MALWARE DETECTION IN COMPUTER SYSTEMS

Avadhesh Gaur
Dept. of Computer Science and Engg.
Indian Institute of Technology, Ropar
Rupnagar, India
2021aim1004@iitrpr.ac.in

Ishan Tripathi
Dept. of Computer Science and Engg.
Indian Institute of Technology, Ropar
Rupnagar, India
2021aim1009@iitrpr.ac.in

I. Motivation

The foundation for this paper was laid long before we began to discuss and implement it. The underlying motto was to implement the machine learning techniques discussed and learned in our course on real-world datasets and solve the real-world problems that are faced by the leading data-centric organizations of the present day.

II. Introduction

Malware is a contraction for Malicious Software. It is an intrusive software that is designed to damage and/or destroy computers and computer systems. A Malware consists of several categories, including computer viruses, ransomware, computer worms, keyloggers, Trojan horses, spywares, etc. The malware industry continues to be a well-organized, well-well-funded market dedicated to evading traditional security measures. Once a computer is infected by malware, criminals can hurt consumers and enterprises in many ways as it can access confidential information or both businesses and personal systems.

Microsoft has over one billion enterprise and consumer customers. Therefore the tech-companies, like Microsoft, are taking this concern very seriously and are deeply invested in improving the security of their systems.

III. Problem Statement

In this term project, we aim to develop various Machine Learning, models using different techniques, to predict if a computer system will soon be hit with malware. The dataset used to achieve this goal is an unprecedented telemetry dataset provided by Microsoft to encourage open-source progress on effective techniques for predicting malware occurrences.

The key point to note is that the objective is to help protect more than one billion machines from damage before it happens.

IV. Literature Review

Malware detection is a long running topic of research and discussion. However, both the quality and the quantity of work done in this area has increased exponentially in the last decade. This shift is primarily due to the increasing focus on use, and “misuse”, of data by individuals and even organisations for their own benefit.

The entire task of Malware analysis can be performed based on three broad categories – based on objective, based on features, and based on the machine learning algorithm used. The research work done in each of the above category can be further divided into several classes of research. This breakdown of the work done in the field of Malware analysis is represented in *figure 1*.

The economics of malware analysis discussed in [1] provided a great starting point as it shed light on the way machine learning has been used so far in the context of malware analysis in Windows environments, i.e. for the analysis of Portable Executables (PE). A decade ago when the mobile technology was rising day-by-day without any concern of malware attacks, a similar paper [2] presented a machine learning based system for the detection of malware on Android devices.

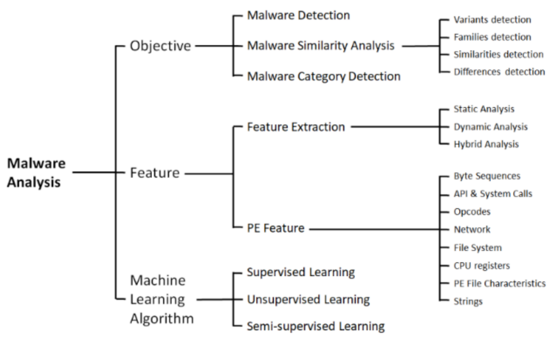

Figure 1: Malware Analysis Techniques

V. Preliminaries

The task at hand is a supervised learning problem of performing binary classification of the telemetry dataset made available by Microsoft. Binary classification can basically be explained as task of classifying the elements of a set into two groups based on the set of classification rules. Some major observations made while performing target label distribution and feature data type distribution are given in the figures 2 and 3, respectively.

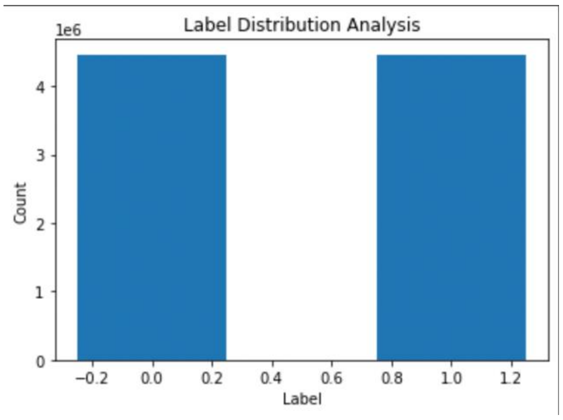

Figure 2: Target Label Distribution

As the dataset is quite exhaustive (~4.38 GB) and consists of 82 features (columns) and 1 binary target class label (0 or 1), it is critical to remove the unnecessary data instances from the dataset before passing it to the machine learning model for learning. Therefore, the data pre-processing steps include removal of features having significantly high ratio of missing values and impractically high cardinality. Later, we also consider finding the correlation between each feature and the target label to eliminate the set of features which have negligible correlation with the target class. Here, the term cardinality refers to the number of unique items (or values) in the feature column. Also, correlation can be interpreted as a statistical term describing the degree to which two variables move in coordination with one another. This relationship can be both, causal and non-causal.

We also present the feature importance of the model trained using the above filtered features and make inferences that can potentially help future researchers building on our work. Feature importance can be inferred as a technique used to calculate a score for each input feature passed in the machine learning model.

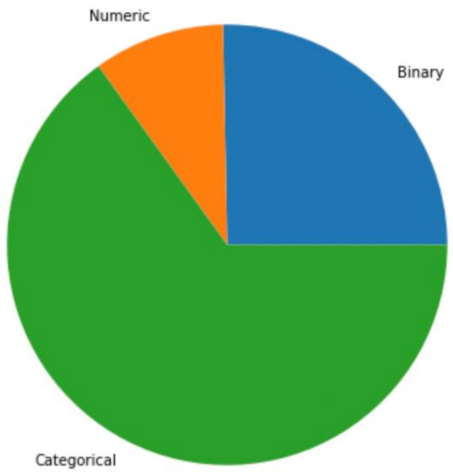

Figure 3: Feature Data Type Distribution

VI. Proposed Model

When performing the pruning of features based on the missing value percentage, we decide to keep the threshold as 40%, i.e. only a feature having less than 40% missing values will be kept in the input dataset. The features such as *PuaMode* (Potentially Unwanted Application Mode), *Census_ProcessorClass, Census_InternalBatteryType, etc.* were pruned because of this step.

Similarly, while analyzing the cardinality of each feature column, we have kept the threshold as 500, i.e. any feature having a cardinality of over 500 is considered impractical and is pruned from the input dataset. This is intuitive and makes sense as, for performing a binary classification task, a feature having a high cardinality will serve more as noise rather than imparting any learning to the dataset. This step resulted in the pruning of the features such as *MachineIdentifier*, *CityIdentifier*, *Census_TotalPhysicalRAM*, etc.

Finally the set of remaining features were each used to find a correlation coefficient which represented the strength of the relationship between the respective feature and the target class label. This step filtered out the features such as AppVersion, Census_DeviceFamily, etc. which were found to have negligible correlation with the target class label.

VII. Setup

The clean data obtained after performing the above data wrangling is split into three parts – training, validation, and test, having the ratio of 60%, 20% and 20% of the total input data, respectively. Also, a sample of 5% of the stratified input data is used where necessary to overcome RAM limitations and save runtime.

A random classifier is built to function as a measure of the accuracy obtained without employing any machine learning technique for malware detection. Then, we build and train a Decision Tree Classifier, a Random Forest classifier, an Artificial Neural Network (ANN) based classifier, and a Light Gradient Boosting Machine classifier and evaluate the performance and runtime of each classifier with each other. The metric used for measuring the performance is the classification accuracy on the validation and the test dataset.

VIII. Results and Discussions

I) Random Classifier

The Random Classifier uses a random function to classify each training data instance. Since malware detection is a binary classification task, the random classifier gives approx. 50% accuracy on each of the training, validation, and test dataset.

II) Decision Tree Classifier

A Decision Tree Classifier creates a decision tree where each node represents a test on an attribute (feature) and each branch descending from the respective node represents one of each feasible values of that attribute (feature). The Decision Tree Classifier used for malware detection on the Microsoft dataset makes use of the 'entropy' criterion and is limited to a max depth of 100. The accuracy achieved by it on the training dataset is over 72 %, whereas the accuracy on the unseen data, i.e. the validation and the test dataset, is approx. 64% each.

The primary reason for the achieved accuracy is that the decision trees are not much influenced by the outliers present in the dataset. Also, the decision trees can inherently handle, both, numerical and categorical variables. However, a small change in the training data instance might result in a significantly different decision tree structure, thereby making the decision tree unstable.

III) Random Forest Classifier

For the implementation of Random Forest Classifier, the parameters used are same as that in the above Decision Tree Classifier. We observe an accuracy of over 77% on the training dataset, whereas an accuracy of over 64% in the validation dataset. The accuracy achieved on the test dataset was found to be approx. 64%. Therefore, a slight increase in the accuracy is observed as compared to the decision tree classifier. However, it takes more time to train as compared to the decision tree classifier.

IV) ANN Based Classifier

Artificial Neural Network (ANN) process the data instances one at a time and learn by comparing their classification of the record with the known actual classification of the record. The ANN Classifier consists of units (neurons), arranged in layers, which convert an input vector into some output. Each unit takes an input, applies a (often nonlinear) function to it and then passes the output on to the next layer. The accuracy obtained on the training dataset was over 69%, whereas the accuracy obtained on the unseen datasets, validation and test, is approx. 65% and 66%, respectively. Therefore, we observe a steady increase in the training accuracy of the ANN based classifier, besides a better performance on the validation and test datasets. Moreover, ANN based classifier took less time to train than the Random Forest Classifier.

V) Light GBM Classifier

The Light Gradient Boosting Machine (LightGBM) classifier was observed to give the best performance compared to all the above classifiers mentioned above. The LightGBM classifier was trained using stratifiedkfold technique, which ensures that each fold of the dataset has the same distribution of the target class label as in the training dataset. It was seen to give an accuracy of over 69% on the training dataset and an accuracy of approx. 68% each on the unseen datasets, i.e. the validation dataset and the test dataset.

VI) Inferences

The feature importance obtained after training the LightGBM classifier is given as:

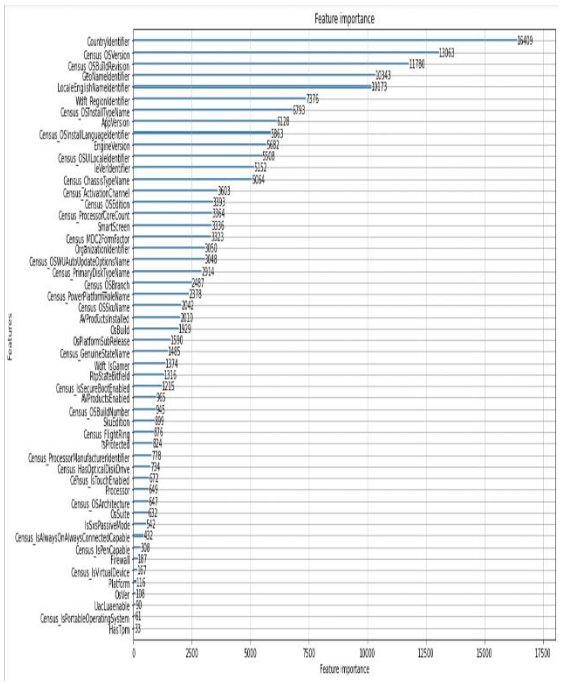

Figure 4: Feature Importance

The observed training accuracy is given in the figure below:

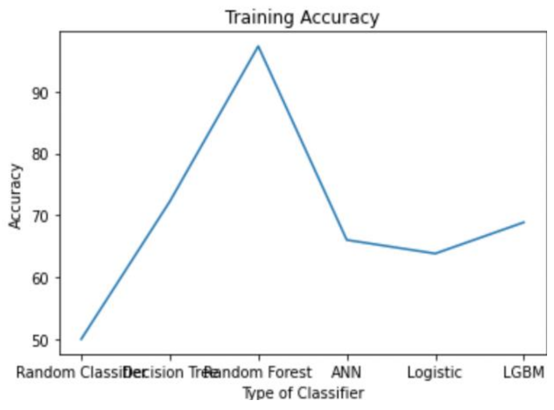

Figure 5: Training accuracy for each classifier

The observed validation accuracy is given in the figure below:

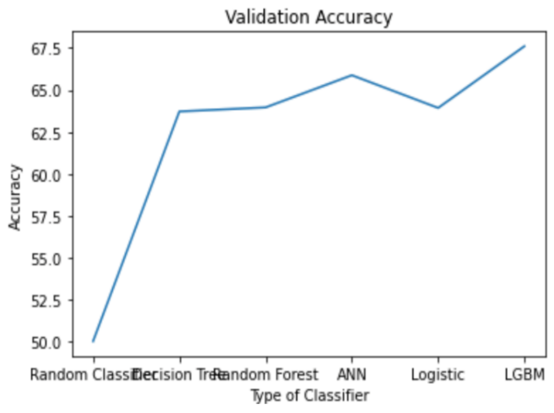

Figure 6: Validation accuracy for each classifier

The observed test accuracy is given in the figure below:

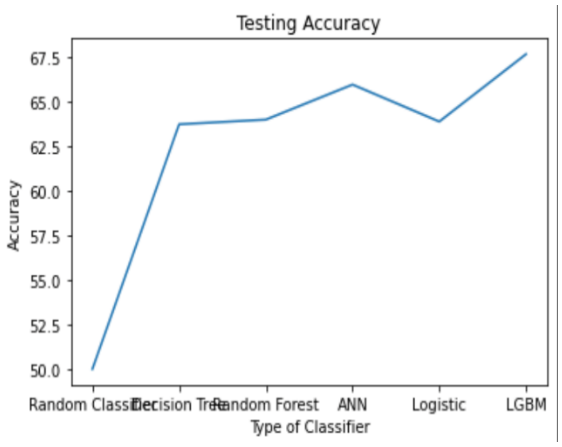

Figure 7: Test accuracy for each classifier

IX Conclusion

Real-world datasets require complex models to be built to give out predictions with utmost accuracy. However, they do not end up being highly accurate on the unseen data. Increasing the complexity even further generally results in losing the ability to explain the model to humans. We use several pre-processing and classification techniques on the given dataset to train a commendable model and found the LightGBM technique to give the highest accuracy on unseen data. LightGBM, originally developed by Microsoft, uses decision-tree based learning algorithms. It has the following advantages, such as faster training speed and higher efficiency, lower memory usage, better accuracy, and capable of handling large-scale data.

X Future Work

This term project report seeks to pave the way for future work along the direction of defining appropriate benchmarks for malware analysis is a priority of the whole research area. This can be achieved by active discussion on malware analysis issues which can provide further ideas worth to be explored. The novel concept of malware analysis economics can encourage further research directions. Appropriate tuning strategies can be provided to balance competing metrics (e.g., accuracy and cost) when designing a malware analysis environment.

XI References

[1] '*Survey of machine learning techniques for malware analysis*' by Daniele Ucci, Leonardo Aniello, Roberto Baldonia; published in 2019

[2] ''*A Machine Learning Approach to Android Malware Detection*' by Justin Sahs and Latifur Khan; published in 2012

[3] '*The rise of machine learning for detection and classification of malware: Research developments, trends and challenges*' by Daniel Gibert, Carles Mateu, Jordi Planes; published in 2020,

[4] '*Survey of Machine Learning Techniques for Malware Analysis*' by Daniele Uccia, Leonardo Aniellob, Roberto Baldonia; published in 2019