Malware Detection

Team Name: .CSV

Nazar Kashif
IMT2018042
International Institute of Information Technology, Bangalore
Nazar.Kashif@iiitb.org

Vinayak Agarwal
IMT2018086
International Institute of Information Technology, Bangalore
Vinayak.Agarwal@iiitb.org

Saad Patel
IMT2018514
International Institute of Information Technology, Bangalore
Mohammad.Saad@iiitb.org

***Abstract*—In Today's world, where everything is digital, a simple virus or bug can result in disasters. A lot of machines become useless because it becomes very difficult to cure a machine after it has been infected with malware. So, it is essential to detect any malware before it has a chance to damage a machine.**

This document is a detailed report on our work on classification of whether a machine will soon be hit with malware or not, given the machine's features. This will help in improving security and performance of a machine.

***Index Terms*—Data Preprocessing, OneHot Encoding, Label Encoding, Cross Validation, Logistic Regression, Random Forest Classifier, XGBoost Classifier, LightGBM Classifier, Randomized Search, Grid Search, EasyEnsemble Classifier,Stacking,**

I. Introduction

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. We need to help protect more than one billion machines from damage BEFORE it happens.

The dataset used in this assignment is a part of Microsoft's malware prediction's dataset of machine properties from Microsoft protection solution, windows defender, which was hosted on Kaggle. Given features of machine, we must predict whether a machine will soon be hit with malware or not, given the machine's features.

Our Final model will help predict the probability of a machine getting infected my malware in the immediate future. This will not only help in reducing damages but also help in increasing security of machines. Our predictions might not be perfect but it at least provide a platform to build on.

II. Dataset

The dataset used in the assignment can be downloaded from here, which was hosted on Kaggle.

Each datapoint in training data consists of 83 columns, out of which there are 81 features and 1 column is machine identifier and 1 column is target variable (HasDetections). Training contains 567730 data points and primarily test data contains 243313 data points. Given the categories and data it's understood that the problem falls under binary classification.

So here machines can be classified into two categories as a) 0 and b) 1. Here class 0 represents a machine which will not get infected with malware. Whereas class 1 represents a machine which will get infected with malware soon.

III. Observations

1. The target is highly imbalanced in the ratio 0.176 (minority:majority) (Below Figure)

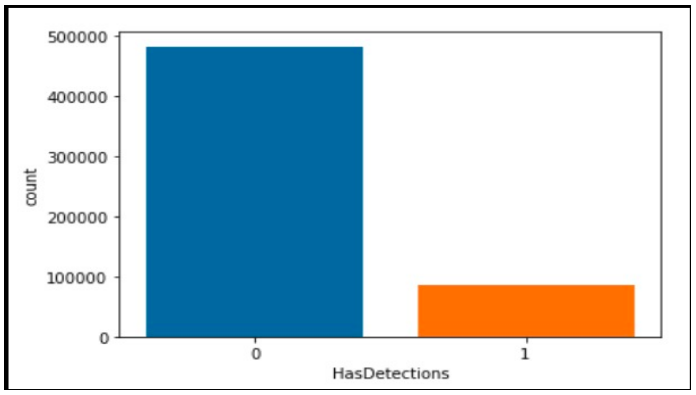

Fig. 1. Distribution of data over the target in the dataset.

2. Null values percentage : Many columns are found to have high missing values percentage. (Below Figure)
a. Pua mode has more than 99% missing values.
b. Census-ProcessorClass has more than 99% missing values.
c. DefaultBrowsersIdentifier has more than 94% missing values.

	nullvaluesPercentage
PuaMode	99.983108
Census_ProcessorClass	99.623818
DefaultBrowsersIdentifier	94.831223
Census_IsFlightingInternal	82.772923
Census_InternalBatteryType	70.533498
Census_ThresholdOptIn	63.156577
Census_IsWIMBootEnabled	63.062131
SmartScreen	36.543932
OrganizationIdentifier	30.824013
SMode	6.746251
CityIdentifier	3.684638
Wdft_IsGamer	3.308209
Wdft_RegionIdentifier	3.308209
Census_InternalBatteryNumberOfCharges	2.855706
Census_FirmwareManufacturerIdentifier	2.269177

Fig. 2. Null value percentages for features in descending order

a. Pua mode has more than 99% missing values.
b. Census-ProcessorClass has more than 99% missing values.
c. DefaultBrowsersIdentifier has more than 94% missing values.

3. Some features have high correlation with other features, and some have low correlation with target variables. (Below Figure)

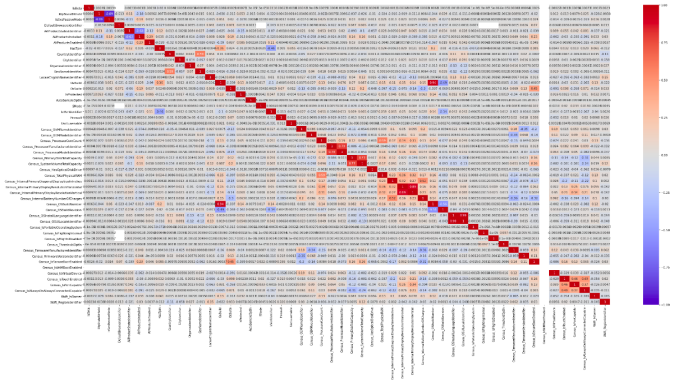

Fig. 3. Heatmap of correlation between target and features

4. Some features also have a high imbalance ratio as >= 0.95, namely Census-IsFlightDisabled, IsBeta, SMode, AutoSampleOptIn, Census-IsportableOperatingSYstem, and Census-IsWIMBootEnabled . (Below Figure)

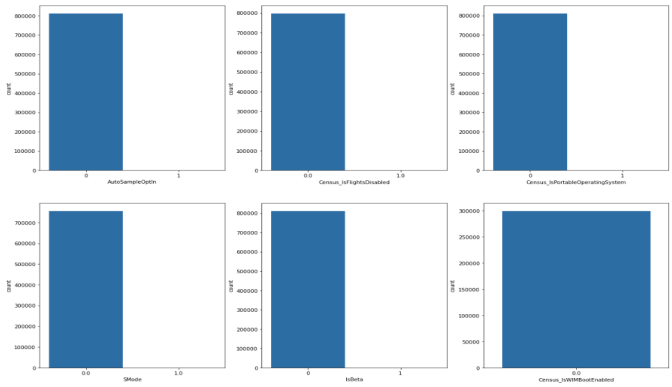

Fig. 4. Count plot indicating imbalance in feature values

5. In the initial dataset given, SmartScreen feature have some categories with different spelling (small and capital letters) So we combined them into one category later.(Below Figure)

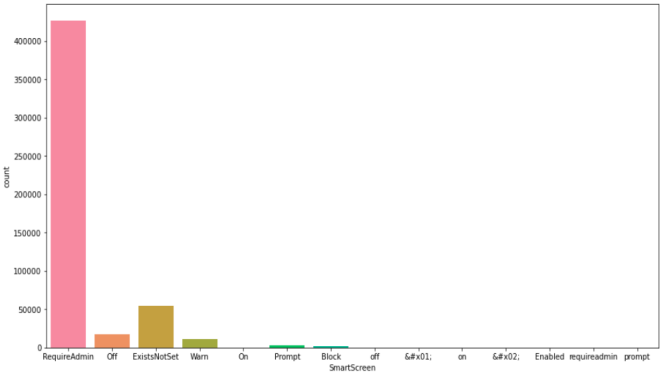

Fig. 5. Distribution of data over the SmartScreen feature in the dataset.

6. In the given dataset, most of the machines have ProductName as win8defender and very few had ProductName as mse, rest of the categories were present rarely(Below Figure)

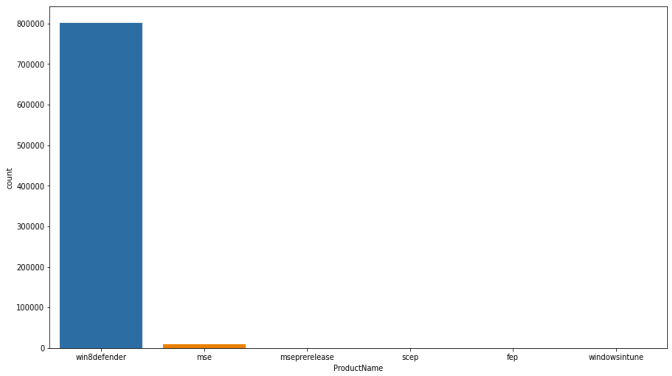

Fig. 6. Distribution of data over the ProductName feature in the dataset.

IV. Data Preprocessing and Feature Extraction

We dropped the features that have a high imbalance ratio >= 0.95, namely Census_IsFlightDisabled, IsBeta, SMode, AutoSampleOptIn, Census_IsPortableOperatingSystem, Census_IsWimBootEnabled.

We filled the missing values in columns with mode of that column. We tried filling various combinations of mean and mode of different columns but mode was found to be best as it was giving the best score.

We have done Label Encoding for the features that have high degree of categories namely AvSigVersion, OsBuildLab, Census_OSVersion, AppVersion, Census_OSSkuName , ProductName, EngineVersion, Census_ChassisTypeName, Census_OSEdition.

We have done One Hot Encoding for the features that have low degree of categories i.e. remaining categorical features. Split train data into train and test for cross validation

We also tried smote and near fit, but it was overfitting and underfitting the data, so we didn't keep that for the final model.

V. Training Details and Results

We tried a variety of combinations of hyperparameters and then found the best combination using grid search and randomised search. We used cv = 5 and cv = kstatified fold(sklearn)

Since there were only 5 submissions per day, we had to take a lot of care that we don't overfit or underfit the data, so many times we tuned the models manually.

We tried stacking the combination of above Xgboost classifier and Lightgbm classifier but we got a low score. So we used imblearn library in which we found easy ensemble classifier to be the best. For the final model, we used both the grid search and randomized search for tuning hyperparameters.

S.No.	Algorithms	AUC - ROC Curve
1	Logistic Regression	0.65030
2	Random Forest Classifier	0.68571
3	XGBoost Classifier	0.69433
4	LightGBM Classifier	0.70448
5	EasyEnsemble Classifier	0.71841

Fig. 7. Algorithms and respective score values.

VI. Conclusion

The model successfully predicts whether a machine will be hit by a malware or not soon with an AUC-ROC curve score of 0.718. The visualization gave an insight and prediction guided us towards improving the model. Such projects have a potential scope to safeguard the machines significantly and also saving lot of resources and time.

VII. Challenges Faced

One of the major challenge was deal with the imbalanced data for which we followed various techniques as described in data preprocessing.

Since there were so many columns, one hot encoding was not able to run on kaggle, so we studied each column separately. Then we tried label encoding on some columns and Onehot encoding on some columns as described above in data preprocessing. Initially we were preprocessing the train and test data separately, so there were inconsistencies in the encodings of the categorical data. So, after that we tried merging the trained and test data set and preprocessed the merged data set.

VIII. References

1. https://seaborn.pydata.org/
2.https://numpy.org/doc/
3.https://in.mathworks.com/help/stats/
4.sklearn.preprocessing.LabelEncoder.html
5.sklearn.preprocessing.OneHotEncoder.html
6.https://xgboost.readthedocs.io/
7.sklearn.ensemble.RandomForestClassifier.html
8.sklearn.preprocessing.StandardScaler.html
9. sklearn.metrics.rocaucscore.html
10.sklearn.modelselection.GridSearchCV.html
11.sklearn.modelselection.RandomizedSearchCV.html
12.imblearn.ensemble.EasyEnsembleClassifier.html
13.https://www.cloudfactory.com/training−dataguide
14. https://www.latexproject.org/help/documentation/