Malware Prediction Using Machine Learning

Priyal Nile
New York University
NY, USA
pan303@nyu.edu

Hitesh Patel
New York University
NY, USA
hlp276@nyu.edu

ABSTRACT

With the digital world usage reaching the pinnacle in this 21st century, there has been an unprecedented use of the computers and servers. Although the digital world is providing a way more flexibility and comforts of data handling as compared to the traditional paper-based business processes, the digital world also has another counter side of getting infected through malicious software, called malware. The malware can impose security threats by making the internet a dangerous place for an organization. It can lead to severe problems such as data stealing, selling personal information, banking and credit card details, etc. The malware threat could serve to be catastrophic for any small- or large-scale industry. This issue can be tackled using modern machine learning techniques to predict the systems malware occurrence in the future based on various systems configuration taken for feature analysis.

KEYWORDS

Malware, Exploratory Data Analysis (EDA), Machine Learning, XGBoost, Neural Network, LightGBM

1. INTRODUCTION

Cyber-attacks and cybersecurity are major concerns for every organization. It can also be said that Malware & security management is a cornerstone of security in the enterprise. There are high chances of a system connected to the internet getting impacted by the Malware, i.e., malicious software. Malware is a blanket term for viruses, worms, trojans, and other harmful computer programs hackers use to wreak destruction and gain access to sensitive information. The ever-increasing prevalence of malware has led to the explorations of various detection mechanisms. The breach in security can lead to data access and which could be catastrophic for any business. We can only detect a fraction of actual malware infections. To handle these successfully, it is crucial to take a proactive approach before a system being impacted by malicious software. We can use the modern machine learning techniques & probabilistic approach for this. It could be helpful to protect millions of machines, which may be on the verge of getting damaged by malware. The successful prediction of a system being malware infected in the future can assist in the prevention of getting the system malware-infected and can be useful for an organization to run their business in a safe cyber-crime free environment.

In our first approach to handling this problem through machine learning technology, we are using the [11] XGBoost Algorithm in Machine Learning, which is an efficient & high-performance algorithm. XGBoost is also known as the regularized version of GBM since it provides an extra edge in terms of supporting Regularization, Parallel Processing, Handling missing values, cross-validation, and ineffective tree pruning.

2. RELATED WORK

[2] Authors Kehinde Oluwatoyin Babaagba from Edinburgh Napier University& Samuel Olumide Adesanya from Redeemer's University Ede of Nigeria had focused on the effect of feature selection in their machine learning model in the paper '*A Study on the Effect of Feature Selection on Malware Analysis using Machine Learning*.' Their model is based upon the MultiLayer Perceptron algorithm & they considered both supervised and unsupervised machine learning approaches for dealing with the malware problem. However, we implemented XGBoost, & tried to implement Neural network and LightGBM Algorithm for dealing with this problem.

[3] As per the paper published by colleagues from the Boston university Hardware Performance Counters Can Detect Malware: Their paper aimed to analyze the effectiveness of using the micro-architectural level information obtained from HPCs to distinguish between

benign ware and malware and evaluate the fidelity of malware detection using HPCs. However, we dealt with predicting the system from getting malware infected in the future.

[4] As per the paper published by colleagues from the Boston university Ensemble Models for Data-driven Prediction of Malware. Infections – Boise state university They first carefully designed domain-based features from both malware and machine-hosts perspectives. Secondly, inspired by epidemiological and information diffusion models, they designed a novel temporal non-linear model for malware spread and detection. They presented ESM, an ensemble-based approach that combines both these methods to construct a more accurate algorithm. Using extensive experiments spanning multiple malware and countries, they had illustrated that ESM could effectively predict malware infection ratios over time (both the actual number and trend) up to 4 times better compared to several baselines on various metrics. Our model was a simpler version of this where we have used XGBoost as a baseline model.

3. PROBLEM DEFINITION AND ALGORITHM

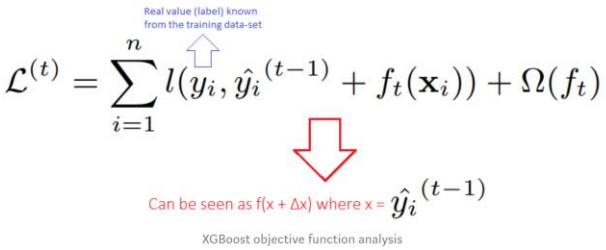
3.1 Task:

The primary goal of the project is to predict a Windows machine's probability of getting infected by various families of malware, based on different properties of that machine. The telemetry data containing these properties and the machine infections were generated by combining heartbeat and threat reports collected by Microsoft's endpoint protection solution, Windows Defender [1]. The following steps would be followed to incorporate this proposed project. Each MachineIdentifier would be a unique identification for a machine. The attribute has_malware_detection confirms whether the malware is detected or not. Using the information in the *train.csv*, we would incorporate the logic for *test.csv*. There would be many attributes that will contribute to the analysis, such as EngineVersion, Product name, Cityidentifier, etc.

3.2 Algorithm:

We incorporated XGBoost as the baseline model for solving the malware detection problem using machine learning. Tree boosting is a highly effective and widely used machine learning method. XGBoost is a scalable end-to-end tree boosting algorithm, which is used widely by data scientists to achieve state-of-the-art results on many machine learning challenges. We propose a novel sparsity-aware algorithm for sparse data and a weighted quantile sketch for approximate tree learning. More importantly, we provide insights on cache access patterns, data compression and sharding to build a scalable tree boosting system. By combining these insights, XGBoost scales beyond billions of examples using far fewer resources than existing systems.

3.3 XGBoost Objective Function

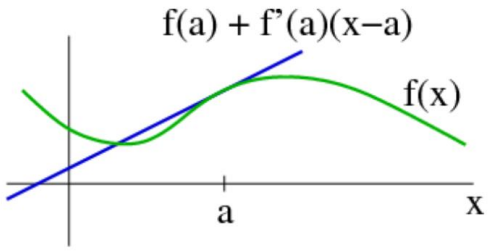


XGBoost objective function analysis

Where, L is a CART Learners, i.e. (Classification and Regression Techniques) [14]

The XGBoost Algorithm internally uses Taylor Approximation. This is essential to transform the original objective function to a function in the Euclidean domain to be able to use traditional optimization techniques.

[15] The best linear approximation for a function *f(x)* at point *a* is:



Taylor linear approximation of a function around a point a

4. EXPERIMENTAL EVALUATION

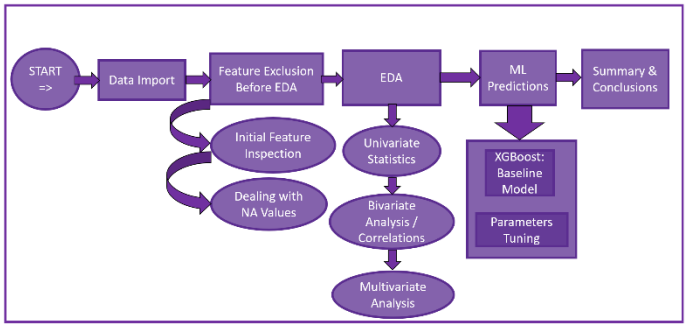
4.1 Data:

Kaggle competition Dataset provided by Microsoft:

Data	DataFile	DataSize	Datapoints
Train Data	Train.csv	4.08 GB	9 Million
Test Data	Test.csv	3.54 GB	8 Million

4.2 Methodology:

FlowChart



We incorporated the above flowchart as an approach for bolstering malware detection using machine learning. The order of the ML predictions is as depicted in the flowchart above. We used the XGBoost model as a baseline ML model for our project. Feature exclusion based on the analysis was carried out before the EDA (Exploratory Data Analysis).

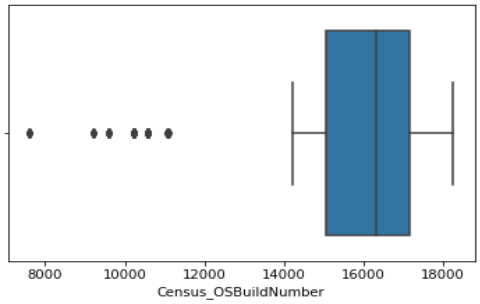
The entire workflow is mentioned in detail below:

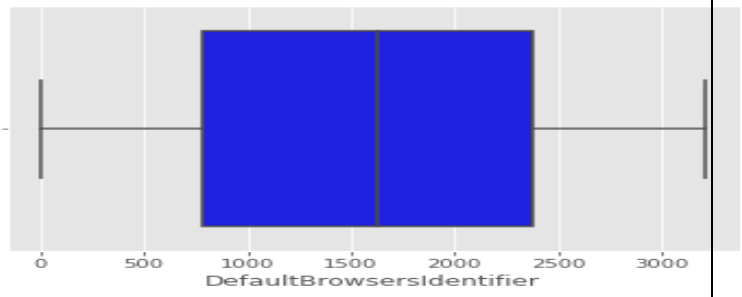
4.2.1 Data Loading:

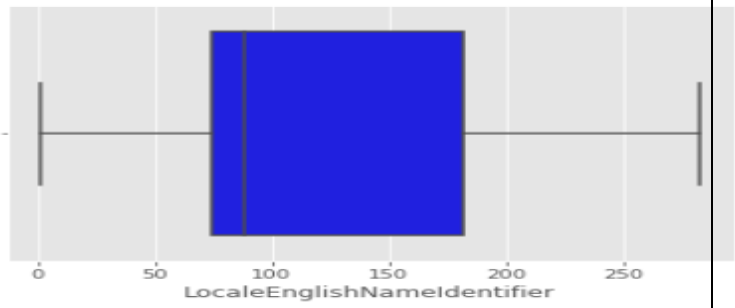
We loaded the train data & test data using the pickle in python to serialize and deserialize data for handling millions of data points

4.2.2 Outlier Analysis:

An outlier is an element of a data set that distinctly stands out from the rest of the data. We have used a box plot to find outliers of those data points that lie outside the overall pattern of distribution.







We can see that Census_OSBuildNumber has few outliers, while DefaultBrowserIdentifier has no outliers and the data lies within the quartile range.

4.2.3 Bi-Variate Analysis:

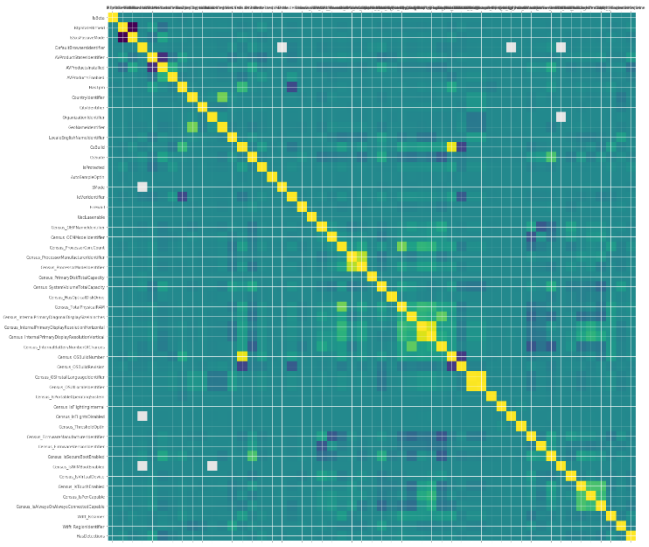
As per this analysis, we do plot for every possible combination of variables with the predictors to determine a set of hypotheses which would have a powerful influence as the predictors & based on our analysis, and we got some fundamental hypothesis which we got from the bivariate analysis.

4.2.4 Uni-Variate Analysis:

We did univariate analysis to determine the distribution of data to see if there are any outliers and to see if there are any missing values present for our dataset. So, the uni-variate analysis is critical for us to see for doing feature engineering. So, if there is a massive set of continuous values, which has a very high variance. So, we can try to take these methodologies to handle those cases. In our case, for the vast values which do not contribute a lot, we are replacing those values with the mode of the population. So, that we could reduce the noise in the data as much as possible.

4.2.5 Correlation Analysis:

The correlation analysis was carried out to make sure that the variables which have high collinearity are not being repeated. So, we removed those columns which have very high collinearity. So, from the graph, we can infer that the column marked in 'yellow' has a very high collinearity. So, we remove such columns that have such high collinearity of more significant than a specified threshold value as per our model design.



4.2.6 Dealing with duplicates and missing values:

To deal with the missing values, all the NaN's we replaced with the mode first. The other approach we adapted is that from the bivariate and univariate analysis, we tried every possible combination & based on its which groups have what type of values in it. Those values were then used to impute the missing values. That type of missing value imputation is beneficial to improve model accuracy.

4.2.7 Analysis of categorical variables:

We tried to understand the distribution of categorical variables and how it affects the outcome as well. A part of which was done as a part of our bivariate analysis as well. Some of the example plots are as shown below:

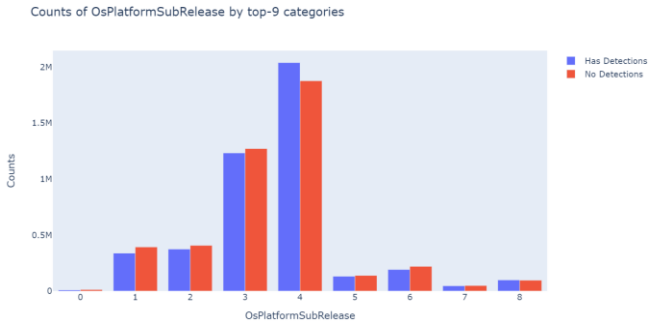


Fig a) Bi-Variate Analysis

For Figure a, we can infer that the 3rd & 4th platforms have a higher number of detections. So, this platform has a higher number of bugs. So, most of the computers had this antivirus. The distribution is maximum for 3 &4. For these categories, we can also infer that the number of detection and No detection is almost the same. So, it has no predictive power. But we can see how the importance of feature selection can be incorporated at this stage.

Fig b)

Bar plot would play a vital role here because it shows what is the distribution of every category in the categorical column and to understand which categories are least represented and which are more important as predictors for our model.

4.2.8 Creating final data:

We are converting every categorical variable into one-hot encoded vectors.

4.2.9 XGBoost Model:

XGBoost stands for "Extreme Gradient Boosting," where the term "Gradient Boosting" originates from the paper Greedy Function Approximation: A Gradient Boosting Machine, by Friedman. It is developed with

both deep consideration in terms of systems optimization and principles in machine learning. The goal of this library is to push the extreme of the computation limits of machines to provide a scalable, portable, and accurate library.

4.2.10 Explanation of why one performs better than others:

Neural Network and LightGBM were also implemented But the computing power required for them was very high. Because of which we could not test it on the test data. But, based on our initial smaller sample of the neural network, the best accuracy was acquired for the neural network. Also, we believe that Neural Network is going to perform better since it supports the nonlinear characteristics and since it works well for data with many variables without saturating in the performance.

4.2.11 Flaws & Challenges in our approach:

We directly imputed missing values. There are much better ways of imputing the missing values which were adapted in the paper.

The volume of data is very high. So, it takes much iterating time. Also, for grid search, it takes days to run the code. So, we needed to use the models that are very GPU compatible to do the entire analysis. We used the GPU from prince.hpc.nyu.edu using the Sbatch scripts and having a dedicated server to run the job. However, for the LighGBM Model, it took 2 days to run as the slurm job on NYU Prince Cluster & terminated in the end due to timeouts. The Neural Network model runs on CPU but is taking much time for completion. We tried to incorporate the Neural Network on GPU, but we faced several errors handling it.

4.3 Results:

ML Model	Data	Accuracy
XGBoost	Training	71.80 %
XGBoost	Testing	69.40 %

From the score, we can infer that our model did not overfit on the training data as we can see that training and testing accuracy are almost similar. The best score achieved by people in Kaggle are 0.7144 in Public Leaderboard and 0.67585 on Private Leaderboard. Our accuracy is 69.40 for the testing data. But we believe that there are better ways of improvement of the accuracy using the neural network.

4.4 Discussion:

There is also a possibility of creating a probabilistic time series modeling. We could also try ensemble approaches. But these are something which is required as a part of future works. We believe that these would yield a very high accuracy. Our model has already solved this issue up to a reasonable extent. It would be exciting to see how we can apply reinforcement learning to this. So that the model keeps learning by itself & tries to predict the malware as new types of malware starts coming into the picture in the future. It could also be used further ahead to design a proper antivirus system that can resist all kinds of attacks on it.

5. CONCLUSIONS

Concerning this machine learning problem, we believe that malware is a crucial thing to be detected and it's one of the essential business problems to be solved. We believe that our machine learning model solved this problem up to a great extent. We can further enhance the performance & accuracy by doing more advanced feature engineering, ensemble modeling.

The baseline model we used, i.e. XGBoost provides a good accuracy of 69.40% which is considerably good.

6. ACKNOWLEDGMENT

We want to thank the several helping hands who contributed to the successful completion of this project. Professor Christopher Policastro, the HPC Team, Section Leader Serkan Karakulak, the graders Raghav Jajodia & Ravi Choudhary for helping us to get to the end of the solution & providing more time for the completion of this project.

7. BIBLIOGRAPHY & REFERENCES

[1] https://www.kaggle.com/c/microsoft-malware-prediction/overview

[2] https://dl-acm-org.proxy.library.nyu.edu/citation.cfm?id=3318448

[3] https://dl-acm-org.proxy.library.nyu.edu/citation.cfm?id=3196515

[4] https://dl-acm-org.proxy.library.nyu.edu/citation.cfm?id=2835834

[5] https://dl-acm-org.proxy.library.nyu.edu/citation.cfm?id=3343639

[6] https://dl-acm-org.proxy.library.nyu.edu/citation.cfm?id=2996769

[7] https://dl-acm-org.proxy.library.nyu.edu/citation.cfm?id=3326818

[8] https://medium.com/datadriveninvestor/what-ive-learned-microsoft-malware-prediction-competition-on-kaggle-3c8189dcc850

[9] https://www.kaggle.com/stuarthallows/using-xgboost-with-scikit-learn

[10] https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.LabelEncoder.html

[11] http://theprofessionalspoint.blogspot.com/2019/03/advantages-of-xgboost-algorithm-in.html

[12] Cornell University: Computer Science: XGBoost: A Scalable Tree Boosting System https://arxiv.org/abs/1603.02754

[13] XGBoost Documentation: https://github.com/dmlc/xgboost

[14] https://towardsdatascience.com/xgboost-mathematics-explained-58262530904a

[15] Introduction to Taylor's theorem: https://mathinsight.org/taylors_theorem_multivariable_introduction

[16] Tianqi Chen, Carlos Guestrin: XGBoost: A Scalable Tree Boosting System https://arxiv.org/abs/1603.02754

[17] Tianqi Chen: Introduction to Boosted Trees: https://homes.cs.washington.edu/~tqchen/pdf/BoostedTree.pdf