Malware Detection

Divyanshu Sharma
MT2020061
International Institute of Information Technology
Bangalore
divyanshu.sharma@iiitb.org

Mitisha Agrawal
MT2020123
International Institute of Information Technology
Bangalore
mitisha.agrawal@iiitb.org

Neeraj Jetha
MT2020079
International Institute of Information Technology
Bangalore
neeraj.jetha@iiitb.org

***Abstract*—In this contemporary, technological age, the internet has been embraced by almost everyone. And with it, the danger of malicious attacks have increased. These attacks are done via Malware, and have resulted in billions of dollars of financial damage.The malware industry is a market which is organised and funded to circumvent existing security measures. Once a computer is infected by malware, cyber criminals can impair consumers and enterprises using different methods.This makes the prevention of malicious attacks a necessity.**

We are presenting a model using traditional machine learning methods to predict the probability if a Windows machine will get infected by various families of malware, based on different properties of that machine.

***Index Terms*—Logistic Regression, Random Forest Classifier, Light Gradient Boosting Method(LGBM), AdaBoost, XGBoost, CatBoost, Ensembling.**

I. Introduction

In today's world which is derived from the internet economy most businesses and individuals are dependent on the information systems and computer networks to collect,execute and store digital content. Every business is moving towards the digital forms of working. Companies like Facebook, Netflix are good examples of such businesses. As a result, cyber threats pose a significant challenge when technology infrastructure is compromised by malicious attacks and many of these attacks are through malware.

Malware, or malicious software, is software created to infect a machine without the user's knowledge or consent. It is actually a generic definition for all sorts of threats that can affect a computer. The objectives of a malware could include accessing private networks, stealing sensitive data, taking over computer systems to make use of its resources, or disrupting computing or communication operations. Malware is deliberately evil, even when camouflaged as genuine software from a spread itself within the network, remains undetectable, causes changes or damage to the infected system or network.[1]

Based on the AVTest 2020 results there are 1122.90 m total malware detected so far and this number has been increasing from 2011 itself where the total malware detected was 65.26 m[2].

The challenge is even more because the malware industry is a large, well-funded and well-organized market. They invest heavily in technologies and capabilities to evade traditional protection, requiring anti-malware vendors to develop counter-mechanisms for finding and deactivating them. In the meantime, they inflict significant financial loss to users of computer systems.

One of the major provocations in front of anti-malware software building companies is the vast amounts of data which needs to be evaluated for potential malicious intent. For example, Microsoft's real-time anti-malware detection products execute on over 600M computers worldwide[3]. This generates tens of millions of daily data points to be analyzed as potential malware. The reason for these elevated volumes of files is that, in order to protect the malware from being caught the malware authors institute polymorphism to the malevolent components as a result the malwares belong to same family,with same behaviour but they appear different and are difficult to track.

This report presents various models using traditional machine learning methods to predict the probability if a Windows machine will get infected by various families of malware, based on different properties of that machine.

The rest of the paper proceeds as follows: **Sec 2** Discusses related work on the models and techniques that have been used to predict the probability of malware infection. **Sec 3** We describe our dataset. **Sec 4** Covers our visualizations,EDA and their inferences. **Sec 5** will discuss data preprocessing. **Sec 6** will cover the model-building, their implementation and evaluation. **Sec 7** will detail the comparison of all models applied with the best possible solution. **Sec 8** we include training details that we used. We conclude in **Sec 9** and in **Sec 10** we outline challenges with our method and possible avenues of future work.

II. Related Work

Shabtai Et Al(2009) provide a taxonomy for malware detection using machine learning algorithms by reporting some feature types and feature selection techniques used in the literature. They mainly focus on the feature selection techniques (Gain ratio, Fisher score, document frequency, and hierarchical feature selection) and classification algorithms (Artificial Neural Networks, Bayesian Networks, Naïve Bayes, K-Nearest Neighbor, etc). In addition, they review how ensemble algorithms can be used to combine a set of classifiers.
Bazrafshan et al. (2013) identify three main methods for detecting malicious software:

- signature-based methods.
- heuristic-based methods
- behavior-based method.

In addition, they investigate some features for malware detection and discuss concealment techniques used by malware to evade detection. Nonetheless, the aforementioned research does not consider either dynamic or hybrid approaches[4].

Razak et al. (2016) provide a bibliometric analysis of malware. It analyzes the publications by country, institution or authors related to malware. Nonetheless, the paper does not provide a description of the features employed by malware detectors and does not consider the state-of-the-art in the field.

Ye et al. (2017) cover traditional machine learning approaches for malware detection, that consists of feature extraction, feature selection and classification steps. However, important features such as the entropy or structural entropy of a file, and some dynamic features such as network activity, opcode and API traces, are missing. In addition, deep learning methods or multimodal approaches for malware detection, which have been hot topics for the last few years, are not covered.

Souri et al. (2018) present a survey of malware detection approaches divided into two categories:

- signature-based methods.
- behavior-based method.

However, the survey does not provide either a review of the most recent deep learning approaches or a taxonomy of the types of features used in data mining techniques for malware detection and classification.

Lastly, Ucci et al. (2019) categorize methods based on:
(i) what is the target task they try to solve
(ii) what are the feature types extracted from Portable Executable files (PEs)
(iii) what machine learning algorithms they use.
Although the survey provides a complete description of the feature taxonomy, it does not outline new research trends, especially deep learning and multimodal approaches[5].

III. Dataset

Train Dataset consists of 567730 rows and 83 columns. Test Datset consists of 243313 rows and 82 columns. Each row in the dataset corresponds to a machine, uniquely identified by a MachineIdentifier. HasDetections is the ground truth and indicates that Malware was detected on the machine.The telemetry data containing these properties and the machine infections was generated by combining heartbeat and threat reports collected by Microsoft's endpoint protection solution, Windows Defender.
Using the information and labels in train Dataset, We have to predict the value for HasDetections for each data point in test Dataset.

IV. Observation

Before getting into preprocessing and feature extraction, it is very important to get to know the distribution of data in order to get better insights for feature selection.

- The dataset is imbalanced, having 482571 non infected and 85159 infected rows.

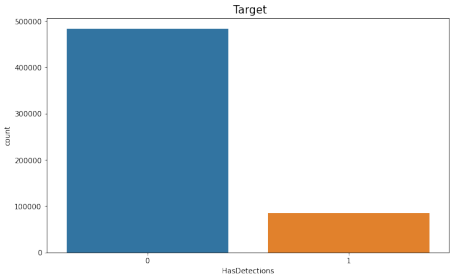

Fig. 1. HasDetection Feature.

- The Dataset have 56 categorical, 19 binary and 8 numerical features. Some features have null values indicated by -1.Fig.2

```
print(len(categorical_columns))
print(len(numerical_columns))
print(len(binary_columns))
```

```
29
39
14
```

Fig. 2. Feature Category

- There are null values present in various features which should be removed first. PuaMode, Censor_Processorclass,DefaultBrowsersIdentifier, Census_IsFlightingInternal, Census_InternalBatteryType, Census_ThresholdOptIn,Census_IsWIMBootEnabled have more number of rows with null values , Fig.3
- Their are many features which are highly correlated. Correlated features do not improve the model and removing them from the model will assist in faster learning

	column_name	percent_missing
PuaMode	PuaMode	99.982738
Census_ProcessorClass	Census_ProcessorClass	99.624469
DefaultBrowsersIdentifier	DefaultBrowsersIdentifier	94.806158
Census_IsFlightingInternal	Census_IsFlightingInternal	82.783189
Census_InternalBatteryType	Census_InternalBatteryType	70.497243
Census_ThresholdOptIn	Census_ThresholdOptIn	63.114156
Census_IsWIMBootEnabled	Census_IsWIMBootEnabled	63.021859

Fig. 3. High NULL Rate Features

of algorithm and in decreasing the bias and over-fitting Fig.4.

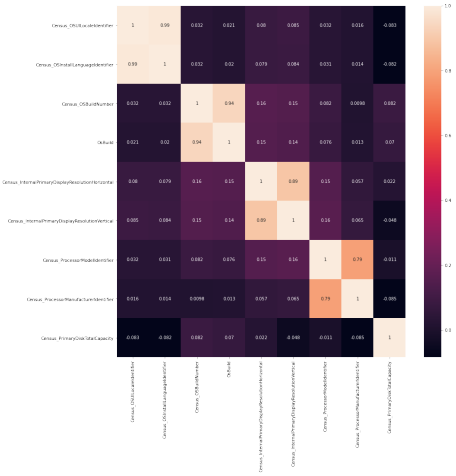

Fig. 4. Features with High correlation

- Graph shows Defenders which are infected. Scep, fep, Windowsintune and mseprerelease are not infected whereas win8defender and mse are highly infected.Fig.5

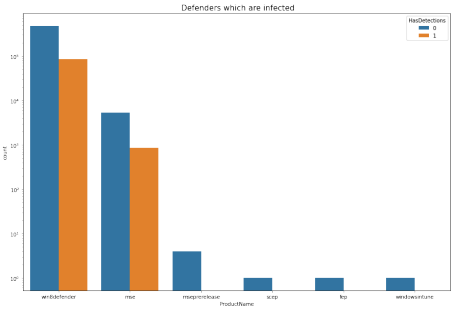

Fig. 5. Defender

- Given graph shows the antivirus installed v/s the count of malware spread. Here we conclude that newest release that is 5.0 is much better and has less infection.Fig.6
- Given graph depicts operating system on which the malware attacks. From all the bars we can see that Windows2016 has less malware detection as compared to other Windows operating system.Fig.7
- On Analysing, Firewall v/s Malware count of device.We found that, Firewall 0.0 works good as compared to

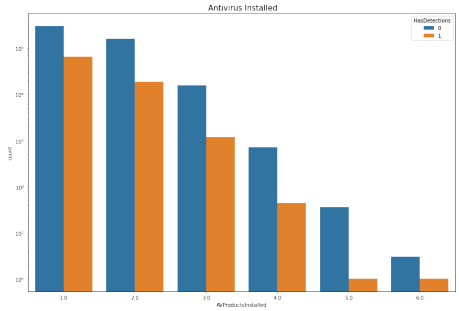

Fig. 6. Antivirus

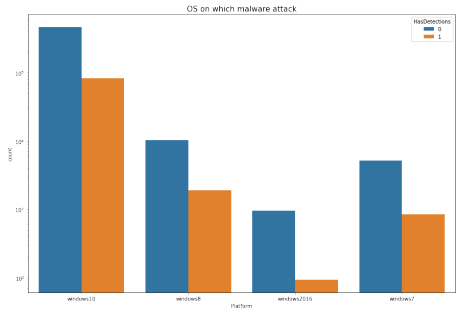

Fig. 7. OS Platform

firewall 1.0 as we have less proportion of infection in 0.0 with respect to 1.0 firewall.

V. DataPreprocessing

Data preprocessing is a process of preparing the raw data and making it suitable for a machine learning model. It is the first and crucial step while creating a machine learning model.

While going through the dataset, we observed that data was not clean and formatted. There was need for cleaning the data and making it suitable for a model fitting which will also help in increasing the accuracy and efficiency of a machine learning model. Hence, preprocessing the data was a very crucial step in training our model.

The dataset contained many features which had NULL values.The features which had more than 50% NULL values did not contribute much to the target feature. We observed that there were 7 features which had more than 50% NULL values, so we removed these features from train as well as test dataset.

The features which had less than 50% NULL values for all of them, we replaced their NULL values with mode values both in train and test dataset.

Additionally, we removed those features which were imbalanced. The threshold we considered for finding the imbalanced features was 0.9. We found out that there were 21 features which crossed the threshold value so we removed those features from the train as well as test dataset.

We divided the features into three parts which were numerical feature , categorical feature and binary feature. Categorical

TABLE I
TOP 5 FEATURE WITH HIGH NULL VALUES

Feature	*Null rate*
PuaMode	0.99
Census_ProcessorClass	0.99
DefaultBrowsersIdentifier	0.94
Census_IsFlightingInternal	0.82
Census_InternalBatteryType	0.70

TABLE II
TOP 5 FEATURE WITH HIGH UNBALANCED VALUES

Feature	*Unbalanced rate*
AutoSampleOptIn	0.99
IsBeta	0.99
UacLuaenable	0.99
Census_IsVirtualDevice	0.98
Census_IsFlightsDisabled	0.98

TABLE III
TOP 5 PAIR OF FEATURES WITH HIGHEST CORRELATION

Feature 1	*Feature 2*	*coeff*
OSUILocaleIdentifier	OSInstallLanguageIdentifier	0.99
OSBuildNumber	OsBuild	0.94
ResolutionHorizontal	ResolutionVertical	0.89
ProcessorModelIdentifier	ManufacturerIdentifier	0.80
PrimaryDiskTotalCapacity	PrimaryDiskTypeName	0.77

feature contained those features which had dtype=object , binary features were those which had only 2 unique values and remaining features we categorised as numerical features. The existence of more than 29 categorical features was a major challenge in this research. It is very common to see categorical features in a data set. However, machine learning algorithm can only read numerical values. It was essential to encode categorical features into numerical values.We used Label Encoding and Frequency Encoding to encode categorical features in this research. Filling missing values is essential before encoding, therefore,we imputed missing values by the most frequent values then, we performed combination of Label Encoding and Frequency encoding while working on the models .

The dataset contained some redundant and highly correlated features. Some of them were almost identical,we identified such columns and removed them from train and then later from test datasets.

Correlation analysis measures the statistical relationship between two different features. Multicollinearity detection must be executed before building the model and deriving variable relationships. In fact, correlated features do not improve the model and removing them from the model made learning of algorithm faster and decreased the bias and over-fitting. Though correlation analysis helped in understanding the association between features in the dataset, it could not explain, or measure, the cause. To achieve the correlations for categorical columns, the first step we performed was encoding and then building a correlation matrix.

VI. MODEL SELECTION

After complete data processing we started applying models on the processed data. We executed some classical machine learning models like Logistic regression, AdaBoost, Random Forest Classifier, XGBoost, LGBM, CatBoost and Ensembling of these models. Since our dataset was unbalanced we tried to balance the dataset using SMOTE (synthetic minority oversampling technique) algorithm but the result/score that we obtained after applying oversampling was not better, than the one we got without oversampling. So we discarded oversampling and applied models on unbalanced dataset.

A. Logistic Regression

Logistic Regression is a Machine Learning classification algorithm that is used to predict the probability of a categorical dependent variable. In logistic regression, the dependent variable is a binary variable that contains data coded as 1 (malware infected) or 0 (not infected). In other words, the logistic regression model predicts P(Y=1) as a function of X[6].

Implementation and Evaluation:The ROC/AUC[7] score that we obtained is 0.5010 which is lowest among all the models.We tried a few combinations of parameters but the score did not increase beyond 0.5010.

B. Random Forest

Random forest is a supervised learning algorithm. The ”forest” it builds, is an ensemble of decision trees, usually trained with the “bagging” method. The general idea of the bagging method is that a combination of learning models increases the overall result. Random forest builds multiple decision trees and merges them together to get a more accurate and stable prediction.

Implementation and Evaluation:While applying the random forest classifier we tried on different sets of the hyperparameters to get better results.We got better results than logistic regression but we could do limited changes in the hyperparameters as when we tried increasing the n_estimators the model became slower and slower, so, The best ROC/AUC value that we could get from random forest classifier is

0.68286 and the accuracy we obtained is 0.8496.The time required to train the data was nearly 3hrs.

C. AdaBoost

AdaBoost helps us to combine multiple "weak classifiers" into a single "strong classifier" .The weak learners in AdaBoost are decision trees with a single split, called decision stumps. AdaBoost works by putting more weight on difficult to classify instances and less on those already handled well. AdaBoost algorithms can be used for both classification and regression problem.

Implementation and Evaluation:We splitted the train data into train and validation in the ratio of 80/20 while creating the model. The result for the AdaBoost algorithm on validation dataset under ROC/AUC score is 0.60780 and the accuracy of the model is 0.8427. We used the base as DecisionTreeClassifier while working on Adaboost. The time required to completely train the data is nearly 6 hrs.

D. XGBoost

XGBoost stands for Extreme Gradient Boosting Algorithm .The XGBoost algorithm was implemented to maximize the efficiency of compute time and memory resources. The algorithm features Sparse Aware implementation by automatically handling missing values from data sets. Additionally, the algorithm allows for continued training of an already fitted model on new data.

Implementation and Evaluation:We splitted the Train data into train and validation in the ratio of 70/30 while creating the model.We tried different combinations of hyperparameters while working with Xgboost and we got better results than the other models we tried so far.The result for the XGBoost algorithm on validation dataset under ROC/AUC metric curve is 0.71238 , accuracy of the model is 0.8582 . The total time required to train the dataset is nearly 1.2hrs.

E. Light Gradient Boosting Method (LGBM)

Gradient boosting utilizes tree-based learning and its base classifier is based on decision trees. Unlike most traditional boosting methods whereby decision trees are grown breadth-wise, LGBM grows leaf-wise. Growing leaf-wise usually results in a lower loss, but it may cause overfitting when data size is small. That is not the case for our project since our dataset has 567730 observations.

LGBM can handle categorical features by just taking the input of the feature names. One of the many advantages of LGBM is that it can handle large size dataset and takes a low memory to run. It also provides a better accuracy as compared to other boosting frameworks.

Implementation and Evaluation:A 5-fold cross validation with a train valid split of 80/20 was used when creating the model. The folds preserved the percentage of samples for each class using stratified kfold. Simply, it helped to ensure that target relative class frequencies was approximately preserved in each train and validation fold.
The result for the LGBM algorithm on validation dataset under ROC/AUC metric curve is 0.7124 .The total time required to train the dataset is 45 minutes.

F. CatBoost

"CatBoost" name comes from two words "Category" and "Boosting". CatBoost is a recently open-sourced machine learning algorithm from Yandex

It is especially powerful in two ways:

- It yields state-of-the-art results without extensive data training typically required by other machine learning methods, and
- Provides powerful out-of-the-box support for the more descriptive data formats that accompany many business problems[9].

Implementation and Evaluation:When applying this model their was no requirement of converting the categorical data to numerical data so we did not perform Label Encoding . The hyperparameters which we tried gave us the best result when we used the iteration as 2000 which we obtained after trying some combinations of parameters.Train data was split into train and validation in the 80/20 while creating the model. The result for the CatBoost algorithm on validation dataset under ROC/AUC metric curve is 0.7238, accuracy of the model is 0.858. The time required to completely train the data is nearly 2.5hrs.

G. Ensembling Models

It is basically Ensembling/combining two or more algorithms could improve or boost your performance.
Ensemble Technique

- Max Voting : In this multiple model used to predict the value of the data points .The predictions by each model are considered as a 'vote'. The predictions which we get from the majority of the models are used as the final prediction
- Averaging: In this method, we take an average of predictions from all the models and use it to make the final prediction.
- Weighted Averaging: All models are assigned different weights defining the importance of each model for prediction.

Implementation and Evaluation:We have used the Weighted Average Technique on the XGBoost and CatBoost model. Train data is split into train and validation in the 80/20 while creating the model. The result for the Ensembling algorithm on validation dataset under ROC/AUC metric curve is 0.712368. We train complete data on both the models and then applied 20% weight of XGBoost and 80% weight of CatBoost for which our Kaggle score came 0.71507. The time required to completely Train the data is nearly 3hrs.

VII. Result Analysis

The AUC score is an indicator of the performance of models, and is the perfect metric to base our observations on. AUC Scores below 0.5 are considered to be bad, and scores between 0.5 and 0.7 are considered to be average. Scores of 0.7 and above are considered to be good scores . We are using it as the defining metric for this assignment, as it is the metric used by Microsoft Corporation in its data competitions pertaining to Malware Prediction.

We applied the classical models Logistic Regression, Random Forest, Adaboost, XGBoost, LGBM, CatBoost and Ensembling with combinations of encoding like frequency encoding and label encoding.But we obtained better results with label encoding.We obtained our best score from Catboost model. CatBoost performed best over all the models which we tried.

TABLE IV
Experiment Results

Experiment Title	*AUC Score*	*Public Score*
Logistic Regression	0.5010	0.52119
Random Forest	0.6828	0.68512
AdaBoost	0.6078	0.61461
XGBoost	0.7123	0.71426
LGBM with K-fold	0.7124	0.71470
Ensembling	0.7123	0.71507
CatBoost	0.7123	0.71553

VIII. Training Details

We trained our model on the 80 percent of the data first and analysed the accuracy and the AUC score, if we saw any improvement in our scores then we trained the model on the complete train data this helped us in saving our time and we could work efficiently. We also used gc.collect() method to call the garbage collector after running the classifier so that unallocated data can be freed. This was useful when we were running multiple classifiers consecutively.

IX. Conclusion

We would like to conclude that we were able to come up with a CatBoost model that gave us the score of 0.71553 while predicting the probability of a Windows machine to get infected by various families of malware.Such assignments have a great scope to protect the machines from malware attacks and ultimately ensure security of the data of the users.

X. Challenges And Future Scope

Identification of insignificant features is a challenge as we need to remove them before we train our model to obtain better results and with a huge dataset its is difficult to analyse the features.For every insignificant feature input to the classifier, the amount of data needed to accurately generalize the classifier increases exponentially and the training dataset density will decrease exponentially. Hence, by having a large number of input variables to the classifier which is noisy and less correlated, we run the risk of over-fitting. This will result in the classifier being affected by noise in the data and cause our classifier to be less accurate.

Long computational time was also a challenge so to overcome these problems we removed the features that we analysed during preprocessing.

With the help of more related data so that patterns between various families of malware could be found out or some sort of common behaviour that can be concluded about the properties of the machine which will be infected by malware, then this could help us to get better results through our models.Moreover using neural networks and pretrained vectors we could achieve better results[8].

XI. Acknowledgment

We would like to thank Professor G. Srinivas Raghavan and our Machine learning Teaching Assistants for giving us the opportunity to work on the assignment and help us whenever we were stuck by giving us ideas and resources to learn from.We would like to thank all the teams on our leaderboard to provide us with healthy competition and give us that desire to work harder and perform better by setting new benchmarks everyday. We would gladly say that we had a great learning experience while working on this assignment.Having a leaderboard was a great driving fuel to work hard and strive for better results everyday by reading more and more articles and implementing various models with lots of combinations.

References

[1] Safir Zawad, Nahian Evan, Raiyan Mansur, Ashub Bin Asad and Muhammad Iqbal Hossain,"Analysis of Malware Prediction Based On Infection Rate Using Machine Learning Techniques", URL:https://www.researchgate.net/publication/342151827

[2] AVTest Malware Statistics, 2020. URL:https://www.av-test.org/en/statistics/malware/

[3] Microsoft. Sam cybersecurity engagement kit, 2018. URL:https://assets.microsoft.com/en-nz/cybersecurity-sam-engagement-kit.pdf.0

[4] URL:https://www.sciencedirect.com/science/article/pii/S1084804519303868

[5] Feature selection and machine learning classification for malware detection.URL:https://www.researchgate.net/publication/283555879

[6] Building A Logistic Regression in Python, Step by Step https://towardsdatascience.com/building-a-logistic-regression-in-python-step-by-step-becd4d56c9c8

[7] Let's learn about AUC ROC Curve! Jocelyn D'Souza https://medium.com/greyatom/lets-learn-about-auc-roc-curve-4a94b4d88152

[8] Microsoft malware detection https://cjango.wordpress.com/portfolio/microsoft-malware-detection-machine-learning-predictive-classifier/

[9] CatBoost: A machine learning library to handle categorical (CAT) data automatically https://www.analyticsvidhya.com/blog/2017/08/catboost-automated-categorical-data/