Malware Prediction

Team: ML_Noobs

Anish Rajan
IMT2018009
IIIT Bangalore
Bangalore, India
anish.rajan@iiitb.org

Gaurav Goel
IMT2018025
IIIT Bangalore
Bangalore, India
gaurav.goel@iiitb.org

Ishmeet Singh Saggu
IMT2018030
IIIT Bangalore
Bangalore, India
ishmeet.singh@iiitb.org

***Abstract*—The malware industry today is a huge industry by itself wherein there are experts who are very skilled in evading the latest technological advancements in security. There are a lot of consumers whose businesses are solely dependent upon machines and a lot of them are prone to malware attacks.**

In this paper we try to analyze this problem based on the Microsoft Malware dataset and present a model based on traditional machine learning techniques to predict such malware attacks before they happen.

***Index Terms*—sklearn, SimpleImputer, LabelEncoder, OrdinalEncoder, MinMaxScaler, LogisticRegression, LinearSVM, XGBoost, LightGBM, RandomForest, SMOTE, train_test_split, ROC Curve**

I. Introduction

In this world of technological advancements, computers play a huge role in businesses and other establishments. Most of the work today, from petty things such as shopping to some important jobs like legal documents exchange and money transfer is done on computers.

With these increasing advancements of digitizing everything, there are concerns about malware attacks. Such attacks to businesses have been proven to be harmful and may result in the collapsement of that business. Banks, if attacked by malwares is fatal not only for the bank but also the million number of people who have their accounts in it. Everything and everyone today rely on computers.

The malware industry today is well funded by itself and is able to evade any advancements in security. With such problems and risks associated with it, there is evidently a growing need to predict such attacks before they happen which we have tried to address in this paper.

The challenge of malware prediction is to effectively identify whenever and wherever attacks would happen. Microsoft has a billion enterprise users and consumers and having workers identify such malware predictions based solely on data is quite impossible. This report presents a quantitative analysis of the predicting such attacks beforehand using traditional machine learning techniques. We have worked on the Microsoft malware dataset which is published by Microsoft and available from a competition hosted on Kaggle. Our model is based on the combination of XGBoost and LightGBM models which has provided us with good results. With the help of data visualizations we were able to effectively encode features and find the correlation between them.

II. Related Work

The problem of malware prediction is that there is no clear distinction between the patterns of a computer going to have a malware attack with that which is safe. It is very hard to do it on the human level. This problem that we are currently working on has been a competition that was already hosted on Kaggle. Also, Microsoft had released this dataset and challenge in 2015 as an open source challenge. There have been very successful advancements in solving these problems using LightGBM models after frequency encoding some of the features [2].

III. DATASET

The dataset used is a part of the Microsoft Malware Prediction dataset which was published by Microsoft and used on a compeition based in Kaggle.

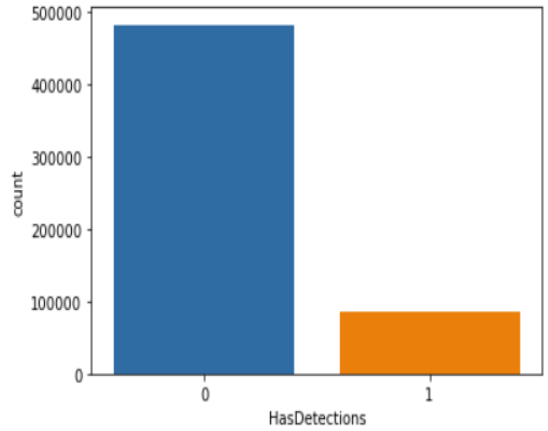
This dataset has 83 columns most of which define the characteristics of the system. There is a column 'HasDetections' which takes two values 0 or 1. 0 denotes that the system was not attacked by a malware and 1 denotes that it was attacked. We need to predict the probability that 'HasDetections' can take the value of 1.

Given the dataset, the problem can be treated as a binary classification problem wherein we have to predict the probabilities of the binary labels.

IV. OBSERVATIONS

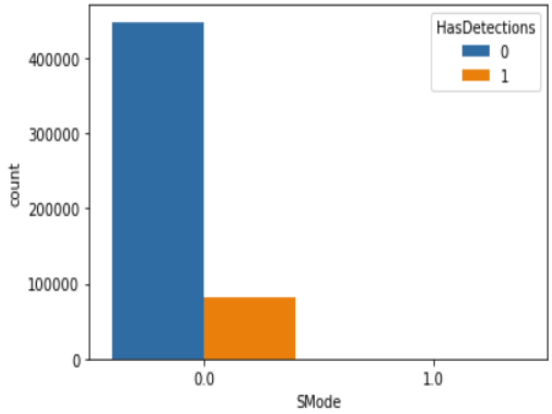
We have visualized our data, before working on it. Visualizing it has provided us with very useful insights.

1) It is important to first see the distribution of different labels in the data.

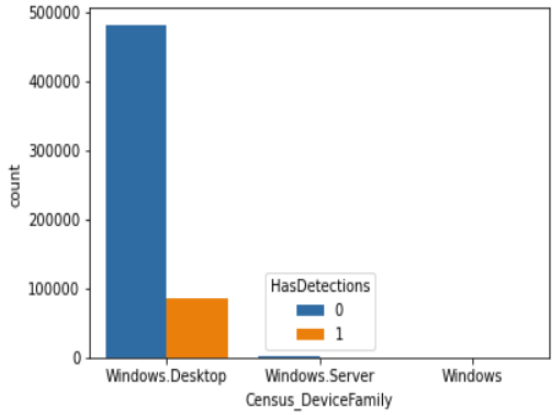


This graph shows that there is an imbalance in the dataset between the classes. The number of datapoints wtherein 'HasDetections' is 0 is overwhelmingly more.

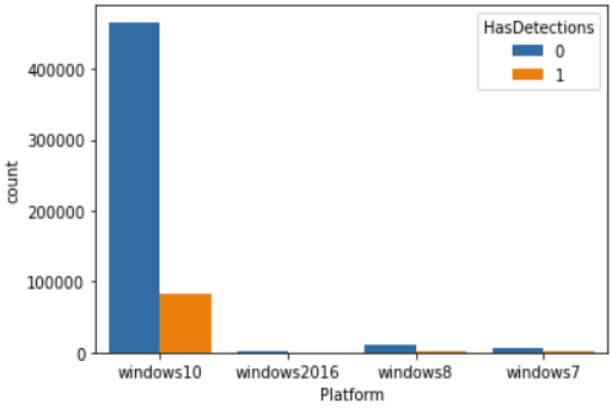
2) Some features don't have much correlation with HasDetections. Hence they have been dropped after seeing their importance in models and plotting their graphs.



99% of the values above belong to class 1, so Smode has been safely dropped.



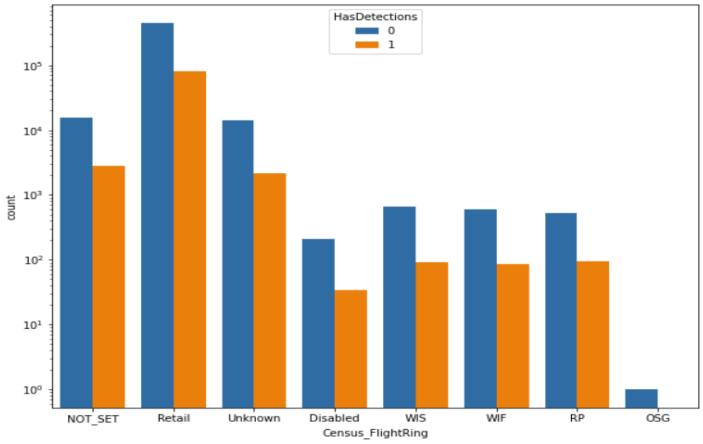
In the above graph most of the classes belong to Windows Desktop and hence this feature has been dropped.

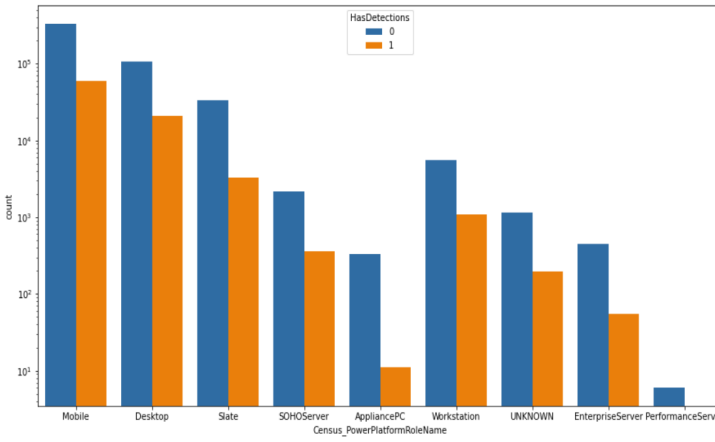


As we can see in the above data most of the Platform values belong to one class(windows10) and hence they do not contribute much to the learning of the models.

In most of the other features which have been used for training the models, not much pattern could be extracted from the data.

3) Some of the features showed very good correlation with the target column





g

V. DATA CLEANING AND PREPROCESSING

To use the data for training we need to first fill in NULL values with appropriate values and change categorical vari-

ables to numerical data. This is essential as training a model requires numerical values.

First of all, some columns were dropped which had a lot of null value percentage as they simply don't provide much meaning in training.

Next we had used SimpleImputer package in sklearn package to impute null values in features. The strategy to impute was to change null values in categorical columns with most_frequent and in numerical features with mean.

Data cleaning was very important as there were some unwanted values in the dataset. For example, some features had categories which were just different in upper and lower case namely 'SmartScreen'. Some features had different categories but with similar meaning. For example, the feature 'Census_PrimaryDiskTypeName' had features called UNKNOWN and Unspecified which essentially mean the same thing so these were changed. Also there were some unwanted numerical values in Census_ChassisTypeName which were then replaced.

We extensively worked with sklearn packages and mainly have used OrdinalEncoder and LabelEncoder on the data. Although, [2] had used frequency encoding, some features in the data had versions and these induce order in the categories. These features are namely, EngineVersion, AppVersion, AvSigVersion, Processor, OsPlatformSubRelease and Census_OSVersion which have been encoded with OrdinalEncoder.

We had also performed feature normalization using MinMaxScaler for some models such as LogisticRegression which helps them train faster.

VI. Model Selection

We had worked with a lot of different models and oversampling methods as there was bias in our data. We had used SMOTE in our data to oversample the less occurring class but this did not seem to work well than normal training. We had used hyperparameter tuning for XGBoost and LightGBM models as soon as we identified that these working best with our data. Finally we did some ensembling with weighted averaging of XGBoost and LightGBM to get some bump in our score.

With individual classifier we had started training with LogisticRegression which gave us a pretty good baseline score to work upon. Then we used LinearSVM to train, but it took a huge amount of time and we could not get any results due to the 9 hour limit on Kaggle notebooks.

The tree based models worked pretty well on the data. RandomForests, XGBoost, LightGBM outperformed all linear models and gave us the best scores. Finally the ensembling of XGBoost, LightGBM gave us our final score on the leaderboard. We wanted to experiment a little more with RandomForest as we could not do it's hyperparameter tuning properly due to the time constraints of the competition and could not include in the ensemble of XGBoost and LightGBM.

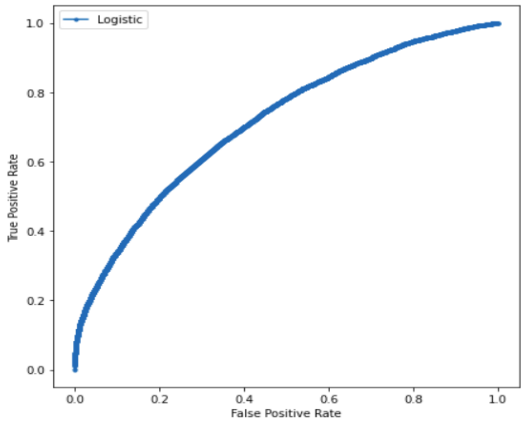
All of the above models were trained on Kaggle Notebooks given the limitation of computation power on our own laptops.

TABLE I
RESULTS OF VARIOUS MODELS

Model	Accuracy	ROC-AUC Score on Train Set	ROC-AUC Score on Test Set
LogisticRegression	0.8498	0.66783	0.66377
SGDClassifier	0.8498	0.661440	0.65636
RandomForest	0.86284	0.907384	0.70269
XGBoost	0.86470	0.814936	0.71400
LightGBM	0.86298	0.80489	0.71357
LightGBM and XGBoost	0.86390	0.814805	0.71584

Here the accuracy is on the train set and the other metrics are calculated on the 90% train set and 10% test set which we have splitted beforehand

ROC CURVE ON TEST SET



VII. Traning Method

In all the models we have first split the training data into 2 parts. 90% of the training data is used for training the model and 10% is used for testing it. This is done by using sklearn's train_test_split. The classes have been equally divided into the train and test set for maintaning consistency.

Earlier we were using Kfold for cross-validation but as the complexity and training time of the models increased, Kfold took a lot of time so we switched to train_test_split. The 10% of the training data was an important criteria to judge our model based on previous models' preformances.

VIII. Conclusion

We were able to build an efficient model for malware prediction given the details of a system. This project has huge application in real life and it is very important to tackle such issues.

ACKNOWLEDGMENT

We would like the thank Professor G. Srinivas Raghavan and Professor Neelam Sinha and all the machine learning TA's especially Tejas Kotha and Nikhil Sai for guiding us along this assignment and hosting this competition. We have learnt different traditional machine learning techniques that are used to solve problems in real life.

IX. PROJECT LINK

The code and report can be viewed here

REFERENCES

[1] 2nd place winner in original contest
[2] Hyperparameter Tuning for XGBoost and LightGBM Models