MALWARE DETECTION SANTANDER TEAMS

Team: TAMEthatBENCH

IMT2018004 - Agam Kashyap

IMT2018073 - Soham Kolhe

IMT2018077 - Tanishq Jaswani

Problem Statement

In this competition, we had to predict whether or not a computer has been infected by some malware. Based on different properties and features provided to us, we had to build a model which predicts the probability of such an infection. This document describes our approach and our observations for the same.

Introduction

Malicious software are abundant in a world of innumerable computer users, who are constantly faced with these threats from various sources like the internet, local networks and portable drives. Malware is potentially low to high risk and can cause systems to function incorrectly, steal data and even crash.

Malware may be an executable or system library files in the form of viruses, worms, Trojans, all aimed at breaching the security of the system and compromising user privacy. Typically, anti-virus software is based on a signature definition system which keeps updating from the internet and thus keeping track of known viruses. While this may be sufficient for home-users, a security risk from a new virus could threaten an entire enterprise network.

So for this challenge, we look at the different states of a machine and their properties to predict the probability of a given machine to detect malware.

Dataset

Observations:

Step1: Data types

Printing the data types of all columns and the unique values they took, we converted to their simplest form as possible for better memory management.

Train.shape: (567730, 83)

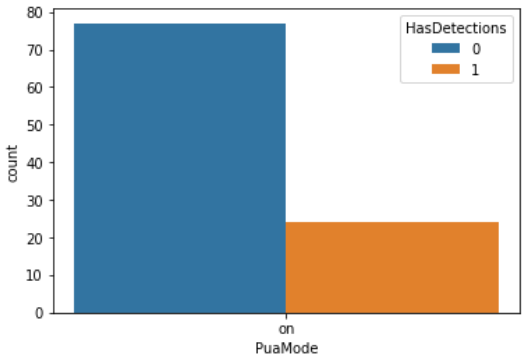
Test.shape: (243313, 82)

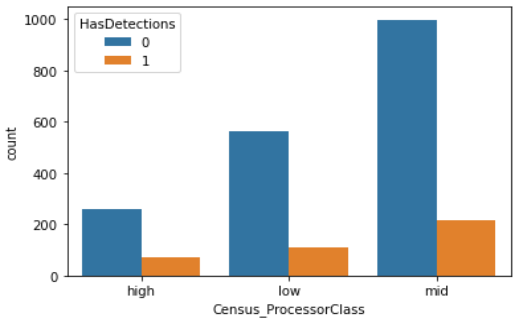
Step2: Null values

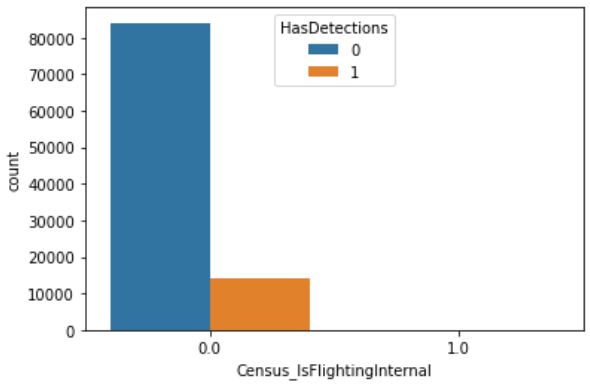
Columns which contained null values more than 70%:

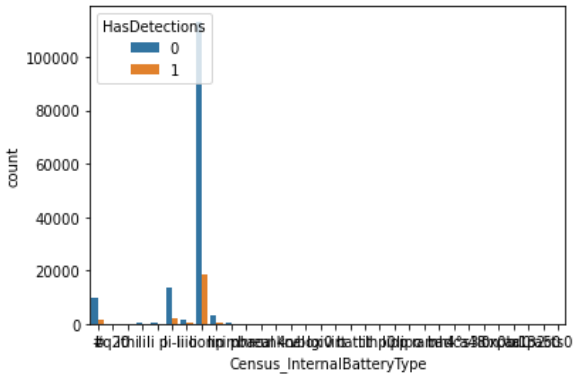
```
PuaMode                         0.999822
Census_ProcessorClass           0.996097
DefaultBrowsersIdentifier       0.948148
Census_IsFlightingInternal      0.827191
Census_InternalBatteryType      0.704935
                                   ...
Census_PowerPlatformRoleName    0.000000
Census_HasOpticalDiskDrive      0.000000
Census_DeviceFamily             0.000000
Census_MDC2FormFactor           0.000000
MachineIdentifier               0.000000
Length: 83, dtype: float64
```

Before dropping these columns we checked their distribution:









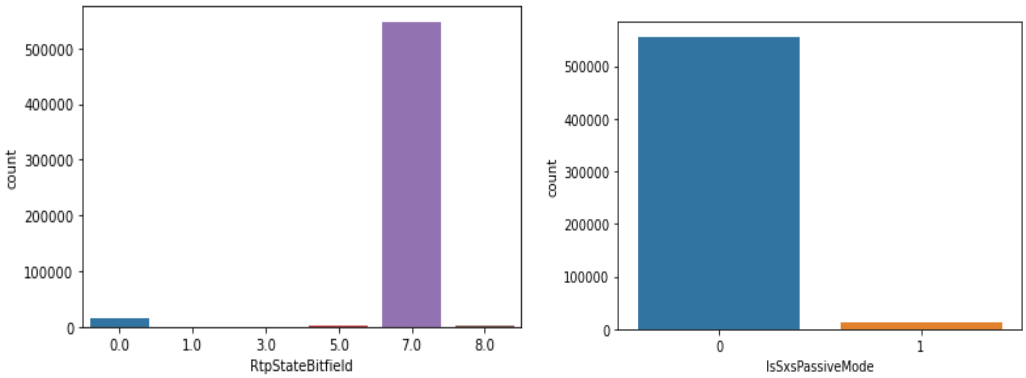
Step3: Skewness

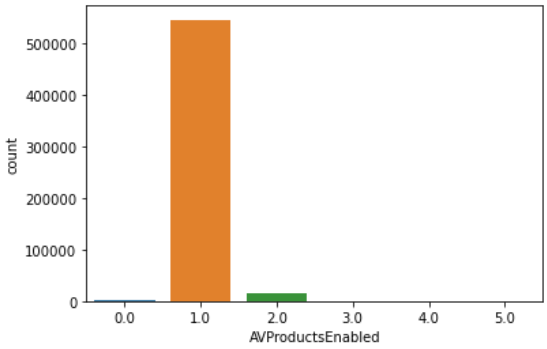
	column	uniq	skewness
75	Census_IsWIMBootEnabled	1	100.000000
28	PuaMode	1	100.000000
5	IsBeta	2	99.999295
68	Census_IsFlightingInternal	2	99.998981
69	Census_IsFlightsDisabled	2	99.998206
...	...	...	...
4	AvSigVersion	6895	1.167633
14	CityIdentifier	39744	1.120959
73	Census_FirmwareVersionIdentifier	25052	1.020502
44	Census_SystemVolumeTotalCapacity	151844	0.632399
0	MachineIdentifier	567730	0.000176

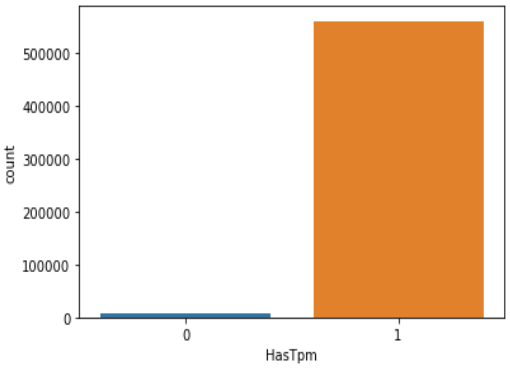
Step4: Correlation

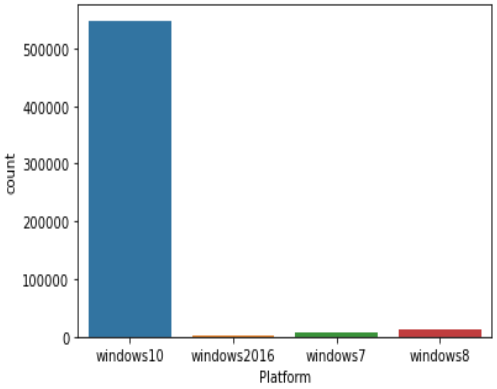
```
HasDetections                                          1.000000
AVProductsInstalled                                    0.103119
AVProductStatesIdentifier                              0.078115
Census_IsAlwaysOnAlwaysConnectedCapable                0.041704
Census_ProcessorCoreCount                              0.039946
Census_TotalPhysicalRAM                                0.038840
Wdft_IsGamer                                           0.038725
IsProtected                                            0.036982
Census_PrimaryDiskTotalCapacity                        0.035369
Census_IsVirtualDevice                                 0.031102
AVProductsEnabled                                      0.029671
Census_IsTouchEnabled                                  0.028765
RtpStateBitfield                                       0.026372
Census_InternalPrimaryDiagonalDisplaySizeInInches      0.025599
Census_InternalPrimaryDisplayResolutionHorizontal      0.022775
Census_FirmwareManufacturerIdentifier                  0.022401
IsSxsPassiveMode                                       0.021879
Census_OSBuildNumber                                   0.020731
OsBuild                                                0.017349
Census_ProcessorModelIdentifier                        0.016958
Census_InternalBatteryNumberOfCharges                  0.016181
OsSuite                                                0.015415
Wdft_RegionIdentifier                                  0.014468
Census_HasOpticalDiskDrive                             0.013538
IeVerIdentifier                                        0.011571
Census_OEMNameIdentifier                               0.011138
Census_SystemVolumeTotalCapacity                       0.011101
Census_IsPenCapable                                    0.011057
```

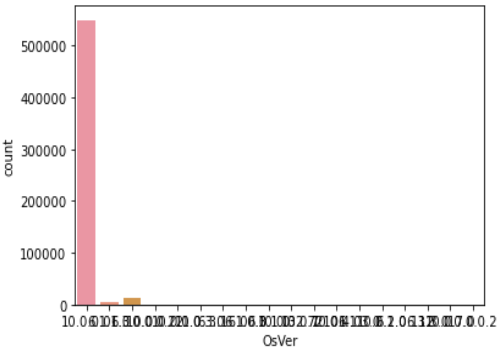
Step5: 90% unbalanced data

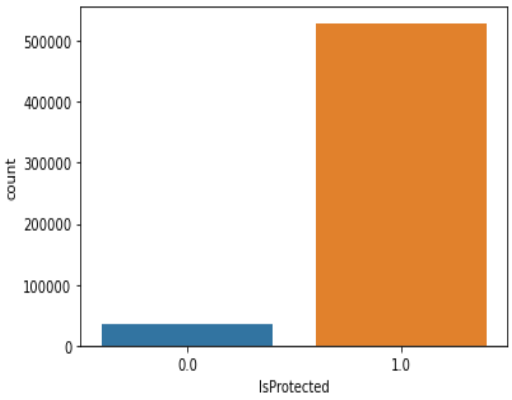


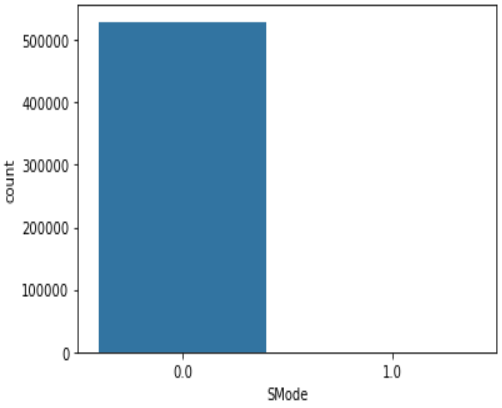

count
500000
400000
300000
200000
100000
0
0.0
1.0
2.0
3.0
4.0
5.0
AVProductsEnabled

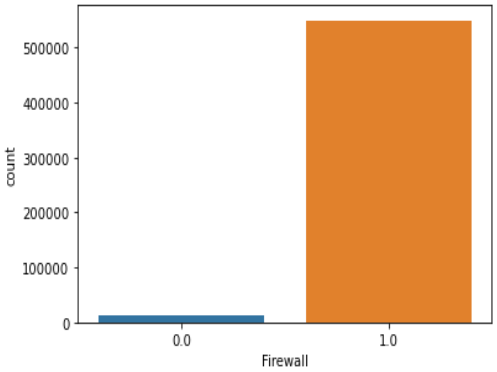

count
500000
400000
300000
200000
100000
0
0
1
HasTpm

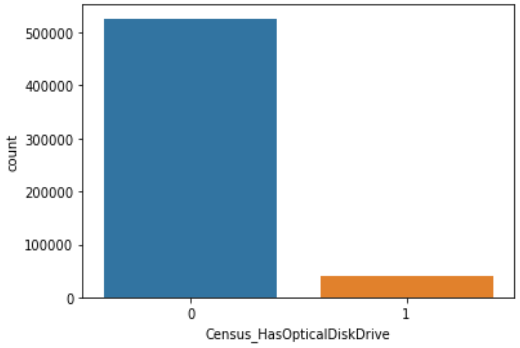

count
500000
400000
300000
200000
100000
0
windows10
windows2016
windows7
windows8
Platform

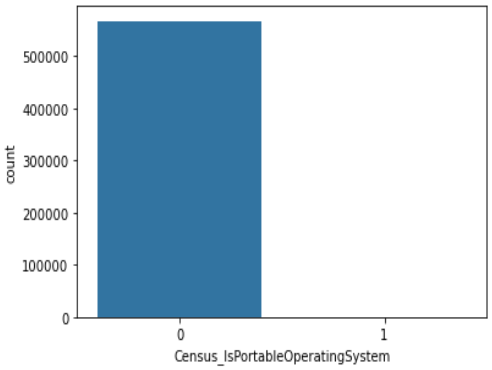

count
500000
400000
300000
200000
100000
0
OsVer

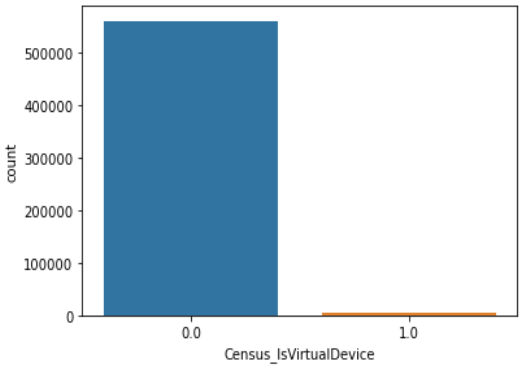

count
500000
400000
300000
200000
100000
0
0.0
1.0
IsProtected

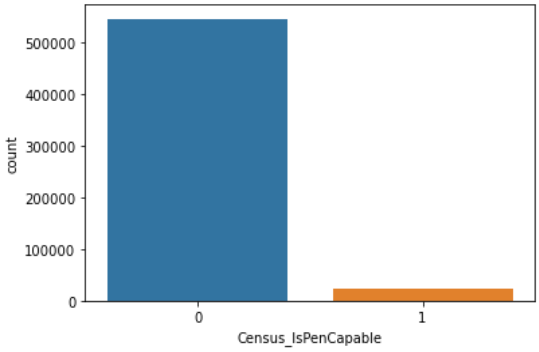

count
500000
400000
300000
200000
100000
0
0.0
1.0
SMode

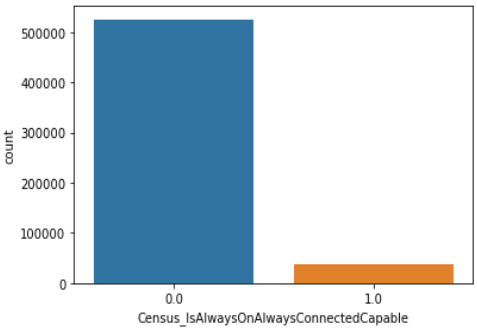

500000
400000
300000
200000
100000
0
count
0.0
1.0
Firewall


500000
400000
300000
200000
100000
0
count
0
1
Census_HasOpticalDiskDrive


500000
400000
300000
200000
100000
0
count
0
1
Census_IsPortableOperatingSystem


500000
400000
300000
200000
100000
0
count
0.0
1.0
Census_IsVirtualDevice


500000
400000
300000
200000
100000
0
count
0
1
Census_IsPenCapable


500000
400000
300000
200000
100000
0
count
0.0
1.0
Census_IsAlwaysOnAlwaysConnectedCapable

Hence, dropping all Imbalanced data i.e. factor>0.9 and mostly null values i.e. factor>0.7

Hence we dropped 29 columns.

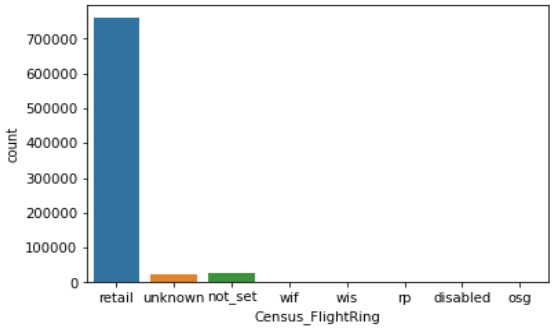
Step6: Filling null values

Filling null values of all the necessary columns:

- Filled null values of boolean columns with mode.
- Filled null values of numerical columns with -1.
- Filled null values of categorical columns with ‘unknown’ and removed unused categories.

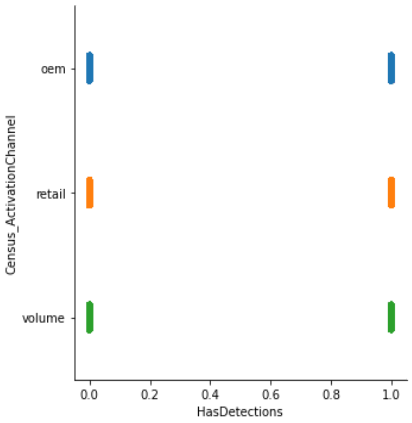
Step6: Combining categories of few columns

1. **Census_FlightRing:**

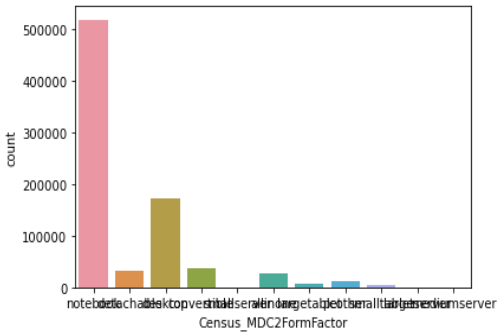


- Combing ‘unknown’ and ‘not_set’ to ‘unknown’
- Combing 'osg', 'rp', 'wis', 'wif', ‘disabled’ to ‘unknown’

2. Census_ActivationChannel

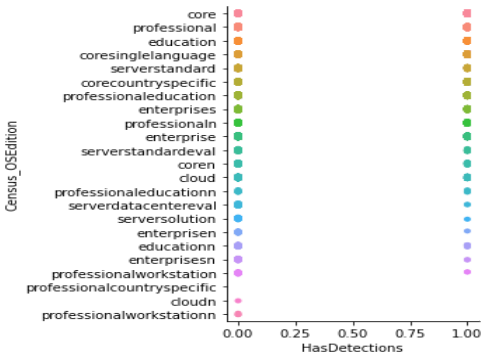


3. Census_MDC2FormFactor:



- Combining all ‘mediumserver’, ‘smallserver’, ‘largeserver’ categories to ‘server’.
- Combining ‘largetablet’, ‘smalltablet’ categories to ‘tablet’.

4. Census_OSEdition



- Combine - ‘enterprise’, ‘enterprises’, ‘enterprisesn’, ‘enterprisen’
- Combine - ‘professional’, ‘professionaleducation’, ‘professionaln’, ‘professionalworkstation’ , ‘professionaleducationn’, ‘professionalworkstationn’, ‘professionalcountryspecific’
- Combine - ‘core’, ‘coresinglelanguage’, ‘corecountryspecific’, ‘coren’
- Combine - ‘serverstandard’, ‘serverstandardeval’, ‘serverdatacentereval’, ‘serversolution’
- Combine - ‘education’, ‘educationn’
- Combine - ‘cloud’, ‘cloudn’

Similarly for Census_OSSkuName, PowerPlatformRoleName and SmartScreen.

Applied label encoding and feature importance to remove more unnecessary columns.

Model

Our final model is based on LightGBM with the following hyperparameters

```
boosting_type ='gbdt' - default
num_leaves = 30
max_depth = 5
learning_rate = 0.1 - default
n_estimators = 500
subsample_for_bin = 200000
objective = None
class_weight = None
min_split_gain = 0.0
min_child_weight = 0.001
min_child_samples = 20
subsample = 1.0
subsample_freq = 0
colsample_bytree = 1.0
reg_alpha = 0.0
reg_lambda = 0.0
random_state = None
n_jobs = - 1
silent = True
importance_type = 'split'
```

- **Why LightGBM:**
    - Faster training speed and higher efficiency: Light GBM uses histogram based algorithm i.e it buckets continuous feature values into discrete bins which fasten the training procedure. Lower memory usage: Replaces continuous values to discrete bins which result in lower memory usage.

Series of scores:

Cases	Public Leaderboard	Private Leaderboard
Case 1	0.65375	0.65533
Case 2	0.65552	0.65980
Case 3	0.66074	0.66316
Case 4	0.71210	0.71516

Case1:

After dropping all Imbalanced data i.e. factor>0.9 and mostly null values i.e. factor>0.7 we realised we have dropped 29 columns in total and our applying random forest classifier our score was 0.65375.

Case2:

After applying LightGBM to the same data processing our score increased to 0.65552.

Case3:

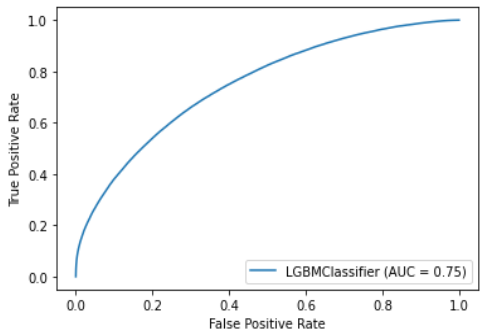
After applying feature importance and removing 3 more unnecessary columns our score increased to 0.65896 and taking it a bit further and dropping 1 or 2 more columns scored went to 0.66074.

Case4:

After all this, we realised removing 32 columns we have lost most of our data and hence we should add a few columns and check the score.

Added 3-4 columns necessary columns which led to an increase in score to 0.67919. Taking this approach further, our score went from 0.67919 to 0.68006 to 0.71210.

Model Score%: 86.05411022845367



Final score: Private Leaderboard: 0.71516 Public Leaderboard: 0.71210

Conclusion:

We would like to conclude that we were able to come up with an efficient model to predict the malware of a given system. Such projects have a potential scope to make the system secure and malware-free.

Acknowledgement:

We would like to thank Professor G. Srinivas Raghavan and our Machine Learning Teaching Assistant, Tejas Kotha for giving us the opportunity to work on this challenge and helping us whenever we were stuck by giving us ideas and resources to learn from. We would also like to thank other teams for being a great competitor and setting a benchmark time by time for the rest of us which acted as a driving fuel for us to constantly work hard and surpass them. We would gladly say that we had a great learning experience while working on the project. The leaderboard was a great motivation to work on the project.

Project Link:

https://www.kaggle.com/c/malware-detection-tejas/overview

References:

- https://lightgbm.readthedocs.io/en/latest/Parameters-Tuning.html
- Random Forest Tutorial