SAMPLE DATA POINT

Sample data point in .asm file	
1	.text:00401000 assume es:nothing, ss:nothing, ds:_data, fs:nothing, gs:nothing
2	.text:00401000 56 push esi
3	.text:00401001 8D 44 24 08 lea eax, [esp+8]
Sample data point in .bytes file	
1	00401000 00 00 80 40 40 28 00 1C 02 42 00 C4 00 20 04 20
2	00401010 00 00 20 09 2A 02 00 00 00 00 8E 10 41 0A 21 01
3	00401020 40 00 02 01 00 90 21 00 32 40 00 1C 01 40 C8 18

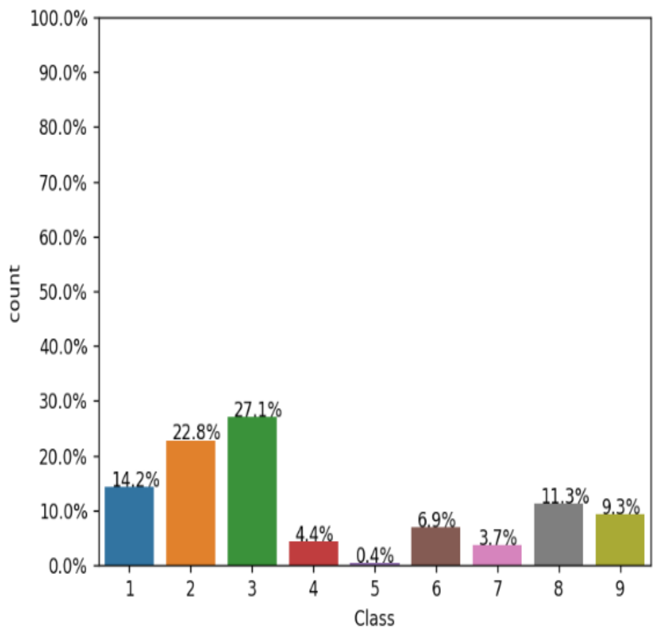

Fig. 2. Class Distribution of Instances.

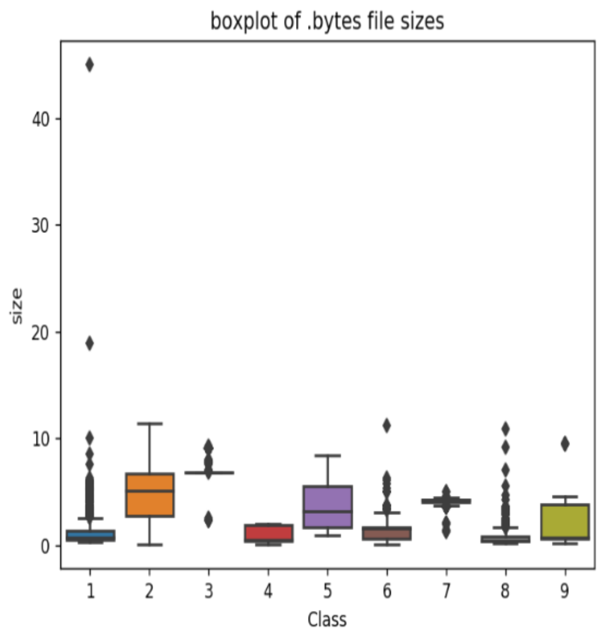

Fig. 3. Box Plot of Byte Files Sizes.

C. Feature Extraction

1) Features related to byte files: As byte files are represented using hexadecimal values, there are 256 distinct values. To pose this as text processing problem, we encode all these 256 values as unigram bag of words. The t-SNE diagram with different perplexities is shown in Fig. 4 and 5. This indicates that some classes are well separated from others. Features extracted from byte files: file_size, unigram_bag_of_words of size 256.

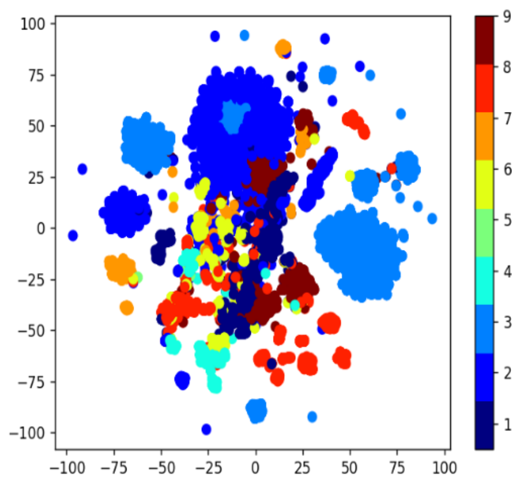

Fig. 4. t-SNE Diagram with Perplexity 50.

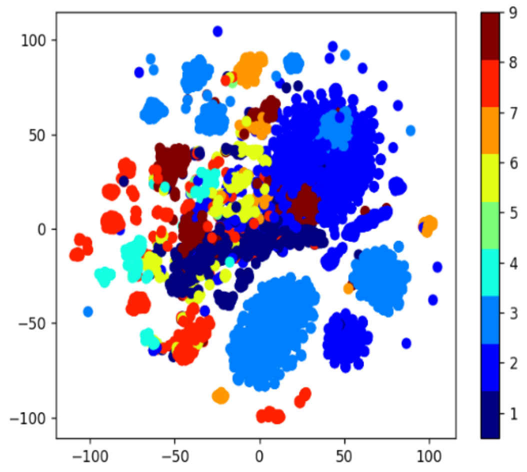

Fig. 5. t-SNE Diagram with Perplexity 30.

2) Features related to asm files: There are 10,868 files of asm of size around 150 GB. The initial observation of asm files says that there are Address, Segments, Opcodes, Registers, function calls and API related words in asm files. We have extracted 52 features from all the asm files. These features consist of file_size, bag of words related to 13 prefixes, 26 opcodes, 3 keywords and 9 registers. As the file size is huge, we use multi-threading with 5 threads to extract these features.

D. Evaluation Measures

1) Multi-class log-loss [17, 18]: Log loss is the common evaluation measure used for multi class classification problems. Multi class log loss is defined as follows:

$$-\frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{c} y_{ij}\log(p_{ij})$$

where, n is the number of instances,

c is the number of classes,

y_{ij} =1 if instance i belongs to class j and

p_{ij} is the predicted probability estimate of instance i belonging to class j.

A pure classifier yields a log loss of 0. The log loss value increases as the probability estimate by the chosen algorithm goes wrong. The aim of machine learning algorithm is to minimize the log loss value.

2) Confusion matrix: A confusion matrix for a n-class problem will be an n X n matrix, where columns correspond to the predicted class labels and the rows corresponds to the actual [19, 20, 21]. The main diagonal gives the correct predictions. That is, the cases where the actual values and the model predictions are the same. In malware classification problem, the matrix is of size 9 X 9. Each cell [i,j] represents number of points of class i are predicted to belong to class j. The ideal value of confusion matrix C can be

C[i,j] = 0 if i≠j

= Number of instances of class i(or j) if i=j

3) Precision: Precision is the fraction of correctly predicted instances out of total predictions for a given class [20, 21]. Precision is good if cost of wrong belongingness prediction to a class.

4) Recall: Recall is the capture of correct predictions among total instances belonging to the class [20, 21]. Recall is good if cost of identifying an instance which is a member of the class. If a patient who is cancerous is not predicted, it is a huge loss to the patient.

The proposed approach is explained in the next section.

III. Proposed Approach

Various machine learning algorithms are used in a multi class environment in this work. The proposed approach is shown in Fig. 6. The algorithms used in this work are briefly explained.

A. Random Model

In random model, we compute the probabilities of each class in the solution shown in Table I purely in random and normalise the sum to be 1. A random model gives us the worst possible log loss value of any algorithm. Any model performing worse than random model can be immediately rejected.

B. k-Nearest Neighbours (k-NN) Classifier [11]

k-NN algorithm is a lazy learning algorithm. It doesn't train the model in advance. The algorithm computes distance of test instance from k nearest instances in the training data. The class to which majority of k nearest neighbours belongs to is taken as the class of the test instance. Determining right k is a challenge in this algorithm. Hyper parameter tuning helps us in finding right k.

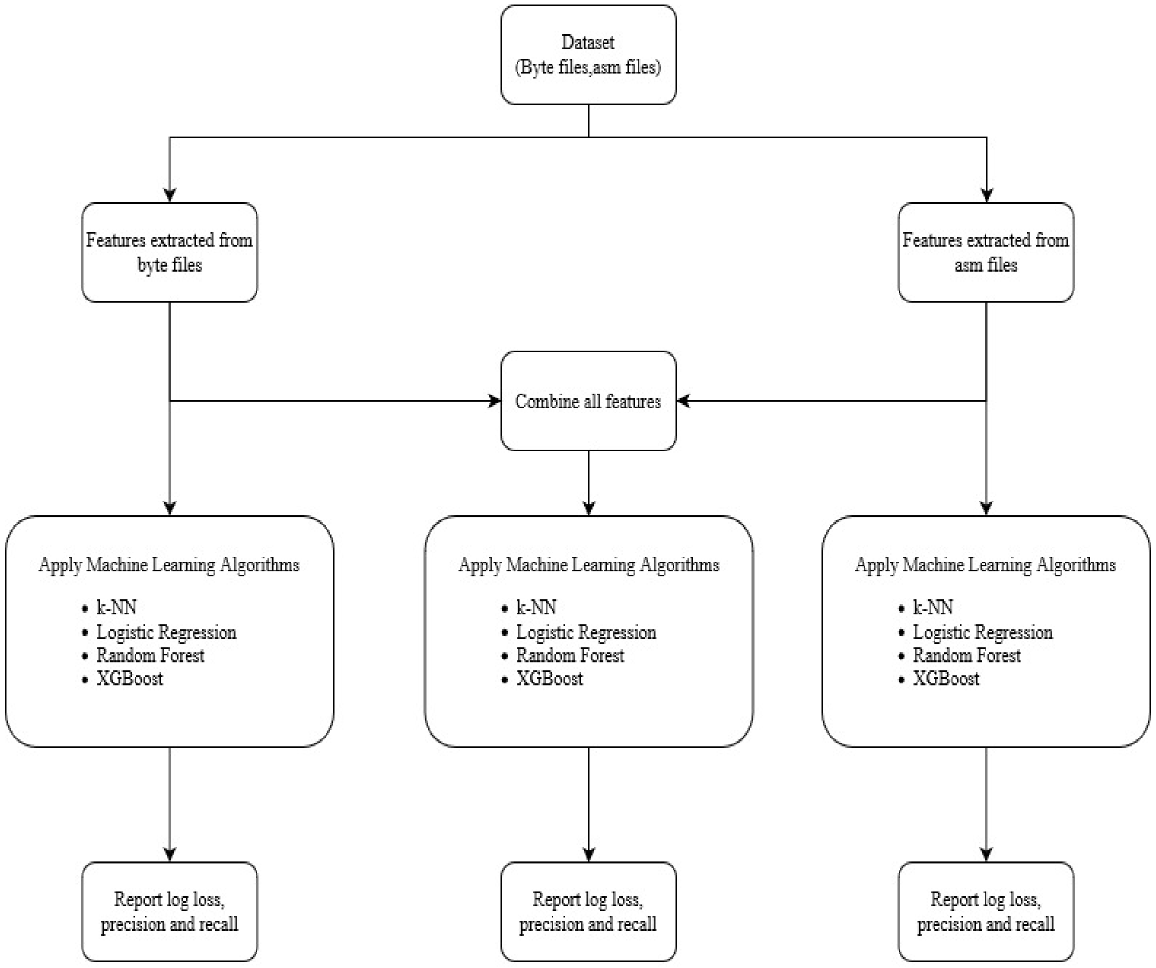
Dataset
(Byte files,asm files)
Features extracted from byte files
Features extracted from asm files
Combine all features
Apply Machine Learning Algorithms
• k-NN
• Logistic Regression
• Random Forest
• XGBoost
Report log loss, precision and recall

Fig. 6. Proposed Approach.

C. *Logistic Regression [12]*

Logistic regression is basically defined for binary classification problem. We use multinomial logistic regression [13], which is a variant of logistic regression for multi class problem. This algorithm predicts the probability of test instance belonging to a class in multi class environment.

D. *Random Forest [14]*

Random forest is an ensemble of decision trees trained with bagging. Random forest algorithm constructs n number of decision trees using train data. The class lable will be determined by majority voting of all these constructed decision trees. The decision tree algorithm can naturally handle multi class case too.

E. *XGBoost [15]*

XGBoost is an optimized distributed gradient boosting library. It utilises Gradient Boosting framework. XGBoost provides a parallel tree boosting method, which is very fast and accurate in many cases. XGBoost is a kind of ensemble. Ensemble learning constructs of a group of predictors that use multiple models and aggregates the performance of each tree. In Boosting technique, the errors made by previous models are tried to be corrected by succeeding models by adding some weights to the models.

Characteristics of XGBoost:

- XGBoost is used in regression as well as classification problems.
- Supports parallel processing.
- Can be able to manage memory very efficiently for large datasets exceeding RAM.
- Supports different kinds of regularizations which helps in reducing overfitting.
- Provides auto pruning of tree.
- Efficiently handles missing values.
- Has inbuilt Cross-Validation.
- Takes care of outliers to some extent.

All the classification algorithms chosen are sensitive to parameters. The experimental setup and parameter setting is discussed in the next section.

IV. Experimental Setup

This section describes the parameter selection of machine learning algorithms used for experimentation. Some classifiers we intend to use are sensitive to parameters. We perform hyper parameter tuning to fix the best parameter. The hyper parameter tuning is shown in Fig. 7 to 10.

k-NN classifier is sensitive to the value of *k* [16]. To find best *k*, we have tested the model with different values of *k* from 1 to 15. The model gives best log loss for *k*=1, as shown in Fig. 7. Therefore, we use *k*=1 in our experimentation.

For Random Forest classifier, we have tested with number of trees varying from 10 to 3000 (Fig. 9). With 1000 trees we could achieve best log loss and low misclassification error. Therefore, we use 1000 trees in random forest. We use XGBoost classifier with 500 trees, 500 estimators with a maximum depth of 5 and learning rate 0.05.

Any machine learning algorithm needs training and testing to determine the performance of the classifier. We split the dataset randomly into three parts train, cross validation and test with 64%, 16%, 20% of data respectively. We use 80% of data for training and 20% for testing.

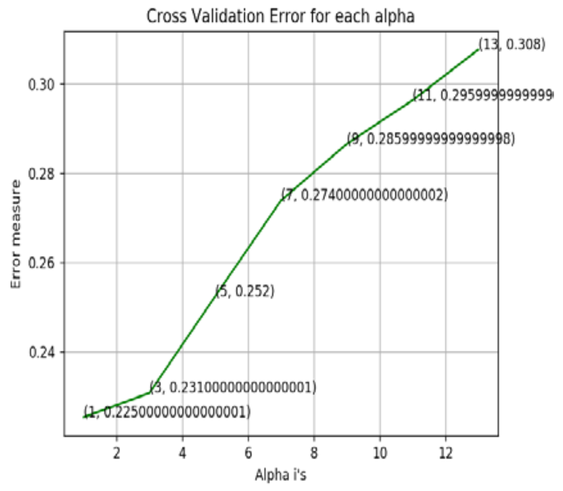

Fig. 7. Hyper Parameter Tuning for k-NN.

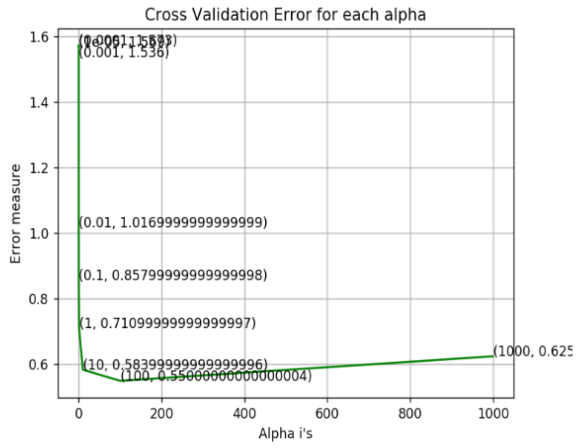

Fig. 8. Hyper Parameter Tuning for Logistic Regression.

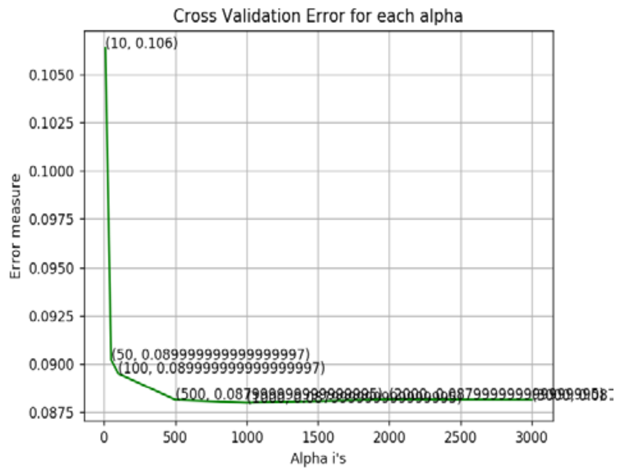

Fig. 9. Hyper Parameter Tuning for Random Forest.

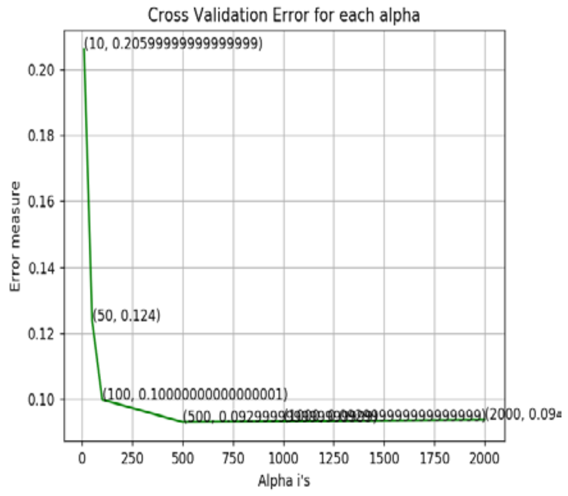

Fig. 10. Hyper Parameter Tuning for XGBoost.

V. Results and Discussion

We experiment with the features extracted from byte files, asm files individually and by combining them all. The following sections present the results.

A. Results on Byte Files

The log loss values on cross validation as well as test data are tabulated in Table III. Random forest classifier achieves low log loss value on cross validation data, whereas XGBoost is the winner on test data as well as misclassified errors.

From Table IV, we can see that the precision and recall of k-NN for class 5 is low compared to other classes. We guess that this is because of very few number of instances in class 5 (Fig. 1). From precision matrix, it is understood that there is a confusion between class 1 and class 5.

B. Results on Features Extracted from asm Files

The log loss values computed using features extracted from asm files are tabulated in Table V. XGBoost obtain better log loss on test data. But precision and recall for class 5 is improved using asm file features as shown in Table VI.

TABLE III. LOG LOSS RESULTS USING ONLY BYTE FILES

Algorithm	**Log loss**		**#misclassified points**
	cross validation	**test data**	
Random model	2.4561	2.4850	88.5000
k-NN	0.2253	0.2415	4.5078
Logistic Regression	0.5499	0.5283	12.3275
Random Forest	**0.0879**	0.0858	2.0239
XGBoost	0.0928	**0.0782**	**1.2419**

TABLE IV. PRECISION AND RECALL USING ONLY BYTE FILES

Classifier↓	**Class →**	1	2	3	4	5	6	7	8	9
KNN	Precision	0.88	0.97	1.00	0.97	0.75	0.89	0.94	0.96	0.91
	Recall	0.96	0.93	1.00	0.96	0.75	0.92	0.91	0.93	0.92
Logistic Regression	Precision	0.76	0.96	0.99	0.78	0.00	0.78	0.96	0.70	0.86
	Recall	0.78	0.89	0.99	0.97	0.00	0.68	0.95	0.88	0.70
Random Forest	Precision	0.94	0.99	0.99	0.95	1.00	0.95	1.00	0.95	0.98
	Recall	0.98	0.99	1.00	0.96	0.87	0.95	0.95	0.93	0.97
XGBoost	Precision	0.95	0.99	1.00	0.95	1.00	0.97	1.00	0.99	0.99
	Recall	0.99	0.99	1.00	0.98	0.75	0.98	0.96	0.95	0.98

TABLE V. LOG LOSS RESULTS USING ONLY ASM FILES

Algorithm	**Log loss**		**#misclassified points**
	cross validation	**test data**	
Random model	2.4561	2.4850	88.5000
k-NN	0.0958	0.0894	2.0239
Logistic Regression	0.4244	0.4156	9.6136
Random Forest	**0.0496**	0.0571	1.1499
XGBoost	0.0560	**0.0491**	**0.8739**

TABLE VI. PRECISION AND RECALL USING ONLY ASM FILES

Classifier↓	**Class →**	1	2	3	4	5	6	7	8	9
KNN	Precision	0.96	1.00	0.99	0.96	0.70	0.98	0.95	0.95	0.97
	Recall	0.97	0.99	0.99	0.91	0.87	0.95	0.97	0.94	1.00
Logistic Regression	Precision	0.89	0.97	0.84	0.97	0.00	0.93	0.47	0.89	0.95
	Recall	0.91	0.99	0.99	0.71	0.00	0.88	0.10	0.83	0.95
Random Forest	Precision	0.97	1.00	0.99	0.98	1.00	0.99	0.96	0.97	0.98
	Recall	0.99	1.00	0.99	0.95	0.87	0.96	0.98	0.96	0.99
XGBoost	Precision	0.97	1.00	0.99	0.98	1.00	1.00	0.96	0.98	0.98
	Recall	0.99	1.00	0.99	0.95	0.87	0.97	0.98	0.98	0.99

C. Results on Both Byte and asm Files

Random forest ensemble and XGBoost clearly obtain better accuracy in both cases of asm as well as byte files. We have used both features in these two models and present results in Table VII. When 257 features related to byte files as well as 53 features extracted from asm files are used for training, log loss result of XGBoost is improved for both cross validation as well as testing data from 0.048 to 0.031.

TABLE VII. LOG LOSS RESULTS USING ASM AND BYTE FILES

Algorithm	**Log loss**	
	cross validation	**test data**
Random Forest	0.0355	0.0401
XGBoost	0.0315	0.0323

VI. CONCLUSION

In this paper, we detect the type of malware that a given file belongs to. We use unigram model to construct bag of words from byte files as well as asm files. Random forest and XGBoost classifiers achieve a better log loss value of 0.031 over other classifiers used in this work. Usage of only byte files failed to detect some class of malware especially class 5, where the number of files are few, but the other information pertaining to asm files could succeed in detecting malwares belonging to all class. In future, we would like to apply advanced text retrieval features on byte files to improve the log-loss.

REFERENCES

[1] Beek, C., et al. "Mcafee labs threats report: August 2019." McAfee Labs (2019).

[2] https://www.kaggle.com/c/malware-classification/data.

[3] Ahmadi, Mansour, et al. "Novel feature extraction, selection and fusion for effective malware family classification." Proceedings of the sixth ACM conference on data and application security and privacy. 2016.

[4] Drew, Jake, Tyler Moore, and Michael Hahsler. "Polymorphic malware detection using sequence classification methods." 2016 IEEE Security and Privacy Workshops (SPW). IEEE, 2016.

[5] Hu, Xin, et al. "Scalable malware classification with multifaceted content features and threat intelligence." IBM Journal of Research and Development 60.4 (2016): 6-1.

[6] Scofield, Daniel, Craig Miles, and Stephen Kuhn. "Fast model learning for the detection of malicious digital documents." Proceedings of the 7th Software Security, Protection, and Reverse Engineering/Software Security and Protection Workshop. 2017.

[7] Kebede, Temesguen Messay, et al. "Classification of malware programs using autoencoders based deep learning architecture and its application to the microsoft malware classification challenge (big 2015) dataset." 2017 IEEE National Aerospace and Electronics Conference (NAECON). IEEE, 2017.

[8] Yuxin, Ding, and Zhu Siyi. "Malware detection based on deep learning algorithm." Neural Computing and Applications 31.2 (2019): 461-472.

[9] Narayanan, Barath Narayanan, Ouboti Djaneye-Boundjou, and Temesguen M. Kebede. "Performance analysis of machine learning and pattern recognition algorithms for malware classification." 2016 IEEE National Aerospace and Electronics Conference (NAECON) and Ohio Innovation Summit (OIS). IEEE, 2016.

[10] Zagi, Luqman Muhammad, and Baharuddin Aziz. "Searching for Malware Dataset: a Systematic Literature Review." 2020 International Conference on Information Technology Systems and Innovation (ICITSI). IEEE, 2020.

[11] Aha, David W., Dennis Kibler, and Marc K. Albert. "Instance-based learning algorithms." Machine learning 6.1 (1991): 37-66.

[12] Kleinbaum, David G., et al. Logistic regression. New York: Springer-Verlag, 2002.

[13] Böhning, Dankmar. "Multinomial logistic regression algorithm." Annals of the institute of Statistical Mathematics 44.1 (1992): 197-200.

[14] L. Breiman, “Random forests,” Machine Learning, vol. 45, no. 1, pp. 5–32, 2001.

[15] Chen, T., & Guestrin, C. (2016, August). Xgboost: A scalable tree boosting system. In Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining (pp. 785-794).

[16] Zhang, Shichao, et al. "Learning k for knn classification." ACM Transactions on Intelligent Systems and Technology (TIST) 8.3 (2017): 1-19.

[17] Ferri, César, José Hernández-Orallo, and R. Modroiu. "An experimental comparison of performance measures for classification." Pattern Recognition Letters 30.1 (2009): 27-38.

[18] Read, Jesse, et al. "Classifier chains for multi-label classification." Joint European Conference on Machine Learning and Knowledge Discovery in Databases. Springer, Berlin, Heidelberg, (2009): 254-269.

[19] Townsend, James T. "Theoretical analysis of an alphabetic confusion matrix." Perception & Psychophysics 9.1 (1971): 40-50.

[20] Powers, David MW. "Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation." arXiv preprint arXiv:2010.16061 (2020).

[21] Ting, K. M. "Confusion Matrix, Encyclopedia of Machine Learning and Data Mining." (2017).