Microsoft Challenge: Malware Prediction

Anonymous CVPR submission

Paper ID

Abstract

A comprehensive approach to predict malware in machine running on Microsoft Operating System which includes performing exploratory data analysis and using 4 models. Models includes 1 baseline and 3 advanced models to predict malware. An in depth analysis is also done for each of the model which include but not limited to the reason of the performance, underlying issues and choice of parameters.

1. Introduction

Malware Infection in case of Operating System has proven to be fatal. Wannacry Ransomware attacks that affected the machines running a particular version of Windows Operating System in 2017 resulted in a major impact to National Health Service hospitals in England and Scotland among others.Our project aims to mitigate the above problem by using Machine Learning models to predict malware in machines running microsoft operating system

2. Related Works

In academia as proposed in [2], [10], [4],[8], [9],[7],[6],[1], [3],[5], there exist extensive research on malware detection.The dataset used in this project was hosted on Kaggle as challenge by Microsoft, and was tackled by AI researchers worldwide with start of art algorithm achieving a score of 0.71444 on the public leaderboard.

3. Methodology

The approach employed by our group is divided into 3 stages. The first stage is Data Exploration which includes pre processing techniques and then feature selection to reduce the feature space. The second stage is application of Machine Learning Models and the third stage is analysis of the performance and the tuning of parameters.

3.1. Data Exploration

In data exploration various inherent properties of the test as well as train dataset is studied and plotted using graphs. On top of it, inferences drawn from these is used to pre-process the train and test datasets. The analysis was done across 3 major criteria. They were the distribution of datatypes, class imbalance and missing data.

3.1.1 Distribution of DataTypes

Since the dataset is large, it becomes imperative to study the distribution of the datatypes. The distribution between categorical, binary and true numeric datatype is necessary as they govern which type of encoding is needed to process the dataset properly. As seen from the chart, the number

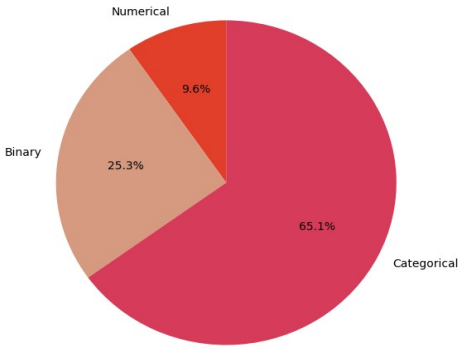

Figure 1. Distribution of DataType

of categorical data constitutes the major part of the dataset for this problem. Therefore while preprocessing the data we need to be mindful of the fact that encoding provides us with some reduced space to work with or we will have to encode by first binning the values in a certain range and then labelling the category.

3.1.2 Class Imbalance

The most essential part of any project is to evaluate if there is any class imbalance in the train dataset. If there is then,

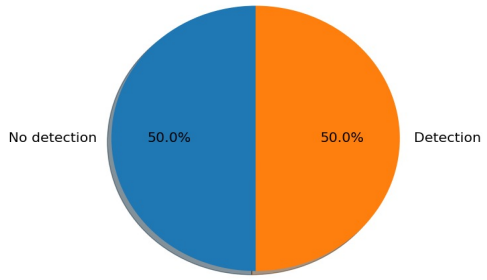

Figure 2. Imbalance in Data

sampling techniques like Synthetic Minority Oversampling TEchnique (SMOTE) needs to be implemented. However in this case the dataset is perfectly balanced.

3.1.3 Missing Data

If most of the data is missing across a certain feature then it is advisable to discard that column rather than replacing the values since such replacement would be underestimate and would prove to be of no use. From the pie chart we

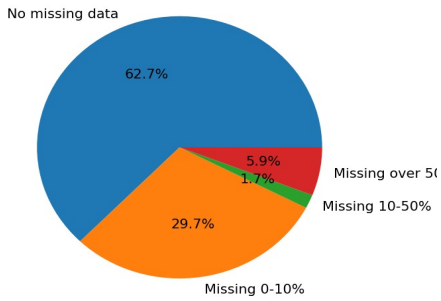

Figure 3. Distribution of Missing Values in Train Dataset

can see that the dataset has a significant number of column with considerable missing data. Columns with more than 30 percent are dropped since for such a large dataset 30 percent is corresponds to a large number.

3.2. Feature Selection

It is essential to evaluate how perfect is the dataset at hand. For this analysis was done for the training set as well as the test set with respect to missing data points. Therefore in the preliminary analysis the features were selected on the base of the percentage of missing values. Since there are features which have over 50 percent of missing values they need to be excluded while training the model. Apart from that features which were highly correlated were dropped also.

3.3. Machine Learning Models

The four models chosen for the study are Support Vector Machines, Random Forest Classifiers, Boosting Models which include LightGBM, XGBoost, CATBoost and Neural Network.

3.3.1 Support Vector Machines

SVM is a supervised machine learning algorithm which can be used for classification or regression problems. It uses a technique called the kernel trick to transform your data and then based on these transformations it finds an optimal boundary between the possible outputs. The reason for using SVM is that we cannot make an assumption of the data being linearly separable hence vanilla logistic regression and a linear kernel SVM would not suffice.

3.3.2 Random Forest Classifier

Random forest, like its name implies, consists of a large number of individual decision trees that operate as an ensemble. Each individual tree in the random forest spits out a class prediction and the class with the most votes becomes our model's prediction. The advantage of using Random Forest over Decision Tree is that a decision tree is built on an entire dataset, using all the features/variables of interest, whereas a random forest randomly selects observations/rows and specific features/variables to build multiple decision trees from and then averages the results. For every machine learning algorithm the objective is to reduce the error. The error term is made up of bias and variance. Random Forest Classifier tries to reduce the error by reducing the variance.

3.3.3 Boosting Models

Boosting is an ensemble learning technique that uses a set of Machine Learning algorithms to convert weak learner to strong learners in order to increase the accuracy of the model. Now Boosting algorithm differ from the Random Forest Classifier in the way that the objective of Boosting is to reduce the error by targeting bias in the error term. In our study we have used 3 boosting algorithm which are namely XGBoost,CatBoost and LightGBM. There are structural differences in which these boosting algorithm differs. LightGBM uses a novel technique of Gradient-based One-Side Sampling (GOSS) to filter out the data instances for finding a split value while XGBoost uses pre-sorted algorithm Histogram-based algorithm for computing the best split. CatBoost works on a similar gradient boosting as LightGBM but is generally found slower than LightGBM

although it works fasts when the number of categorical columns are more in number. The preprocessing of data includes converting categorical data into numeric values. Now for a few columns the categorical data is actually a version number which actually runs into hundreds of thousands. In this study the conversion is done by putting such categorical values into buckets and assigning those buckets a particular number. There are few fundamental difference in which the models work. As explained in previous section, RandomForestClassifier and XGBoost can not handle categorical data without conversion into suitable data types.

3.3.4 Neural Network

Neural Networks have the following advantage over other traditional techniques. They doesn't assume the base features to be informative individually or that it can progress by greedily optimizing along each base feature direction at a time and they naturally build features hierarchically.

4. Results and Analysis

In our study we try to present why one model might be outperforming the other models and we present our rational behind the choice of the parameters. The parameters tuned with respect to the models are as follows: **minsample leaf :** The minimum number of samples required to be at a leaf node. This parameter is similar to minsamples splits, however, this describe the minimum number of samples of samples at the leafs, the base of the tree.

max depth : Used to limit the max depth for tree model. This is used to deal with over-fitting when data is small. Tree still grows leaf-wise. The limit was set to 3 for XGBoost model and no limit was specified for the LGBM model.

n jobs : tells the engine how many processors is it allowed to use. A value of -1 means there is no restriction whereas a value of 1 means it can only use one processor.

max features : represents the number of features to consider when looking for the best split.

CONFUSION MATRIX FOR SVM

637354	478000
413904	701113

CONFUSION MATRIX FOR RANDOMFOREST

637354	478000
413904	701113

CONFUSION MATRIX FOR XGBOOST

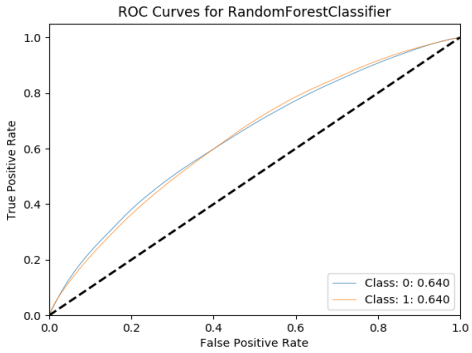

Figure 4. ROC curve for RandomForest Model

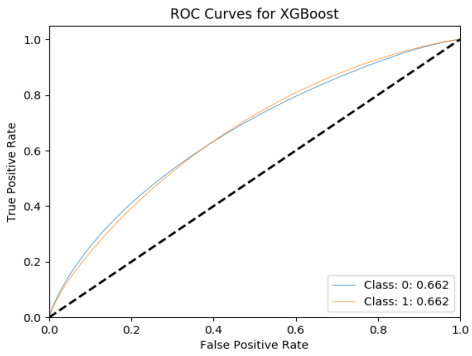

Figure 5. ROC curve for XGBoost Model

610873	504481
350913	764104

CONFUSION MATRIX FOR LIGHTGBM

637354	478000
413904	701113

5. Conclusion

Preliminary result indicate that tree based boosted models are better than the rest models with XGBoost outperforming the simple RandomForest classifier model. Also apart from the score obtained on the challenge, another area which we are trying to focus is to reduce the False Negative Rate and again XGBoost has proven to provide better results.

References

[1] M. Christodorescu and S. Jha. Static analysis of executables to detect malicious patterns. 12, 03 2004. 1

[2] M. G. Schultz, E. Eskin, F. Zadok, and S. Stolfo. Data mining methods for detection of new malicious executables. pages 38–49, 02 2001. 1

[3] G. Jacob, P. Milani Comparetti, M. Neugschwandtner, C. Kruegel, and G. Vigna. A static, packer-agnostic filter to detect similar malware samples. volume 7591, 01 2010. 1

[4] J. Jang, D. Brumley, and S. Venkataraman. Bitshred: feature hashing malware for scalable triage and semantic analysis. pages 309–320, 10 2011. 1

[5] D. Kirat, L. Nataraj, G. Vigna, and B. Manjunath. Sigmal: A static signal processing based malware triage. 12 2013. 1

[6] M. Narouei, M. Ahmadi, G. Giacinto, H. Takabi, and A. Sami. Dllminer: Structural mining for malware detection. *Security and Communication Networks*, 8:n/a–n/a, 04 2015. 1

[7] I. Santos, F. Brezo, X. Ugarte-Pedrero, and P. Bringas. Opcode sequences as representation of executables for data-mining-based unknown malware detection. *Information Sciences*, 231:203–216, 08 2013. 1

[8] G. Yan, N. Brown, and D. Kong. Exploring discriminatory features for automated malware classification. pages 41–61, 07 2013. 1

[9] Y. Ye, T. Li, Y. Chen, and Q. Jiang. Automatic malware categorization using cluster ensemble. pages 95–104, 07 2010. 1

[10] J. Z. Kolter and M. A. Maloof. Learning to detect and classify malicious executables in the wild. *Journal of Machine Learning Research*, 6:2721–2744, 12 2006. 1