Malware Detection
Team : Matrix

Urja Srivastava
MT2020037
IIIT Bangalore
urja.srivastava@iiitb.org

Mehak Dogra
MT2020090
IIIT Bangalore
mehak.dogra@iiitb.org

Shuhi Maheshwari
MT2020167
IIIT Bangalore
shubhi.maheshwari@iiitb.org

***Abstract*—The problem given in the competition is a Machine Learning problem to predict the probability whether a Windows machine is infected by the Malware or not. This is checked on the basis of different properties/characteristics of the machine.**

***Index Terms*—Exploratory data analysis,Data preprocessing,Feature engineering, encoding, Model Ensembling,Stacking.**

Problem Statement

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 are provided with certain properties of the Window's machine using which we need to detect if the machine is infected with the Malware.

Dataset

The data set given in the competition contain multiple feature properties and each row in this dataset corresponds to a machine, uniquely identified by a MachineIdentifier.There are 83 columns in the dataset, amongst which there are some row values marked with ”NA”

While the dataset provided here has been roughly split by time, the complications and sampling requirements mentioned above may mean you may see imperfect agreement between your cross validation, public, and private scores! Additionally, this dataset is not representative of Microsoft customers' machines in the wild; it has been sampled to include a much larger proportion of malware machines.

Uniquely identified MachineIdentifier in each row corresponds to HasDetections indicating that Malware was detected on the machine.Using the information and labels in train.csv, you must predict the value for HasDetections for each machine in test.csv.

I. Introduction

Malware detection is crucial with malware's prevalence on the Internet because it functions as an early warning system for the computer secure regarding malware and cyber attacks. It keeps hackers out of the computer and prevents the information from getting compromised.As someone who works in computers, you try your best to ensure that malware doesn't affect your system.

The malware can be detected in the system using some features. These features can help us to detect if the system is affected by the malware. We are given with such features using which our model can predict the same.

II. Data Visualization

The dataset consists of features as columns. These are different kinds of data types that is int 64 or float64 while other are categorical data. In the dataset, there are certain features that have very high number of missing values(approx 99 percent).There is a high amount of correlation (higher than 0.6) amongst the numerical features. In the object data types, there is large number of categories. In the dataset, we can also observe that the dataset is highly unbalanced.

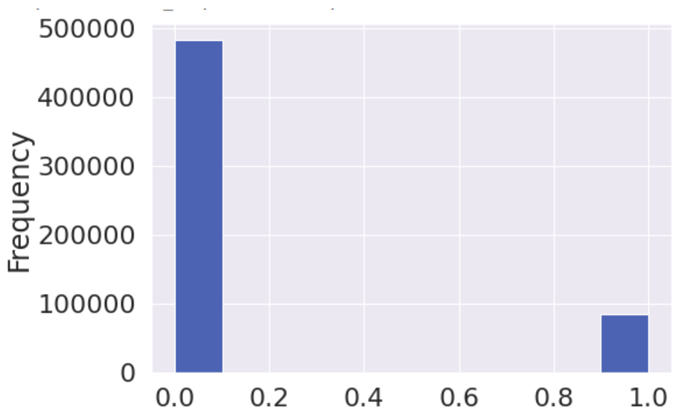


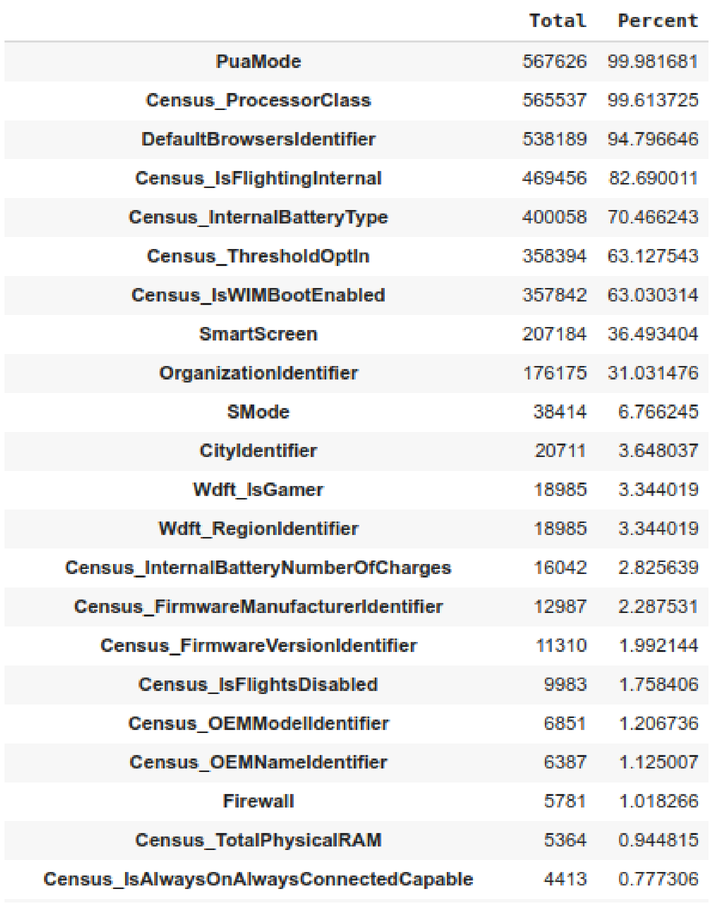
Fig. 1. Target Analysis

III. Data cleaning and Pre-processing

There are total of 83 features in the train dataset while in test dataset there are 82 features. It is observed that hasDetection feature is missing. It is also observed from the dataset that the data is highly skewed and contains a lots of missing values. In the dataset, there are both numerical as well as categorical features.

Handling Missing Data

For the missing values, we first checked the percentage of the missing values by isnull() option from where we got ot the percentahe og NAN in each column. Later, we selected those rows and handled them accordingly.

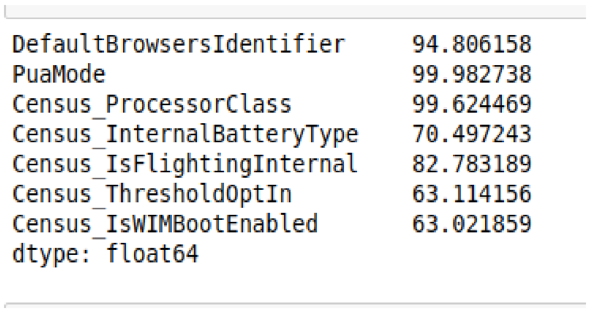


	Total	Percent
PuaMode	567626	99.981681
Census_ProcessorClass	565537	99.613725
DefaultBrowsersIdentifier	538189	94.796646
Census_IsFlightingInternal	469456	82.690011
Census_InternalBatteryType	400058	70.466243
Census_ThresholdOptIn	358394	63.127543
Census_IsWIMBootEnabled	357842	63.030314
SmartScreen	207184	36.493404
OrganizationIdentifier	176175	31.031476
SMode	38414	6.766245
CityIdentifier	20711	3.648037
Wdft_IsGamer	18985	3.344019
Wdft_RegionIdentifier	18985	3.344019
Census_InternalBatteryNumberOfCharges	16042	2.825639
Census_FirmwareManufacturerIdentifier	12987	2.287531
Census_FirmwareVersionIdentifier	11310	1.992144
Census_IsFlightsDisabled	9983	1.758406
Census_OEMModelIdentifier	6851	1.206736
Census_OEMNameIdentifier	6387	1.125007
Firewall	5781	1.018266
Census_TotalPhysicalRAM	5364	0.944815
Census_IsAlwaysOnAlwaysConnectedCapable	4413	0.777306

Fig. 2. Missing Values %

A. Dropping

For the columns with more than 50percent of the missing values we drpped the columns. There are 7 columns with over 50percent of NAN values. In the figure below we can see the columns which were dropped in the process.



```
DefaultBrowsersIdentifier      94.806158
PuaMode                        99.982738
Census_ProcessorClass          99.624469
Census_InternalBatteryType     70.497243
Census_IsFlightingInternal     82.783189
Census_ThresholdOptIn          63.114156
Census_IsWIMBootEnabled        63.021859
dtype: float64
```

Fig. 3. Columns with highest missing values %

B. Imputation

To handle missing data amongst the numeric features we employ mean strategy of imputation. For categorical features Label Encoder automatically assigns a numeric category to missing (NaN) values.

C. Recursive Feature Elimination

RFE is popular because it is easy to configure and use and because it is effective at selecting those features (columns) in a training dataset that are more or most relevant in predicting the target variable. There are two important configuration options when using RFE: the choice in the number of features to select and the choice of the algorithm used to help choose features. Both of these hyperparameters can be explored, although the performance of the method is not strongly dependent on these hyperparameters being configured well.

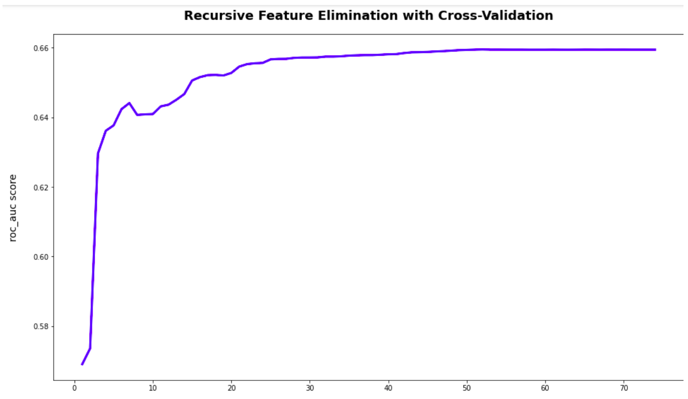
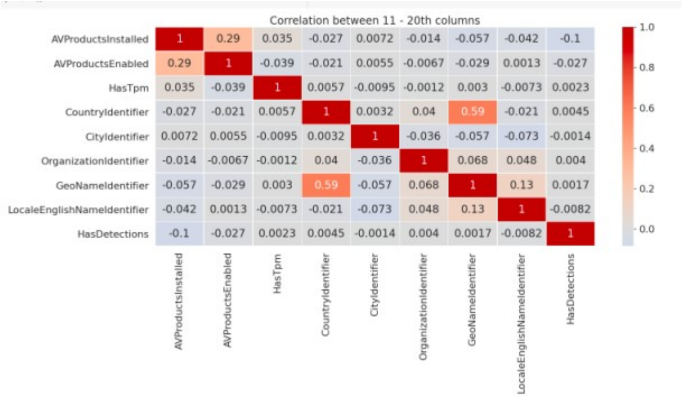


Fig. 4. Recursive Feature Elimination

D. Feature Selection

In other notebook,for the numerical data, we observed a very high correlation. To dropped the columns that we need to remove which were highly correlated i.e., columns with correlation values greater than 0.7 in order to avoid multidimensionality in the data-set.



	AVProductsInstalled	AVProductsEnabled	HasTpm	CountryIdentifier	CityIdentifier	OrganizationIdentifier	GeoNameIdentifier	LocaleEnglishNameIdentifier	HasDetections
AVProductsInstalled	1	0.29	0.035	-0.027	0.0072	-0.014	-0.057	-0.042	-0.1
AVProductsEnabled	0.29	1	-0.039	-0.021	0.0055	-0.0067	-0.029	0.0013	-0.027
HasTpm	0.035	-0.039	1	0.0057	-0.0095	-0.0012	0.003	-0.0073	0.0023
CountryIdentifier	-0.027	-0.021	0.0057	1	0.0032	0.04	0.59	-0.021	0.0045
CityIdentifier	0.0072	0.0055	-0.0095	0.0032	1	-0.036	-0.057	-0.073	-0.0014
OrganizationIdentifier	-0.014	-0.0067	-0.0012	0.04	-0.036	1	0.068	0.048	0.004
GeoNameIdentifier	-0.057	-0.029	0.003	0.59	-0.057	0.068	1	0.13	0.0017
LocaleEnglishNameIdentifier	-0.042	0.0013	-0.0073	-0.021	-0.073	0.048	0.13	1	-0.0082
HasDetections	-0.1	-0.027	0.0023	0.0045	-0.0014	0.004	0.0017	-0.0082	1

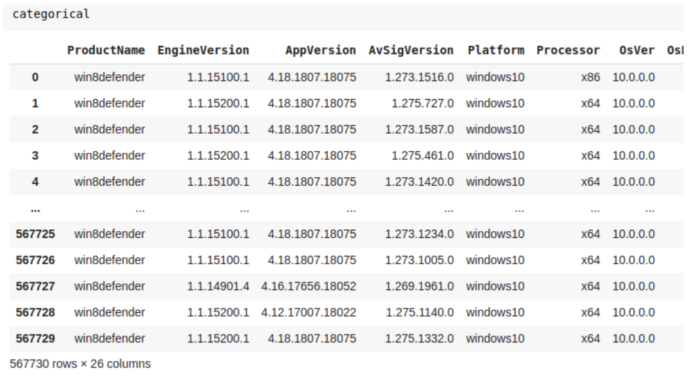
Fig. 5. HeatMap-Correlation Matrix

E. Encoding

On the categorical data, after the split, we used Hot Encoding technique to convert the categorical data into a form that could be provided to ML algorithms to do a better job in prediction. After that we used Frequency Encoding in which machine replaces the category by the frequency -or percentage- of observations in the dataset. In the end we concatenated the spilt dataset and the dataset we get after encoding the categorical variables. from which we get 48 columns.

IV. Feature Engineering

Feature Engineering and handling of features by data type. We divided the dataset in smaller datasets depending on their feature types. Firstly We worked with categorical data which had the datatype "object".On making this division We got 26 separate columns. On those columns we applied different techniques to extract more information from them. We first split the categorical data and stepped further with encoding. Similarly we did get a separate dataset for int data type and other for float data type.



```
categorical
```

	ProductName	EngineVersion	AppVersion	AvSigVersion	Platform	Processor	OsVer	Os
0	win8defender	1.1.15100.1	4.18.1807.18075	1.273.1516.0	windows10	x86	10.0.0.0	
1	win8defender	1.1.15200.1	4.18.1807.18075	1.275.727.0	windows10	x64	10.0.0.0	
2	win8defender	1.1.15100.1	4.18.1807.18075	1.273.1587.0	windows10	x64	10.0.0.0	
3	win8defender	1.1.15200.1	4.18.1807.18075	1.275.461.0	windows10	x64	10.0.0.0	
4	win8defender	1.1.15100.1	4.18.1807.18075	1.273.1420.0	windows10	x64	10.0.0.0	
...	...	...	...	...	...	...	...	
567725	win8defender	1.1.15100.1	4.18.1807.18075	1.273.1234.0	windows10	x64	10.0.0.0	
567726	win8defender	1.1.15100.1	4.18.1807.18075	1.273.1005.0	windows10	x64	10.0.0.0	
567727	win8defender	1.1.14901.4	4.16.17656.18052	1.269.1961.0	windows10	x64	10.0.0.0	
567728	win8defender	1.1.15200.1	4.12.17007.18022	1.275.1140.0	windows10	x64	10.0.0.0	
567729	win8defender	1.1.15200.1	4.18.1807.18075	1.275.1332.0	windows10	x64	10.0.0.0	

567730 rows × 26 columns

Fig. 6. categorical columns

A. Merging Different DataSets

After all divisions of subsets and editing of the datasets of different data types we will merge all the data sets to form a single data set on which the final model will work.

B. Adding new Features

Following two extra features were added in the dataset- Avnew - difference between the number of av products installed and the number of av products enabled. Gamer with firewall- feature interaction to capture whether gamer has firewall enabled or not.

V. Model Ensembling

Ensemble modeling is a process where multiple diverse models are created to predict an outcome, either by using many different modeling algorithms or using different training data sets. The ensemble model then aggregates the prediction of each base model and results in once final prediction for the unseen data. The motivation for using ensemble models is to reduce the generalization error of the prediction.

A. Logistic Regression

The logistic classification model (or logit model) is a binary classification model in which the conditional probability of one of the two possible realizations of the output variable is assumed to be equal to a linear combination of the input variables, transformed by the logistic function.

B. Light GBM

Light GBM is a fast, distributed, high-performance gradient boosting framework based on decision tree algorithm, used for ranking, classification and many other machine learning tasks.Since it is based on decision tree algorithms, it splits the tree leaf wise with the best fit whereas other boosting algorithms split the tree depth wise or level wise rather than leaf-wise.

C. XGBoost

XGBoost is an optimized distributed gradient boosting library designed to be highly efficient, flexible and portable. It implements machine learning algorithms under the Gradient Boosting framework. XGBoost provides a parallel tree boosting that solve problems in a fast and accurate way.

D. Stratified K-Fold

Approach used for model evaluation is the train/test split and the k-fold cross-validation procedure. Both approaches can be very effective in general, although they can result in misleading results and potentially fail when used on classification problems with a severe class imbalance. Instead, the techniques must be modified to stratify the sampling by the class label, called stratified train-test split or stratified k-fold cross-validation.Each fold tries to depicts or duplicates the original data.

VI. Training the model

After feature engineering, our number of useful features reduced.Reducing the features also decreased the complexity of the model as well.After modifying the model parameters we could improve the model accuracy.Also we tried different supervised learning models to improve our accuracy.We trained our data on the following algorithms:

- Linear Model(LM)
- Logistic Regressor(LR)
- LightGBM(LGBM)
- XGBoost(XGB)
- Stratified KFold(SKFold)

S.No.	Algorithm	roc_auc score
1.	Linear Model	0.58026
2.	Logistic Regression	0.66807
3.	Logistic + Random Forest	0.69921
4.	LGBM+XGB+CatBoost	0.71672
5.	Stratified KFold LGBM	0.71730
8.	Stratified KFold=5 + XGBoost	0.72193

The k value must be chosen carefully for your data sample.A poorly chosen value for k may result in a mis-representative idea of the skill of the model, such as a score with a high variance,or a high bias, (such as an overestimate of the skill of the model).

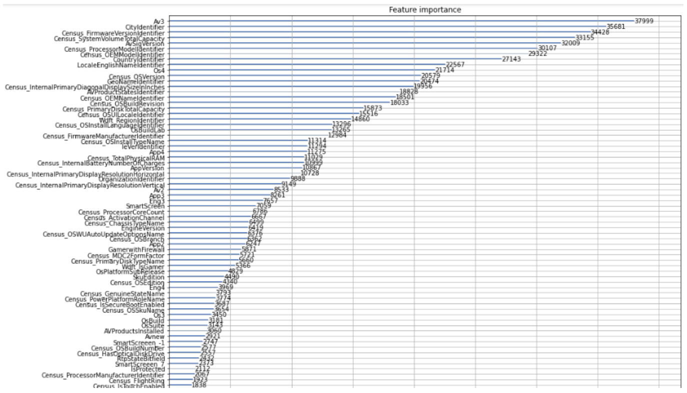


Fig. 7. Execution Importance

Our engineered features are doing very well on the feature importance graph and gave us that boost in prediction accuracy.

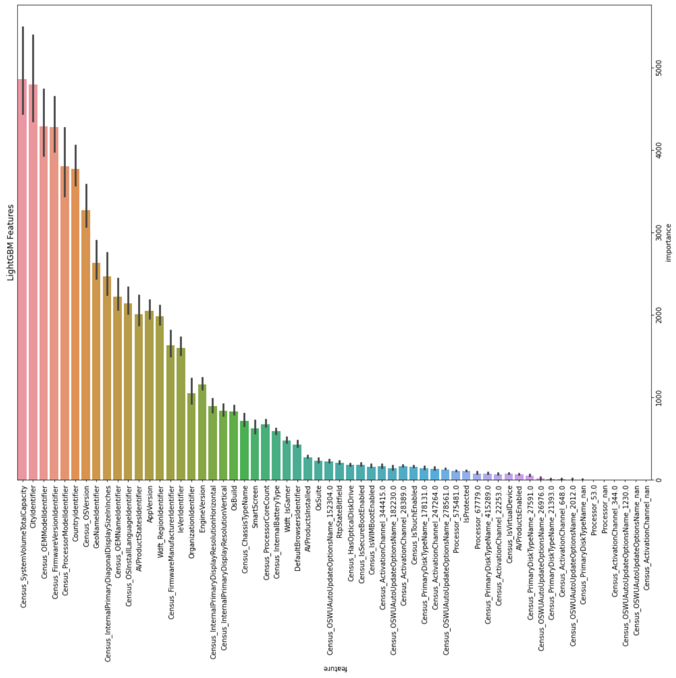


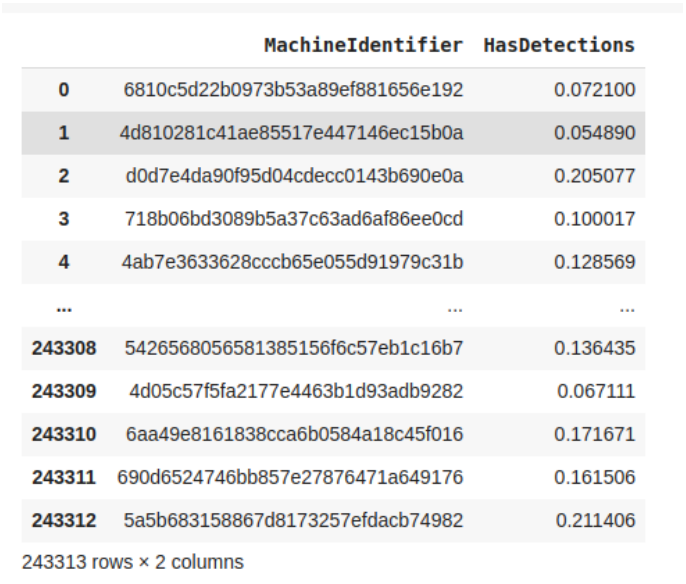
Fig. 8. Feature Relevance & Selection

VII. Conclusion

Using the stratified Light GBM and Stratified XGB,our model was able to predict whether the malware was detected by the system or not with an accuracy of 72.193% .Also we could predict which are the more dominant features that helps in better prediction thereby increasing the model accuracy.

VIII. Acknowledgement

We would like to thank Professor G. Srinivas Raghavan and our Machine Learning teaching assistant Vibhav Agarwal,for giving us the opportunity to work on the project and helping us out in the initial stages in numerous occasions. His highly detailed lectures on various topics helped us understand what we were actually doing.
Leader-board was great motivation to work on the project and the competition forced us to read up various articles and papers which gave us ideas and enthusiasm for the project.



	MachineIdentifier	HasDetections
0	6810c5d22b0973b53a89ef881656e192	0.072100
1	4d810281c41ae85517e447146ec15b0a	0.054890
2	d0d7e4da90f95d04cdecc0143b690e0a	0.205077
3	718b06bd3089b5a37c63ad6af86ee0cd	0.100017
4	4ab7e3633628cccb65e055d91979c31b	0.128569
...	...	...
243308	5426568056581385156f6c57eb1c16b7	0.136435
243309	4d05c57f5fa2177e4463b1d93adb9282	0.067111
243310	6aa49e8161838cca6b0584a18c45f016	0.171671
243311	690d6524746bb857e27876471a649176	0.161506
243312	5a5b683158867d8173257efdacb74982	0.211406

243313 rows × 2 columns

Fig. 9. Prediction of our final model

IX. References

[1] Jovan Sardinha. An introduction to model ensembling. [Online]. Available: https://medium.com/weightsandbiases/an-introduction-to-model-ensembling-63effc2ca4b3
[2] Philip Hyunsu Cho, Nan Zhu et.al., XGBoost. [Version: 1.2.0]. [Online]. Available: https://xgboost.readthedocs.io/
[3] Jérémie du Boisberranger, Joris Van den Bossche et.al., sckit-learn: Open source scientific tools for Machine Learning. [Latest] [Online]. Available: https://scikit-learn.org/
[4] Microsoft Corporation Revision 9597326e. LightGBM. [Latest]. [Online]. Available: https://lightgbm.readthedocs.io/
[5] Andrei (Andrey) Khropov, annaveronika et.al., catboost.ai. [version: 0.24.3]. [Online] https://github.com/catboost/catboost