Malware Detection

Team name - NSA

Ankita Paul
MT2020053
ankita.paul@iiitb.org

Neha Kothari
MT202010
neha.kothari@iiitb.org

Soham Chatterjee
MT2020071
soham.chatterjee@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.**

Index Terms - *Feature Engineering, Logistic Regression, Light Gradient Boosting Machine, SMOTE, Stratification*

I. Introduction

A vulnerability assessment against malware (malicious software) is one of the most important topics in the cyber security domain. Everyone wants to feel safe, right? The problem of malware infection may touch everyone from individual private people, through companies, up to governmental agencies. The malware itself is infectious software (mostly Trojan horses and viruses, but not only) that tries to misuse or cause damage to a computer, server as well as to extract/steal sensitive, confidential or private information. Malware can get into the system by using security defects in the software itself, e.g. in various versions/builds of the operating system, internet browsers or their plugins. Therefore, operating systems and other software need to be updated from time to time in order to improve their performance and security.

The goal is to predict a Windows machine's probability of getting infected by various families of malware, based on different properties of that machine.

The rest of the paper proceeds as follows: In Sec. 2 we describe our dataset and the evaluation criteria, Sec. 3 covers our visualizations, EDA and their inferences Sec. 4 will discuss about Data pre-processing and feature extraction, Sec. 5 will discuss training methods and comparison of our model with the various other approaches.

II. Dataset and Evaluation Metric

A. Dataset

Each row in this dataset corresponds to a machine, uniquely identified by a MachineIdentifier. HasDetections is the ground truth and indicates that Malware was detected on the machine.

We have a huge dataset, where most features are categorical.

There are 567730 data points in the training data and the primary test data consists of 243313 data points for which the labels are not released. There are a total of 83 features in the train dataset having both numerical as well as categorical features.

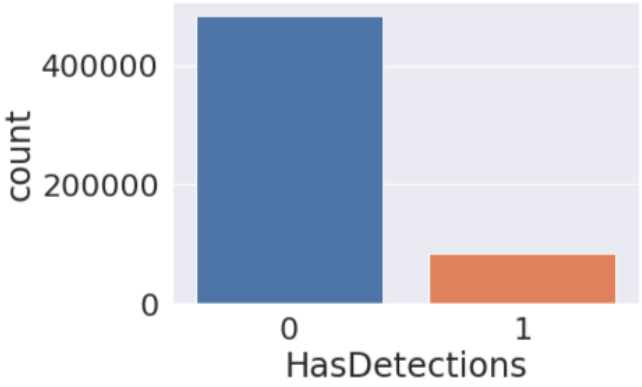
B. Evaluation Metric

Evaluated on the area under the ROC curve between the predicted probability and the observed label. The evaluation cannot be based on "accuracy" metric since the results can be binary and the dataset is unbalanced. So even if we predict "0" for the entire HasDetection, we will still get a high "accuracy" but our results will not be correct.

III. Visualizations and EDA

Before getting into pre-processing and feature extraction, it is very important to get to know the distribution of data in order to get better insights while feature selection. We are presenting a few of those here:

A. Distribution of data over the class labels in the dataset

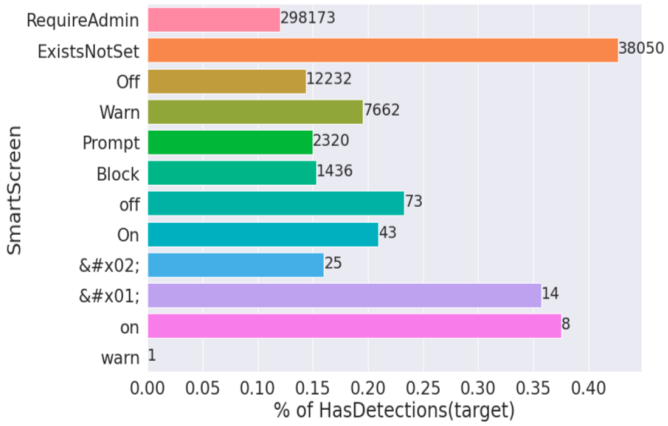


This shows that maximum data we have didn't have any detections of malware.

B. Features with a lot of missing values:

Feature	Unique_values	Percentage of missing values
PuaMode	1	99.984324
Census_ProcessorClass	3	99.614429
DefaultBrowsersIdentifier	562	94.860937
Census_IsFlightingInternal	1	82.738978
Census_InternalBatteryType	29	70.499886
Census_ThresholdOptIn	2	63.118031
Census_IsWIMBootEnabled	1	63.023797

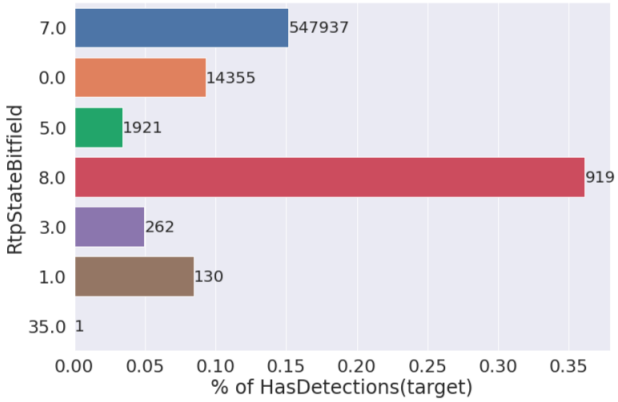
C. *SmartScreen - categories with same meaning but different syntax*



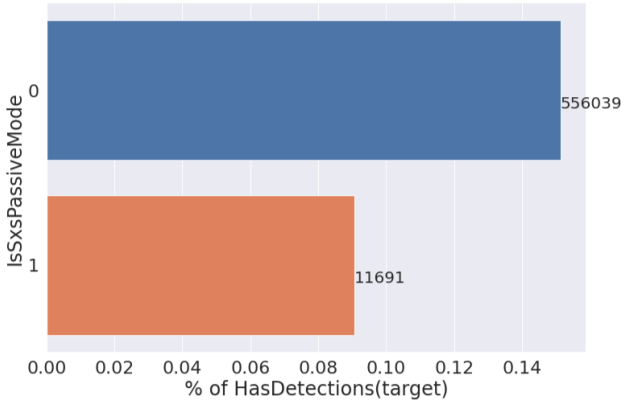
There are a few features which show similar behaviour. Features of same type but different names are present.

D. *Imbalance Features*

Apart from target column, there are several features which are imbalanced. For example:



Here RtpStateBitfield category 7 dominates the feature. Whereas category 8, 3 and 1 are below 1000.



IV. Data Preprocessing and Feature Extraction

To make the training better we need to modify the data, which essentially means that we need to convert the raw data we were given to more sophisticated data which would then be ready to undergo subsequent processing. Basically, we need to extract meaning out of raw data before we can use it to train our models. Models trained on meaningful data always have an upper hand to those trained on raw/less meaningful data. Hence, pre-processing the data is a very crucial step in training our model.

As observed in our EDA, the dataset is highly imbalanced and has a lot of missing values, so before engineering new features, we need to clean the data and the existing features.

First, we dropped the features that had more than 90% missing data. Trying to figure out the missing information is not possible in such cases.

Also there are features which have only 1 unique value. These were dropped because they wouldn't add any extra information. Similarly 'MachineIdentifier' was dropped because it served as a primary key.

Some features had values that meant the same thing, eg. 'SmartScreen' had different categories - ('On','on'),('Off', 'off)',('warn', 'Warn') - words that differ syntactically but have the same semantic meaning. We combined such similar categories of some features.

Several features were imbalanced. So we decided to combine categories with value_count less than a certain threshold value, depending on the distribution of all categories, and created a new category called "OTHER". While doing this, we looked for all the categories that were present in the test set but not in the train set. These completely new categories were also converted to "OTHER".

To handle missing data amongst the numeric features we employed a strategy of mean imputation. So any data which was missing was filled with the mean value of the numeric feature. For categorical features, we encoded the features such that NaN values were automatically assigned a numeric category. It may be possible that certain features were NaN because they had default values which were not recorded.

As shown in the EDA, there were some features where some of the values didn't make sense to us, or some features simply had too many categories in them. Instead of grouping them together and assigning a random value, we decided to have some fixed values to which they will be mapped. For eg: Size of RAM = 4094MB, so we decided to round them off to the nearest value, in this case 4096MB. This avoids the situation where, for example, both 4094MB and 511MB are mapped to a random number like 1 or 0. Instead they are assigned to their closest bucket.

We added new features by extracting information from version or build related features. It's possible that older or newer versions may be more prone to corruption due to outdated security or a new update that makes the system more vulnerable to a breach.

Additional numeric features were created. It is possible to get the aspect ratio from the horizontal and vertical

resolution features. Similarly we can get other display related information from it, like dpi, megapixels of display, etc. from "Census System Volume Total Capacity", "Census Primary Disk Total Capacity", etc., we extracted additional memory related features.

After cleaning up all the features and feature engineering, the remaining categorical values were encoded. Each unique category was assigned a unique index. While label encoding, if any NaN values were encountered (null values which were not previously handled), they were also added to a new category.

V. Training and Results

Malware Detection is a typical Classification problem. The first model we used to fit our dataset was Logistic Regression. However, Logistic Regression does not have sufficient hyper parameters to tune when compared with the Gradient Boosting Models. In machine learning problems, Stratified sampling is much better compared to Random Sampling since the sample obtained after stratification is a good representation of the entire distribution. So, we did cross validation along with Stratification on the gradient boosting models like Xgboost and LightBoost. Since our dataset contains a lot of categorical features, XgBoost did not perform well since it only accepts numerical values similar to Random Forest. LightBoost, on the other hand, can also handle categorical features by taking the input of feature names. It uses a special algorithm to find the split value of categorical features. As an enhancement, we also carried out SMOTE (Synthetic Minority Oversampling Technique), we balanced our dataset and then trained it on a Light Boost in a Stratified 5 fold cross validation manner.

Below is the list of scores which we obtained with different techniques:

Model	Score
Logistic Regression	0.54029
Stratified KFold XGB	0.70243
Stratified KFold LGBM + SMOTE	0.71815
Stratified KFold LGBM	0.71954

VI. Conclusion

Using the stratified Light GBM, our model was able to predict whether the malware was detected by the system or not with an accuracy of 71.954%. Also we could predict which are the more dominant features that help in better prediction thereby increasing the model accuracy.

Acknowledgment

We would like to thank Professor G. Srinivas Raghavan, Prof. Neelam Sinha and our Machine Learning Teaching Assistants, Arjun Verma, Tejas Kotha, Shreyas Gupta, for giving us the opportunity to work on the project and help us whenever we were stuck by giving us ideas and resources to learn from. A special thanks to Meghna Dubey and Nihar Kanwar for having a healthy discussion regarding various approaches in pre-processing, feature selection, algorithm optimizations etc. We had a great learning experience while working on the project and the leader board was a great motivation to work on the assignment. The competition forced us to read up various articles and papers which gave us ideas and enthusiasm for the project.

References

[1] Jovan Sardinha. An introduction to model ensembling. [Online]. Available: https://medium.com/weightsandbiases/an-introduction-tomodel-ensembling-63effc2ca4b3

[2] Philip Hyunsu Cho, Nan Zhu et.al., XGBoost. [Version: 1.2.0]. [Online]. Available: https://xgboost.readthedocs.io/

[3] Jer´ emie du Boisberranger, Joris Van den Bossche et.al., sckit-learn: Open ´ source scientific tools for Machine Learning. [Latest] [Online].

[4] Microsoft Corporation Revision 9597326e. LightGBM. [Latest]. [Online]. Available: https://lightgbm.readthedocs.io/

[5] Microsoft Malware detection: https://medium.com/@gabriel.pirjolescu/microsoft-malware-prediction-data-exploration-e262a439662c

[6] SMOTE(Synthetic Minority Oversampling Technique) https://rikunert.com/SMOTE_explained

[7] Scikit learn Stratified K-fold documentation https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.StratifiedKFold.html

[8] Stratified Sampling https://medium.com/@dhivyarao94/stratified-sampling-machine-learning-b622189ae77

[9] Light GBM vs XgBoost

https://towardsdatascience.com/catboost-vs-light-gbm-vs-xgboost-5f93620723db#:~:text=Unlike%20CatBoost%20or%20LGBM%2C%20XGBoost,supplying%20categorical%20data%20to%20XGBoost.