Malware Detection: Project Report

Neetha Reddy
IMT2018050
neetha.reddy@iiitb.org

Nikitha Adivi
IMT2018051
nikitha.adivi@iiitb.org

Rutvi Padhy
IMT2018519
rutvi.padhy@iiitb.org

***Abstract*—This is a detailed report on our attempt to predict a machine's probability of getting infected by malware, based on different properties of the machine. The properties incorporate exhaustive identification of the machine's constituents, enablement and other quantitative information. Based on features corresponding to these properties, we tried building a model that effectively carries out malware detection.**[1]

***Index Terms*—Feature Engineering, Machine learning, XGBoost, Light GBM, Grid Search, Walk Forward Cross Validation, Time Series Split**

DATASET

We were given data representing the properties of a Windows machine. Each row in the given dataset corresponds to a machine, uniquely identified by 'MachineIdentifier'.

The data pertaining to infections was generated by combining heartbeat and threat reports collected by Windows Defender, now known as Microsoft Defender, the anti-malware component of Microsoft Windows that delivers comprehensive, ongoing and real-time protection against software threats like malware. 'HasDetections' indicates whether or not malware was detected on the machine.

Note : The dataset was sampled such that it meets certain business constraints, keeping user's privacy intact and also, realistically limiting the time for which the machine was run. The dataset was also sampled to include a large proportion of malware machines.

I. INTRODUCTION

Malware is a malicious software that is intentionally developed to infiltrate or damage a computer system without the consent of the owner. Malware includes, among others, viruses, worms and Trojan horses. Malware detection refers to the process of detecting the presence of malware on a host system or distinguishing whether a specific program is malicious or otherwise.

The malware industry happens to be a well-organized and well-funded market dedicated to evading traditional security measures. And once a computer is infected by malware, criminals can hurt consumers and enterprises in many ways. Which makes malware one of the most serious security threats. It spreads autonomously through vulnerabilities or carelessness of users and in order to protect a computer from infection or remove malware from a compromised computer system, it is essential to accurately detect malware. Taking into account the seriousness of the issue at hand, we tried building a model which accurately detects such security threats.

[1]This is an InClass Competition for AI511 course 2020 at IIIT Bangalore. Link to competition- https://www.kaggle.com/c/malware-detection-arjun-revised

II. DATA PRE-PROCESSING

The data coming from the train set and that coming from the test set was preprocessed simultaneously. An integral part of data preprocessing is handling null values, which was done as follows :

- ***Dropped features with too many null values***: Features with more than 2,00,000 null values were dropped. The features dropped were 'DefaultBrowsersIdentifier', 'PuaMode', 'SmartScreen', 'Census_ProcessorClass', 'Census_InternalBatteryType', 'Census_IsFlightingInternal', 'Census_ThresholdOptIn' and 'Census_IsWIMBootEnabled'.

- ***Imputed null values for remaining features***: The features that remained were categorized as numerical and categorical. They were dealt with accordingly. The null values in some features were filled as the median of the numerical values available. The null values in other features were filled as indicated by 'top' in describe() that gives the highest counted value of the categorical values available. For example, null values in 'Census_PrimaryDiskTypeName' are filled as 'HDD', null values in 'Census_ChassisTypeName' are filled as 'Notebook' and null values in 'Census_PowerPlatformRoleName' are filled as 'Mobile'.

III. DATA VISUALISATION

The correlation amongst the various features can be visualized using a heatmap. From Fig. 1., we can observe that the features 'Census_OSBuildNumber' and 'Cen-

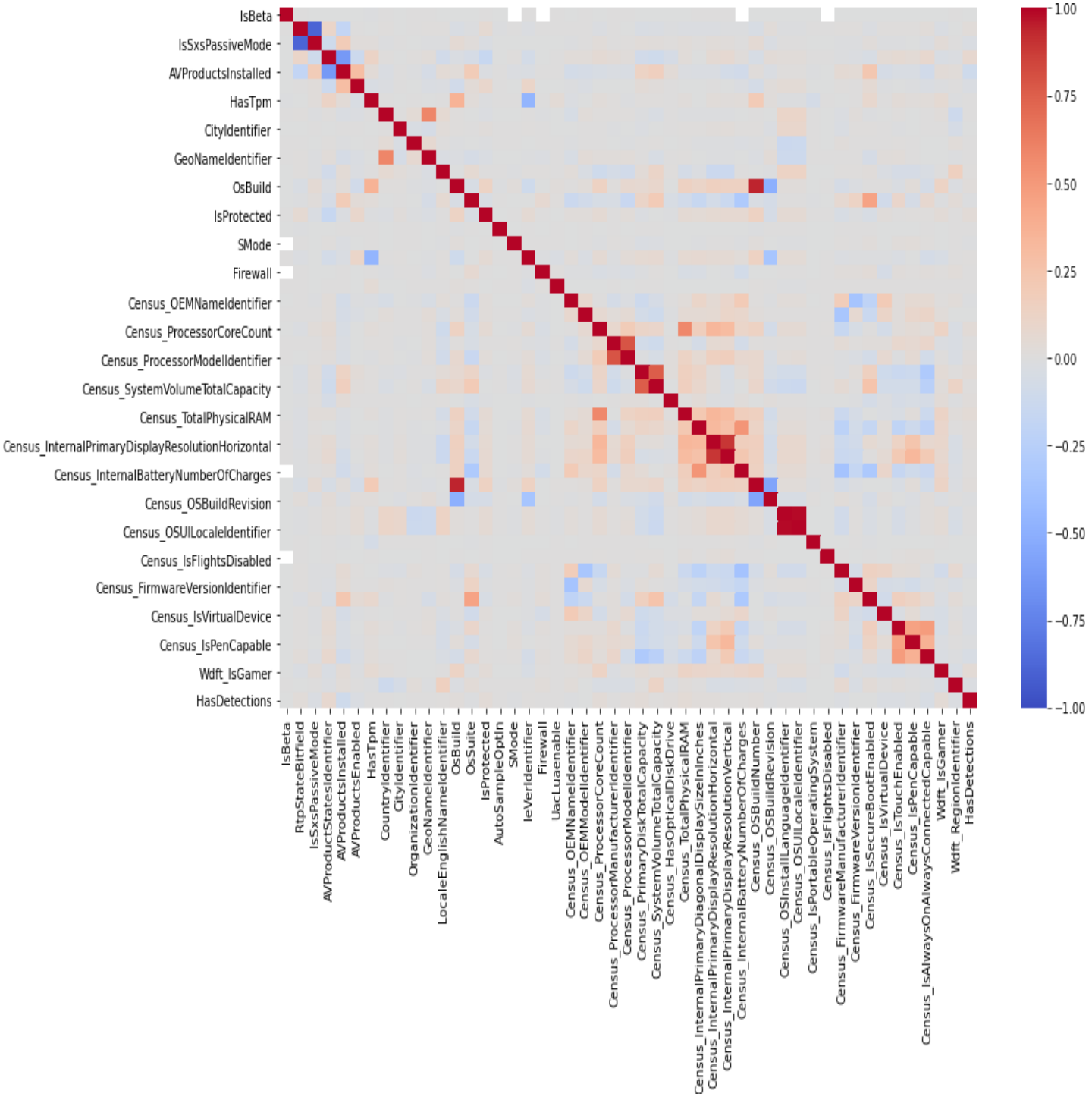

Fig. 1. Correlation Matrix or heatmap

sus_OSUILocaleIdentifier' are highly correlated and hence can be dropped. Some additional features were dropped after visualising the correlation amongst various features. Followed by which, features holding categorical data were encoded accordingly.

IV. Feature Engineering

Features that hold categorical data have either been one-hot encoded or label encoded. Such features with less than or equal to 20 unique categorical data have been one-hot encoded and otherwise, label-encoded, as follows:

- ***One-hot encoded***: 'ProductName', 'Platform', 'Processor', 'OsVer', 'OsPlatformSubRelease', 'SkuEdition', 'Census_MDC2FormFactor', 'Census_DeviceFamily', 'Census_PrimaryDiskTypeName', 'Census_PowerPlatformRoleName', 'Census_OSArchitecture', 'Census_OSBranch', 'Census_OSInstallTypeName', 'Census_OSWUAutoUpdateOptionsName', 'Census_GenuineStateName', 'Census_ActivationChannel' and 'Census_FlightRing'.

- ***Label encoded***: 'EngineVersion', 'AppVersion', 'AvSigVersion', 'OsBuildLab', 'Census_ChassisTypeName', 'Census_OSVersion', 'Census_OSEdition' and 'Census_OSSkuName'.

V. Training and Results

- Our first approach was to try EasyEnsemble with RandomForest Classifier. Since Easy ensemble from imblearn helps in dealing with imbalanced data.

- Then, LightGBM and XGBoost were used for prediction. Hyperparameters were tuned using hyperopt. LightGBM performed better comparatively.

- Since the data also had around 25 categorical variables, CatBoost was used and the features were used without encoding. But it didn't perform well.

The dataset was an unbalanced dataset. This implied the usage of undersampling and oversampling techniques:

- LightGBM after SMOTE oversampling.

- RUSBoost which uses undersampling. As the LGBM model gave the best results until then, LightGBM was used as a base estimator for the same.

TABLE I
Area Under Receiver Operating Characteristic Curve (AUC) on various algorithms

S.No.	Algorithm	AUC
1	EasyEnsemble with RF	0.62297
2	LightGBM	**0.68953**
3	CatBoost	0.66933
4	LGBM with SMOTE oversampling	0.68207
5	XGBoost	0.68090
6	RUSBoost (with LGBM as base estimator)	0.66596

References

[1] https://www.microsoft.com/en-in/windows/comprehensive-security

[2] https://link.springer.com/referenceworkentry/10.1007%2F978-1-4419-5906-5_838

[3] https://www.kaggle.com/somang1418/tuning-hyperparameters-under-10-minutes-lgbm/data