MALWARE PREDICTION

Team : The Drifters

Hruthik Chevuri
IIIT
BANGALORE
IMT2018507
hruthik.chevuri@iiitb.org

Anirith Pampati
IIIT
BANGALORE
IMT2018516
anirith.pampati@iiitb.org

Rohan Kumar
IIIT
BANGALORE
IMT2018515
rohan.kumar@iiitb.org

***Abstract*—The malware industry continues to be a well-organized, 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.**

We are trying to predict the malware using the given dataset. We are building a model using traditional machine learning algorithms and preprocessing steps and measures to detect the malware.

***Index Terms*—Pre-processing, Label encoding, One hot encoding, feature Scaling, Regression, Random Forest Classifier, Logistic Regression, SVM, Decision Trees, Regularizaton, Parameter Tuning, Gradient Boosting, XGBoost calssifier, LGBM classifier**

I. Introduction

In this era, there is a lot of advancement in the technology. The number of people using internet, laptops, mobiles and other devices are increasing exponentially. In this time, it is evident that many computers, mobiles and even other electronic gadgets like GPS trackers are affected severely by malware. Malware(Malicious Software) is any application, program or even a file that entered into our system can cause severe damage to our computer. These may steal the data or delete the sensitive information in our system or may be even alter the functioning of the device. There are wide possibilities of causing damage through these malicious softwares. There are many types of malware like Viruses, Worms, spyware etc.

The challenge here is to predict the possibility that the system is affected by the malware given the specifications like Processor, Operating system, Defender securites in the system, geographical location, Census records and various other information related to the system or device.

The report tells the methods and analysis of large amount of information of the system mentioned above.It also tells the methods and traditional machine learning algorithms used for predicting the malware detection using the information. We also included our approach in deciding our final model.And finally with the help of our machine learning model, we are able to predict the possibility of malware in the system upto an extent.

The report will discuss these topics as follows: **Sec-II** discusses the beginning and some related work on the detection on the malware. In **Sec-III** , we describe our dataset and it's features.**Sec-IV** shows our visualizations for understanding our dataset.**Sec-V** describes about the pre-processing steps taken to make our data ready for building our model and also feature scaling. **Sec-VI** describes our model.**Sec-VII** will describe the parameters of XGB and LGBM classifier we have taken and **Sec-VIII** has references and conclusion.

II. Related Work

Malware detection has always been a concern area of research in recent years. Several methods and techniques have been proposed to counter the growing amount and sophistication of malware.

Machine Learning Based Malware Detection. Recently, machine learning methods (e.g., Support Vector Machines (SVM), Decision Trees (DT)) have been used to detect and classify unknown samples for malware family due to its scalability, rapidity, and flexibility. Schultz et al. [1] first proposed to apply the data mining method to detect malware. Santos et al. [2] proposed a hybrid malware detection tool based on machine learning algorithms called OPEM that utilizes a set of features obtained from static and dynamic analysis of malicious code . The above articles have been an inspiration for our approach in this malware prediction problem.

III. Dataset

With more than one billion enterprise and consumer customers, Microsoft takes this problem very seriously and is deeply invested in improving security. The dataset used in

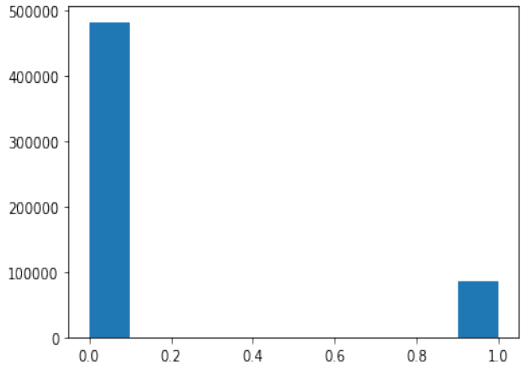
this project is the microsoft malware prediction dataset which is hosted on kaggle .
Each datapoint in the training data consists of 83 coloumns. There are 567730 datapoints in the training data and the primary test data consists of 243313 datapoints for which the label 'Has detections' is not released.

Given all the columns of the data, there are some columns with more than 70 percent of the columns with null values. Also here there are large number of features(83 columns), which consists of columns with different data types like objects, integers.

IV. Observations

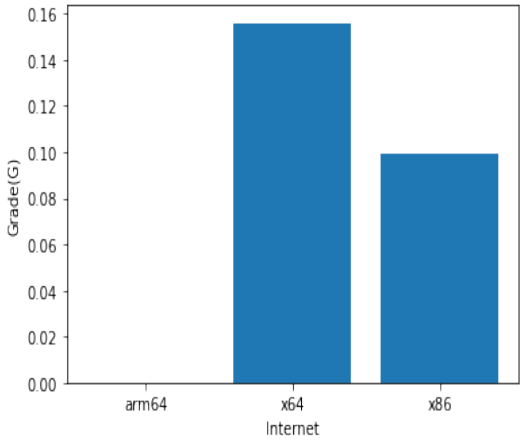
In building any machine learning model, pre-processing is very important to be done. But before doing that, we observe some trends of features in the dataset which helps us in understanding the distribution of the data. So here are few of those observations:-

A. *Distribution of data over Has detections label*



As the problem statement clearly mentions, The training data is an imbalanced dataset.

B. *Malware found on different processor*



We can observe that the probability of malware found on x64 processors is high compared to other processors. We can visualize so many relations like these from the dataset.

V. Data Preprocessing

Data preprocessing is a technique for transforming our collected raw data. This raw data may be skewed, different data types, consists of noise and some values may be even incomplete.The raw data may even consist of numbers in string format. Our aim through preprocessing is to transform this raw data into an understandable format with proper interpretations.

As we know that the efficiency of our machine learning model highly depends on the quality of the data that we give to it for training. Therefore, preprocessing step in training our model plays a vital role in improving our overall model performance.

We have done these preprocessing steps to transform our raw data in a better way. 1) Filling the missing values 2)Dropping some unwanted columns 3)Categorical encoding(Label and One hot encoding) 4) Feature scaling

Filling the missing values: As we know that the collected raw data may consist of many missing values in many different columns. We have filled the missing values for different features using the mean or mode of the column which suits the column better. We have filled the missing values in various features using different parameters like mode, mean or a new label 'missing value'. These fillings are done differently for different columns based on their data type and number of null values in the entire column.

Columns with missing values		
Feature name	missing values (out of 243313 datapoints)	percentage of null
PuaMode	243266	99.9
CensusProcessorClass	242359	99.6
DefaultBrowsersIdentifier	230835	94.8

These columns consists of very large number of null values. They don't provide much statistics for us. Therefore dropping these columns might help.

Categorical encoding : Any structured dataset may contain a mixture of numbers and text. But a machine can understand numbers but not text. So there is a necessity to convert these text into some numerical format. Categorical encoding is a process of converting categories to numbers.

About Label Encoding : Label Encoding is a popular encoding technique for handling categorical variables. In this technique, each label is assigned a unique integer based on alphabetical ordering. But when the label encoding is performed, it inherently gives some ordering of the data. Due to this, there is a very high probability that the model captures the relationship between data such that class1 ¡ class2 ¡ class3 etc. It is better to encode in this format if

there exists some order for the class labels.

About One hot encoding : One-Hot Encoding is another popular technique for treating categorical variables. It simply creates additional features based on the number of unique values in the categorical feature. Every unique value in the category will be added as a feature. If there are n unique values , it will create n additional features. It increases the computation cost but it will not give ordering like label encoding.

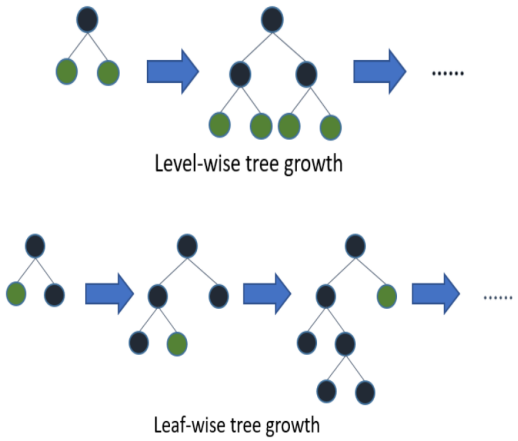
VI. Model Selection

So far we have done preprocessing to our dataset to train our model better. The important task here is to implement suitable machine learning model to the problem which gives better results. Since it is a classification problem, Initially we have tried out classification algorithms like logistic regression , decision trees and also Naive Bayes classifier.These models gave less accuracy around 60%.

We started to approach towards ensemble techniques like bagging, boosting techniques to improve the accuracy. We tried out with a bagging technique , random Forest Classifier: Its is a classification algorithm which consists of large number of individual decision trees that operates as an ensemble.A large number of relatively uncorrelated decision trees operating as a committee will give better performance than the individual decision trees. The important point to remember is that the decision trees are most sensitive to the training data. A slight change in the training data may result in significant change in the tree's structure. This ensemble technique has improved our accuracy beyond 60%.

Bagging can be thought as a committee of experts and boosting can be thought as a group of specialists. Both make the final decision by averaging the N learners but in case of bagging, it is an equally weighted average. In case of boosting, more weight will be to those learners which perform better. We can't say which ensemble technique outperform other technique but it will be based on the data. There are several ways to determine these weights in the boosting ensemble.

We tried those boosting ensemble technique XG-boost classifier and Light-GBM classifier. XGboost is an advanced implementation of gradient boosting algorithm(ensemble that works for both regression and classification problems). It also has regularization which reduces the chance to overfit and improves overall performance. Coming to the LGBM, it is also a gradient boosting ensemble technique but it is useful when the dataset is huge. The difference of LGBM from other gradient boosting algorithms is that it follows leaf-wise approach while other algorithms work in a level-wise approach. Leaf wise growth may overfit thye model. so we need to take care for the max_depth parameter It can be understood better as below:



This picture has been taken from [8] These boosting techniques have significantly improved the model accuracy and also LGBM classifier takes less time to run on our dataset.These have improved our model accuracy beyond 70%.

Then we have came up with a model which is a weighted average for XGBoost classifier and LGBM classifier.Weighted average is averaging both the results but with specific weights assigned to each. Here LGBM classifier works out slightly better result as compared to XGB calssifier. So weight for LGBM classifier is given more compared to the other. we have predicted the result with weights being 0.75 for the LGBM classifier and 0.25 for the XGBoost classifier which gave slightly better performance than the individual classifiers. Through this model, we got the prediction result for the primary dataset more than 71%.

Parameter tuning

We have selected the model which gives better accuracy in predicting the result.But we also need to take care of the parameters for our XGboost and LGBM classifiers.Changing these parameters might even overfit /underfit the model to the dataset. Do utmost care must be taken while tuning the parameters.

For XGBoost classifier: The first step is to set the learning rate. It should be somewhere between 0.05 to 0.30. Generally it will be taken to be 0.1. The next step is to set the tree specific parameters like the 'n_estimators' parameter(no. of trees in the XGB model) which is default set to 100. But we can try changing it to a large number in hundreds or thousands. Increasing it doesn't crossing a certain limit will not improve the model. We have set this to 350. Next the size of the decision tree in the XGBoost can be set. This can be set by 'max_depth'. It's default value is three. we tuned it to 8 as we observed an increase at this point. Here setting the depth to a high value may lead to overfitting. So care must

be taken while tuning this parameter.

Another tree specific parameter is 'colsample_bytree'. All colsample_by* parameters have a range of (0, 1], the default value of 1, and specify the fraction of columns to be subsampled. colsample_bytree is the subsample ratio when constructing the tree.After tuning we have set it to 0.8. Later comes the regularization parameter. As we all know that the regularization term helps the model not to become complex and it is controlled by the coefficient alpha . So in this way we have set our hyper parameters for the XGBoost calssifier through the trail and error method. Finally we have came with the following parameters : "xgb.XGBClassifier(objective="binary:logistic",n_estimators = 350, max_depth = 8, colsample_bytree=0.6,learning_rate = 0.1, reg_alpha = 0.005, random_state=42)".

For LGBM classifier: A similar approach as mentioned above is taken while tuning the parameters for this classifier too. 1) Fixing the learning rate.2) Fixing the tree specific parameters 3)Regularization parameter alpha. So the final parameters set are : "LBMClassifier(objective='binary', reg_alpha=1.0, reg_lambda=1.0,min_child_samples=175, num_leaves=256,learning_rate=0.05, n_estimators=1000,max_depth=10, metrics ='auc')".

VII. CONCLUSION

We would like to conclude that we have tried out the possible ways that we know so far in the course to predict the malware in the system for the given dataset. Projects like these have a scope of improvement through advance machine learning models.

VIII. ACKNOWLEDGEMENT

We would like to our Professors G. Srinivas Raghavan, Neelam Sinha and our Machine Learning Teaching Assistants for helping us whenever we were stuck. And especially pushing the ensemble discussion forward in the course helped us a lot in implementing ensembles in our assignment.They helped us in improving our model accuracy.

Leaderboard positions motivated us in working towards better optimizations everyday. We would like to thank our fellow teams for maintaining healthy competitions in the leaderboard.

REFERENCES

[1] M. G. Schultz, E. Eskin, E. Zadok, and S. J. Stolfo, "Data mining methods for detection of new malicious executables" in Proceedings of the IEEE Symposium on Security and Privacy (S and P), pp. 38–49, May 2001.

[2] Santos et al. [22] proposed a hybrid malware detection tool based on machine learning algorithms called OPEM that utilizes a set of features obtained from static and dynamic analysis of malicious code.

[3] Akhil Kadiyala and Ashok Kumar. "Applications of python to evaluate the performance of decision tree-based boosting algorithms".

[4] ALAKH SETHI, "One-Hot Encoding vs. Label Encoding using Scikit-Learn" 2017, [Online; posted, MARCH 6, 2020,].[Online]. Available: https://www.analyticsvidhya.com/blog/2020/03/one-hot-encoding-vs-label-encoding-using-scikit-learn

[5] Jason Brownlee, "How to Tune the Number and Size of Decision Trees with XGBoost in Python" 2017, [Online; posted,September 7, 2016].[Online]. Available:https://machinelearningmastery.com/tune-number-size-decision-trees-xgboost-python

[6] Scikit-learn documentation , " GradientBoostingClassifier"

[7] Jason Brownlee, "HHow to Configure XGBoost for Imbalanced Classification" 2017, [Online; posted, February 5, 2020].[Online]. Available:https://machinelearningmastery.com/xgboost-for-imbalanced-classification

[8] AARSHAY JAIN, "Complete Guide to Parameter Tuning in XGBoost with codes in Python" 2016, [Online; posted,, MARCH 1, 2016,].[Online]. Available:https://www.analyticsvidhya.com/blog/2016/03/complete-guide-parameter-tuning-xgboost-with-codes-python/

[9] Guolin Ke et al. "Lightgbm: A highly efficient gradient boosting decision tree". In: Advances in Neural Information Processing Systems. 2017, pp. 3146–3154