Malware Classification

On

Microsoft Malware Dataset

Abstract—**Detection of unknown malware has been a challenge to governments, businesses and end-users. The dependency on default signature-based detection methodologies has proven to be inefficient as years pass. Experts and researchers have tried to investigate the use of data analytics techniques in order to accurately detect unknown malware.**

The initial goal of this project is to predict a Windows machine's probability of getting infected by various classes of malware, considering unique properties of respective machine. Hence, the data required for malware prediction can be any information about the state of a computer which is hit by a malware attack. As there are several classes of malware attacks, machines may react differently in response when attacked. Therefore, it is recommended to collect a huge and varied quantity of data about computers that are undergone through attacks. Most of the data comes from the system behavior of the machine and the type of the machine.

Objective: Analyze the different methods and compare their solutions in order to provide the pros and cons of different methods and approaches that were used to tackle this problem and derive a conclusion on which method is more effective for predicting the malwares effectively into their respective families based on multiple evaluation metrics.

INTRODUCTION

In today's digital economy, most business and individuals are reliant on computer networks and information systems to process and store digital content. Modern businesses are not only transforming their paper-based content into digital forms but also generating new business models considering their current digital assets. As a result, cyber threats pose a significant challenge when technology infrastructure is compromised by malicious attacks. Many of these attacks involve using malware. Malware is a malicious software that infects the host machine without knowledge of its owner. It then carries out unauthorized activities, such as damaging the host machine or steal valuable data. Malware activities can be local to the host machine or propagated through the network to infect other machines as well.

One of the major challenges faced by anti-malware software today are the vast and varied amounts of data which needs to be evaluated to identify potential malicious intent. Malwares are classified into two categories - first generation malware and second-generation malware. The family/class of malware depends basically on how the malware affects system, functionality of the program and growing mechanism. The first-generation malware deals with the idea that the structure of malware remains consistent, while the second-generation malware deals with keeping the action consistent, and the structure of malware changes, after every iteration resulting in the generation of new structure. Such dynamic characteristic trait of the malware makes it harder to identify, and secure. The most historical and famous techniques considered for malware detection are heuristic based, signature based and normalization. In past years, machine learning has been an admired approach for malware defenders.

I. DATASET

The malware dataset comprises of a set of known malware files representing a mix of 9 different classes/families and is almost half a terabyte when uncompressed. Each malware file comprises of an identifier, a 20-character hash value which uniquely identifies the file and a class label, associated is an integer representing one of the 9 family names to which the malware is classified to belong. For each file, the raw data contains the hexadecimal representation of the file's binary content. The dataset also includes a metadata manifest, which is a log containing various metadata information extracted from the binary, such as function calls, strings, etc. The motive is to classify malware to one of the 9 classes. The dataset can be downloaded from the Kaggle website under the name "BIG Malware dataset from Microsoft".

Class	# of Samples	Type
Ramnit	1541	Worm
Lollipop	2478	Adware
Kelihos_ver3	2942	Backdoor
Vundo	475	Trojan
Simda	42	Backdoor
Tracur	751	TrojanDownloader
Kelihos_ver1	398	Backdoor
Obfuscator.ACY	1228	Any Obfuscated
Gatak	1013	Backdoor

Table 1. Malware class samples in dataset

Below is the description of some of the major features from the dataset :

a) String n-grams:

Strings can be found in executables in the form of comments, URLs, print message, function names, library imports commands. Sometimes these strings are ofter unique to malware which can be used as an identifier. N-grams is a technique in text mining where it represents a number of consecutive terms or characters in a phrase.

b) Byte-sequence n-grams:

Byte-sequence is the translation of the executables in the structure of a machine code. The result is a series of hexadecimal file representation. Researchers also apply n-grams technique as well on byte-sequence.

c) PE Headers:

PE files are standard file structure in Windows Operating system. It encapsulates all the information needed by the OS to load the file. PE files contain sections that stores references to DLL libraries required to be imported and exported, code and data required to run the file, and resources needed by the executable. Some variables extracted from PE Headers can be used to detect malware.

d) DLL Libraries:

PE files contain reference to DLL library in the rdata section of the PE Header. It lists all the required DLL imports and exports.

e) DLL Function Calls:

PE Headers specify which libraries to import and export, and which function within those libraries will be executed at run-time.

f) OpCode:

Operational Code is a sequence of machine instructions that executes in succession until the end of the execution or until another condition met. OpCode can only be extracted after disassembly of the executables.

It is identified that as dataset grows, there is an issue of scalability. This issue increases time complexity, storage requirement and decreases system performance. To overcome these issues, reduction of data set is necessary. Two approaches can be used for data reduction namely Instance Selection and Feature Selection. In our approach, Instance Selection is used to reduce the number of instances(rows) in dataset by selecting most appropriate instances. On the other hand, Feature Selection is used for the selection of most relevant attributes(features) in dataset. These two approaches are very effective in data reduction as they result in less storage, time complexity and improve the accuracy of classifiers.

II. Literature Review

Traditionally, detecting malicious files is being done using signature-based methods by anti-virus software. The downside of using such technique is that it does not detect new malware because its signature is not in their repository. Thus, the exposure to threats posed by that malware is significantly high until the anti-virus providers update their malware signature repository. The inability of signature-based techniques to detect unknown malware motivated researchers to investigate the use of machine learning techniques to identify threats posed by unknown malicious attacks. In this section, we investigate the recent proposals made by researchers in detecting malware using machine learning algorithms. We compare the results of those proposals in a more robust manner.

A typical machine learning experiment in malware analysis space starts with collecting a dataset of malicious and benign executables. The dataset is then divided into training and testing sets; the former to train the classification model, and latter to evaluate the accuracy against unknown occurrences. Some of the models, covered in this survey, used cross-validation as a way to split the data into training and testing set. Usually, such models used ten-fold cross-validation in which the data is partitioned into ten equal pieces, the model is then trained on nine pieces and evaluated against the last piece. This operation is repeated ten times until all the pieces are tested at least once.

III. Methodology

A. Classification

a) Logistic Regression

Logistic regression uses a sigmoid function to help map the probabilities into any values between 0 and 1. The advantage of Logistic regression is that it is good for binary classification and suitable for this project as the probabilities of whether malware is detected are any

values between 0 and 1. It is also easy to implement and train. However, the disadvantages are that we need to identify important independent variables for training so during data preprocessing, we need to identify the correlated variables and remove them and keep the important variables. Additionally, Logistic regression cannot solve nonlinear problems.

b) KNN

The k-nearest neighbors (KNN) algorithm is a simple, supervised machine learning algorithm that can be used to solve both classification and regression problems.

We have used this in our project as our dataset is huge and KNN provides optimal results as it considers only the k nearest neighbors instead of computing over the entire dataset. KNN algorithms use data and classify new data points based on similarity measures. Classification is done by a majority voting to its neighbors. To optimize KNN, it is run over multiple values of k(neighbors) and find out the value of k providing best performance.

c) Decision Tree

A decision tree is a flowchart-like tree structure where an internal node represents feature, the branch represents a decision rule followed by each leaf node representing the outcome. The topmost node in a decision tree is the root node and it learns to partition in a recursive manner on the basis of the attribute value.

We have used Decision Tree in our project, as it is a white box type of ML algorithm. It shares internal decision-making logic, which makes it easily understandable and is not available in the black box type of algorithms. Its training time is faster compared to the neural network algorithm. Also, It is a distribution-free method, which does not depend upon probability distribution assumptions. Decision trees can handle high dimensional data with good accuracy and are easy to understand and interpret.

d) Naïve Bayes

Naïve Bayes is one of the simplest supervised learning algorithm and statistical classification technique based on Bayes theorem.

We have used this in our project as it considers the effect of a particular feature in a class is independent of the other features and these classifiers have high accuracy and speeds on large datasets.

e) Support Vector Machine

SVM is a classification algorithm that creates a separation line called hyperplane, which is the largest distance between the extremes of classes from dataset. We have used this in our project as it works best with extreme cases of a dataset and for multi-class classification, the problem is split into binary problems and then done one-versus-one or one-versus-all comparison.

IV. Feature Selection

Feature selection is extremely important in machine learning primarily because it serves as a fundamental technique to direct the use of variables to what is most efficient and effective for a given machine learning system. In our dataset, we have 1805 features. So, Sequential Forward Selection or Sequential backward Selection would not work easily on our dataset. However, We still try to perform Sequential Forward Selection for 3 and 10 features. For the trial purpose, we have divided our dataset into 5 parts in a Stratified Sampling manner. For approx. more than 2000 instances and 1804 features, google collab TPU took 10 min for 3 features selection and took more than 40 min to select 10 features. So, It would not be a feasible solution to perform feature selection in 10k instances.

So, we decided to perform different features selection method for our project. Since Decision tree performs very well on our dataset, We took this method to one step further. We tuned the decision tree model for different hyper parameter and choose the parameters which give the highest accuracy in our classification. Later this tuned model is used to select top 10/25/50/75/100 features from our dataset based on the accuracy.

V. Evaluation Measures

The metrics presented below were used to evaluate the algorithms performance. There are four basic entities used for overall evaluation :

i. True Positive(TP) is the number of actual positives that were correctly classified.
ii. True Negative(TN) is the number of actual negatives that were correctly classified.
iii. False Positive(FP) is the number of actual positives that were identified as negatives.
iv. False Negative(FN) is the number of actual negatives that were identified as positives.

a) Confusion Matrix :

A confusion matrix is a table that allows for the visualization of the performance of a model by showing which values the model thought belong to which classes. It has a N ×N size, where n is the number of

classes, with the columns representing the actual classes and the rows the predicted classes.

b) Precision :

The precision is an evaluation measure evaluating the model by calculating the fraction of correctly identified positives. Its formula is:

$$\text{Precision} = TP / TP + FP$$

c) Accuracy :

The accuracy is evaluation measure evaluating the model by calculating the fraction of correct predictions over the total number of predictions. Its formula is:

$$\text{Accuracy} = TP + TN / TP + TN + FP + FN$$
$$= TP + TN / P + N$$

d) Recall Score :

The recall score evaluates the model by calculating the fraction of actual positives that were correctly identified. Its formula is:

$$\text{Recall} = TP / TP + FN$$

e) Support Score :

The support score is the number of occurrences of a class in the total number of predictions. It is used as part of the F1 - micro-average metric.

f) F-1 Score :

F-1 score is basically the harmonic mean of both precision and recall score. For multi-class classification, We have used micro-averaging instead of macro-averaging, as clearly the data has differences in class sizes.

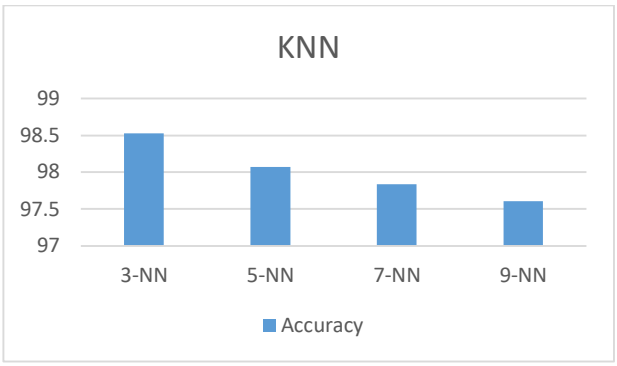
VI. Experimental Setup

The experiment was conducted on a machine operating on Windows 10 pro-64-bit and the programming was done in Spyder IDE using Python3.7 language. Loading the data and data handling in models was done using Pandas library, Arithmetic calculations were performed using NumPy library, Plots were visualized using MatPlot library and Models were implemented using Sklearn library.

VII. Result Analysis

A) K-Nearest Neighbor

Perhaps the most naive approach would be the k-nearest neighbor algorithm, KNN in short. The name already explains that an input data's class is determined by the K nearest neighbors.



We trained the model with different values of k = 3, 5, 7 and 9. Accuracy of the model decreased substantially with the increased in number of neighbors. The accuracy almost fell by 1% when increased value of k=3 to k=9. We then trained and tested KNN models, using a single feature, to see how accurate a single feature can achieve.

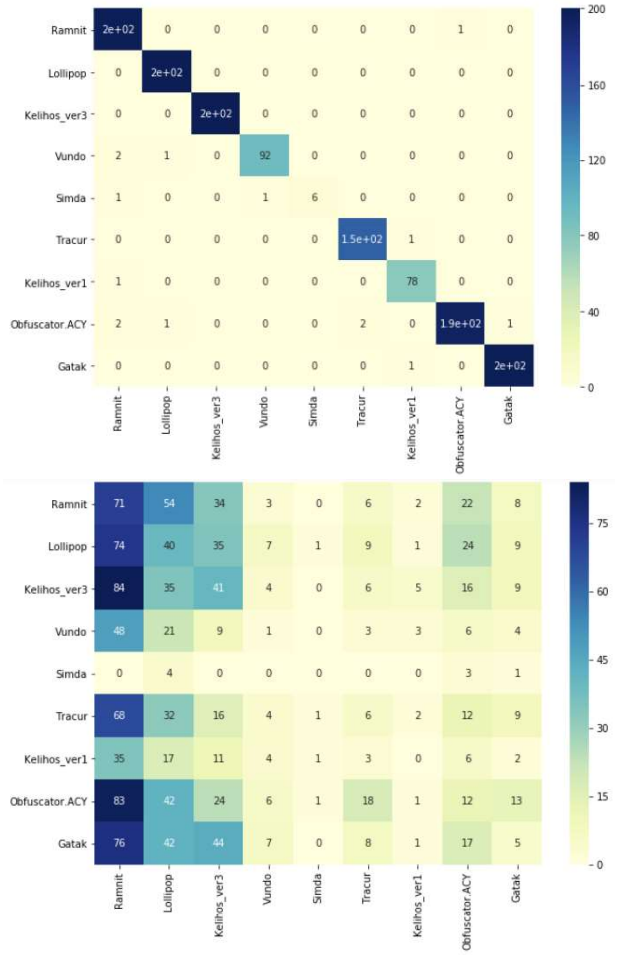


Fig 1. Confusion matrix (i) KNN trained with 4gram (ii) KNN trained with dll

Only '4gram' achieved high performance when used alone, Whereas other features could not outperform. As visible in Fig 1.(i) 4gram is

concentrated around its diagonal compared to Fig 1.(ii) dll and other features the matrix is concentrated to the left, resulting in increased concentration of false negatives, which indeed decreases Recall and Accuracy.

Further to improve the accuracy of the model, We removed the classes which were a rare occurrence of malware type. The overall accuracy improved to 99.43%(Table 2). Meanwhile, we tracked the performance over both random_state and shuffle=true, Its impact on accuracy.

Accuracy=99.42708333333333%

	precision	recall	f1-score	support
1	1.00	0.99	1.00	271
2	1.00	1.00	1.00	519
3	1.00	1.00	1.00	582
4	0.99	0.98	0.99	105
6	0.97	0.99	0.98	170
7	0.97	1.00	0.99	68
9	1.00	0.98	0.99	205
accuracy			0.99	1920
macro avg	0.99	0.99	0.99	1920
weighted avg	0.99	0.99	0.99	1920

Table 2. KNN Evaluation matrix after removing Simda(5) and Obfuscator.ACY(8) classes

B) Logistic Regression

Logistic Regression has higher time complexity also it takes more time to compute compared to Decision Tree and SVM. Logistic gave and accuracy of 98.66% which is fairly decent and higher when compared with KNN. It also gave a higher precision, recall and F1-score for the rare classes of malware(Simda & Obfuscator) when compared with KNN over the entire dataset.

Furthermore, to improve performance, Parameter tuning was performed considering penalty, solver(liblinear, newton-cg, lbfgs, saga), C, and max iterations.

Accuracy=98.66605335786569%

	precision	recall	f1-score	support
1	0.97	0.98	0.98	313
2	1.00	1.00	1.00	489
3	1.00	1.00	1.00	614
4	0.99	0.99	0.99	86
5	0.86	0.86	0.86	7
6	0.96	0.96	0.96	136
7	0.99	1.00	0.99	87
8	0.97	0.94	0.95	245
9	0.99	1.00	1.00	197
accuracy			0.99	2174
macro avg	0.97	0.97	0.97	2174
weighted avg	0.99	0.99	0.99	2174

Table 3. Logistic evaluation matrix

C) Decision Tree

Decision Tree among all the models had the least computational time. It gave an overall accuracy of 98.52% without tuning over entire dataset. When compared with other models, Decision Tree was the fastest though it had to compromise with accuracy.

To overcome the accuracy part, We tuned the Decision Tree on 5-fold cross validation set considering parameters such as criterion, max_depth, min_weight_fraction_leaf and splitter. Its accuracy increased to 98.84% and the best parameters found were {'criterion': 'gini', 'max_depth': 10, 'min_weight_fraction_leaf': 0.0, 'splitter': 'best'}

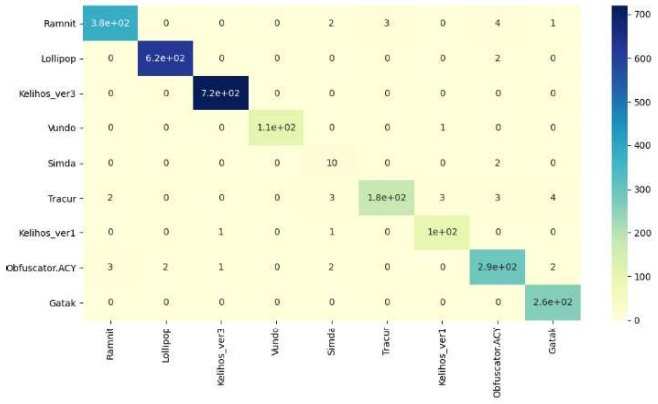


Fig. 2 Decision Tree confusion matrix

D) Naïve bayes

In Naïve Bayes we have used both Gaussian and Multinomial along with 10-fold cross validation to compare both performances. Overall performance accuracy of Naïve Bayes was low when compared to other models, but it was faster. For GaussianNB,

Accuracy	*Fold 1*	*Fold 2*	*Fold 3*	*Fold 4*	*Fold 5*	*Fold 6*	*Fold 7*	*Fold 8*	*Fold 9*	*Fold 10*
Multinomial NB	*94.027*	*95.694*	*94.583*	*95.277*	*94.722*	*95.277*	*96.111*	*94.305*	*94.436*	*93.741*
Gaussian NB	*93.259*	*93.006*	*92.392*	*92.392*	*92.883*	*91.533*	*93.742*	*90.797*	*92.269*	*92.638*

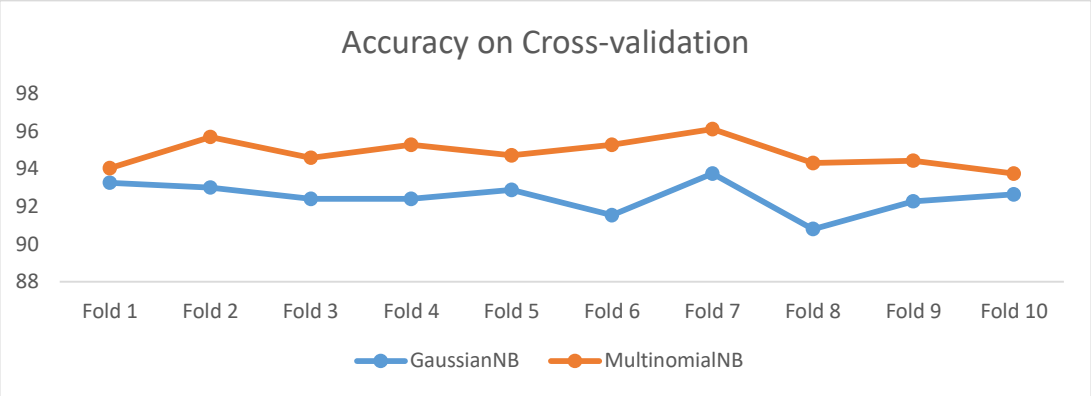


Table 5.Performance accuracy Naïve Bayes on 10- folds

accuracy came out to be 92.1% when compared to 95.3% accuracy of MultinomialNB.

```
[[378   2   0   0   0   4   2   0   2]
 [  7 585   1   0   0  13   0   0   5]
 [  0   0 740   0   0   1   0   0   0]
 [  0   0   0 109   0   7   0   0   0]
 [  0   0   0   0   0   0   0   0   0]
 [  1   0   2  11   0 167   3   0   5]
 [  1   3   0   0   0   7  78   0   4]
 [  0   0   0   0   0   0   0   0   0]
 [  4   0   3   0   0  24   1   0 230]]
```

	precision	recall	f1-score	support
1	0.97	0.97	0.97	388
2	0.99	0.96	0.97	611
3	0.99	1.00	1.00	741
4	0.91	0.94	0.92	116
6	0.75	0.88	0.81	189
7	0.93	0.84	0.88	93
9	0.93	0.88	0.91	262
accuracy			0.95	2400
macro avg	0.92	0.92	0.92	2400
weighted avg	0.96	0.95	0.95	2400

Table 4. MultinomialNB (i) simple confusion matrix (ii) evaluation matrix

Overall MultinomialNB outperformed GaussianNB in terms of accuracy across each fold when considered 10-fold cross validation as shown in Table 5.

E) Support Vector Machine (SVM)

SVM in terms of computation speed was faster than Logistic Regression but slower than Decision Tree. Initially SVM provided accuracy of 48%, But after processing(i.e., Normalization, Shuffling, Scaling, etc.) its accuracy almost doubled to 98.2% also reducing the computation time to half. Further to improve the accuracy we performed parameter tuning with 5-fold cross validation. It did improve the accuracy to 98.9% but along with large increase in computation time 3886sec(1 Hour) compared to just 56sec without tuning.

```
time cost:
56.320324182510376 seconds
Classifiction Report :
```

	precision	recall	f1-score	support
1	0.97	0.98	0.98	321
2	1.00	0.99	0.99	471
3	1.00	1.00	1.00	570
4	0.97	0.98	0.97	93
5	1.00	0.55	0.71	11
6	0.96	0.98	0.97	152
7	0.97	0.99	0.98	90
8	0.94	0.93	0.94	247
9	1.00	0.99	0.99	219
accuracy			0.98	2174
macro avg	0.98	0.93	0.95	2174
weighted avg	0.98	0.98	0.98	2174

Table 6. SVM evaluation matrix without parameter tuning

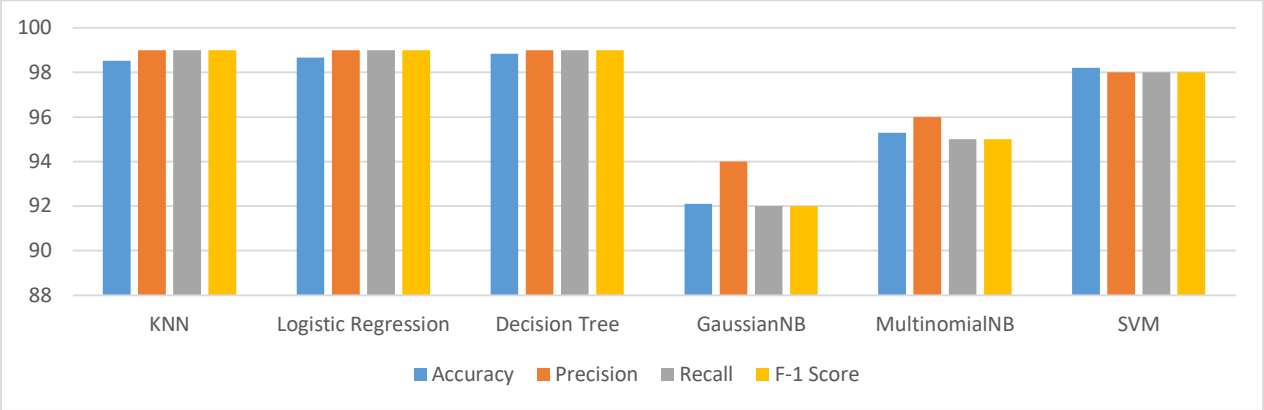


Fig. 3. Evaluation metrics over all models

VIII. Conclusion

The problem addressed in this paper was classifying malwares using signature techniques. The report clearly concludes that Data analytics methods were able to classify each malware class with a maximum accuracy of 99.49%. Initially, we deduced the accuracy of each basic model. Amongst all, Decision Tree performed the best followed by Logistic and KNN. Due to high accuracy and less computation time of Decision tree, we implemented feature selection over it and got the best 75 features which boosted the accuracy.

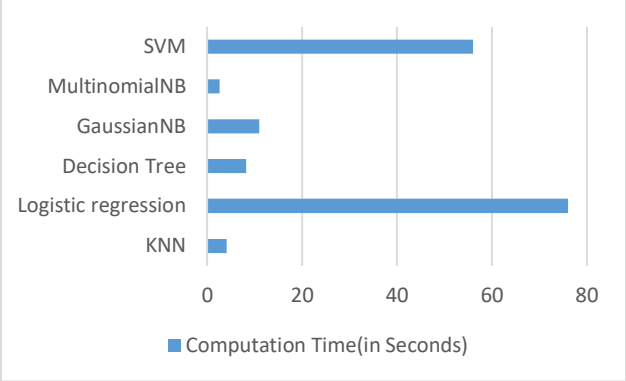


Fig. 4. Time Complexity of all models

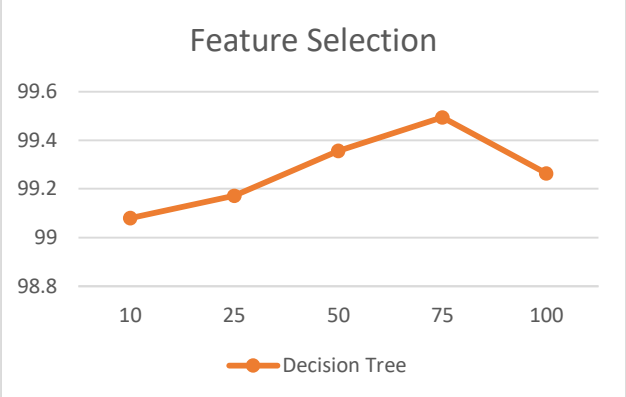


Fig. 5. Finding best top features in Decision tree

IX. Appendix

[1] Final code python file contains the main source code.

[2] Decision Tree tuning python file contains code for tuning Decision Tree and further to feature selection.

[3] Feature Selection prediction python file contains code for feature selection referring to Decision Tree.

[4] Logistic Regression tuning python file contains tuning for logistic regression. Has been made separate file due to its high time complexity.

[5]https://www.kaggle.com/muhammad4hmed/malwaremicrosoftbig

References

[1] Sanjay Sharma, Rama Krishna, and Sanjay Sahay,"Detection of Advanced Malware by Machine Learning Techniques" BITS Pilani Goa

[2] Y. Elovici, A. Shabtai, R. Moskovitch, G. Tahan, and C. Glezer, "Applying Machine Learning Techniques for Detection of Malicious Code in Network Traffic," Annual Conference on Artificial Intelligence. Springer Berlin Heidelberg, pp. 44–50

[3] Nicolas-Alin Stoian, "Machine Learning for Anomaly detection in IoT networks: malware analysis on IoT 23 dataset" University of Twente

[4] A. Shabtai, R. Moskovitch, C. Feher, S. Dolev, and Y. Elovici, "Detecting unknown malicious code by applying classification techniques on OpCode patterns," Security Informatics, vol. 1, no. 1, p. 1, 2012.

[5] Royi Ronen, Marian Radu, Corina Feuerstein, E. Yom, M, Ahmadi,"Microsoft Malware classification challenge"22 feb 2018

[6] Mohamad Baset, H. Ragab,"Machine Learning for Malware detection"Dec 2016 Heriot Watt

[7] D. Bilar, "Opcodes As Predictor for Malware," International Journal of Electronic Security and Digital Forensics, vol. 1, no. 2, pp. 156–168, 2007.

[8] S. K. Sahay and A. Sharma, "Grouping the Executables to Detect Malwares with High Accuracy," Procedia Computer Science, vol. 78, no. June, pp. 667–674, 2016.

[9] Sukriti Bhattacharya, Héctor D. Menéndez, Earl T. Barr, and David Clark. Itect: Scalable information theoretic similarity for malware detection. CoRR, abs/1609.02404, 2016

[10] Jake Drew, Michael Hahsler, and Tyler Moore. Polymorphic malware detection us-ing sequence classification methods and ensembles. EURASIP Journal on Information Security, 2017(1):2, Jan 2017

[11] Nak-Hyun Kim, Byung ik Kim, and Tae jin Lee. Performance analysis of the malware classification method in accordance with the changes in assembly code. 2016