Microsoft Malware Prediction

A Malware prediction Machine Learning Model

1. Introduction

Imagine one day you open your laptop and type in the password. To your surprise it's wrong, you try it again in hope that previously you made a mistake in typing it. It is still the wrong password. You realise something wrong with your system and suddenly your screen becomes black, a text pops up on your command prompt (Black screen). "If you want access to your system, pay $10,000". This is called Ransomware, one of many types of attacks that are possible on our systems with the help of injecting malware into our systems. Malware is nothing but a piece of code that enters our system and transfers ownership access to some other system

that's mentioned in the malware. Now, the person using this transferred ownership system can access all your data.

Most of this can be avoided by installing a powerful Antivirus software. But, can we reduce our chances of being attacked by following a few simple steps so that we can save a great amount of money that might be paid if we consider antivirus software options?

This is where our Objective of this project starts. Microsoft wanted to find these types of patterns in their Operating system (Windows), so it conducted a kaggle challenge by providing data about systems using windows OS and also provided us with information of whether that system is affected by any malware.

Objective of this project is to understand whether any of the given information is connected to a system getting infected by a virus. If yes, what are those features or specifications. And using all this information and knowledge, develop a machine learning model that predicts the probability of a system being affected by a malware.

2. Dataset and Processing

2.1 Dataset information

Dataset is available on the kaggle website. It contains nearly 9 million rows where each row represents a system that runs on windows os. And it has totally 83 features (columns) that specify entire information about systems. We have a target variable, HasDetections which is a boolean. If it's 1 - malware is in the system else no malware.

2.2 Data Cleaning

Initially, because of the manganomonity of the data, I planned to use another package called Dask which works on the concept of lazy loading. But, my system was powerful enough to load the dataset without any issues, So to reduce the loading time I have dropped the idea of using Dask and continued with pandas. Due to more features and rows, It's better to get rid of those columns that won't help our

prediction model. Started by checking the number of null values in each column and calculated null values percentages per column.

	Column	Missing columns count	Missing %
0	PuaMode	8919174	99.974119
1	Census_ProcessorClass	8884852	99.589407
2	DefaultBrowsersIdentifier	8488045	95.141637
3	Census_IsFlightingInternal	7408759	83.044030
4	Census_InternalBatteryType	6338429	71.046809
5	Census_ThresholdOptIn	5667325	63.524472
6	Census_IsWIMBootEnabled	5659703	63.439038
7	SmartScreen	3177011	35.610795
8	OrganizationIdentifier	2751518	30.841487
9	SMode	537759	6.027686
10	CityIdentifier	325409	3.647477
11	Wdft_IsGamer	303451	3.401352
12	Wdft_RegionIdentifier	303451	3.401352
13	Census_InternalBatteryNumberOfCharges	268755	3.012448
14	Census_FirmwareManufacturerIdentifier	183257	2.054109
15	Census_IsFlightsDisabled	160523	1.799286
16	Census_FirmwareVersionIdentifier	160133	1.794915
17	Census_OEMModelIdentifier	102233	1.145919
18	Census_OEMNameIdentifier	95478	1.070203
19	Firewall	91350	1.023933
20	Census_TotalPhysicalRAM	80533	0.902686

Fig 2.1 - Null percentages per column.

Dropped all those features whose null values percentage was more than 60%.Then checked for high cardinality columns and dropped them too. Then checked for high cardinality columns and dropped them too.Pandas profiling package helped in dropping those columns as it provided a report on each column. Left with 72 features, the next target is to eliminate nulls from the remaining features. It was handled by imputing most frequent values in categorical and mean in numerical columns.Later dropped all those rows with null and could retain 90% of the data with was 7.98 million rows.

2.3 Correlation Matrix

Using the cleaned data frame and selecting numeric features from it, generated a correlation matrix. It revealed few correlations as shown in figure below (Fig 2.2) .

1.) IsSxsPassiveMode and RtpStateBitfield are strongly correlated with value 0.9. It makes me decide to drop one of the columns with lesser information but decided to move forward for now as it helps in finding few insights in EDA where later we can drop one column while modeling.

2.) Census_InternalPrimaryDisplayHorizontal and Census_InternalPrimaryDisplayVertical are also strongly correlated.

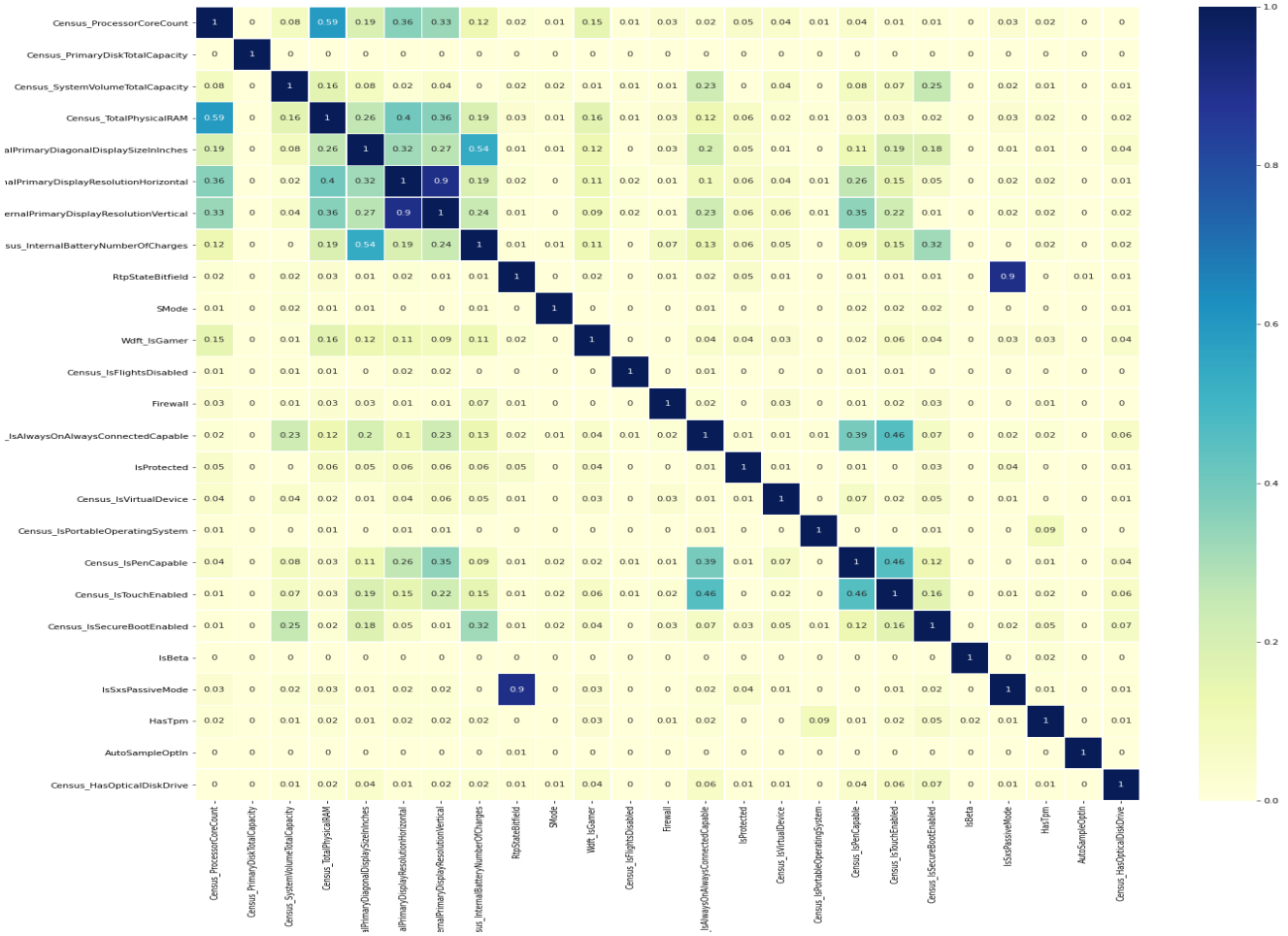


Fig 2.2 - Correlation Matrix

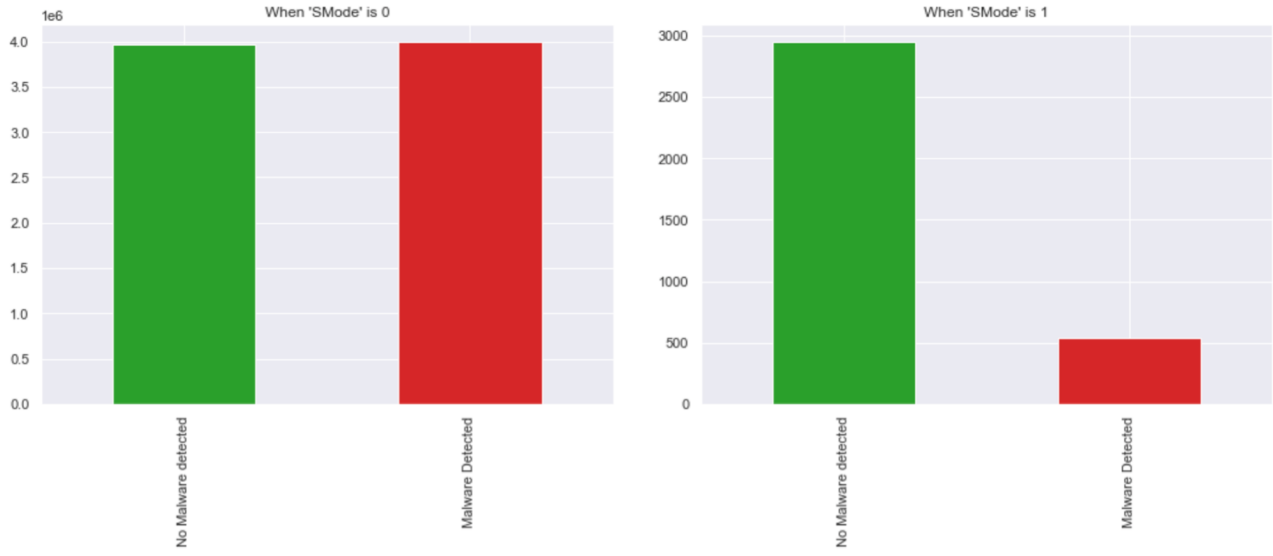
3. EXPLORATORY DATA ANALYSIS (EDA)

We are left with 72 features and nearly 8 million rows to understand the data and bring out some insights in a presentable format. As we know we can visualise all types of columns in the same way. I have decided to segregate features into 3 types, Boolean, Numeric and Categorical data. By doing so we left with 17 - Boolean columns, 37 - Categorical columns and 18 - Numerical columns.

3.1 Boolean Columns

As it has only 2 outputs and our target variable also has only 2 outputs. So, Came up with a visualisation technique where we first split using the target variable and then split on the boolean column to get insights. Let's see an example of column SMode which can be On(1) or Off(0). Smode functionality is that if we enable it, windows will only allow users to use applications from Microsoft Store.

SMode Vs HasDetections



SMode	No Malware detected	Malware Detected	Total of each category
0.0	3970036	3990340	7960376
1.0	2945	537	3482

SMode	% of Malware detected
0.0	50.127532
1.0	15.422171

Fig 3.1 - Boolean columns visualisation - SMode vs HasDetections

Shown in Fig 3.1, we will have a summary on each boolean column. We can interpret this as follows: When SMode is not enabled, Malware attacks are equally distributed informing that SMode - 0 is not affecting the target variable. But, then the image on right is telling a different story. We see a significant decrease in Malware Detected bar, To be precise it has fallen from 50 to 15%. This infers that SMode when enabled, reduces the chance of being attacked by malware. It is not that surprising given the fact that SMode only allows Microsoft Store applications which are secured.

In the same way I could bring out the following insights for Boolean columns :

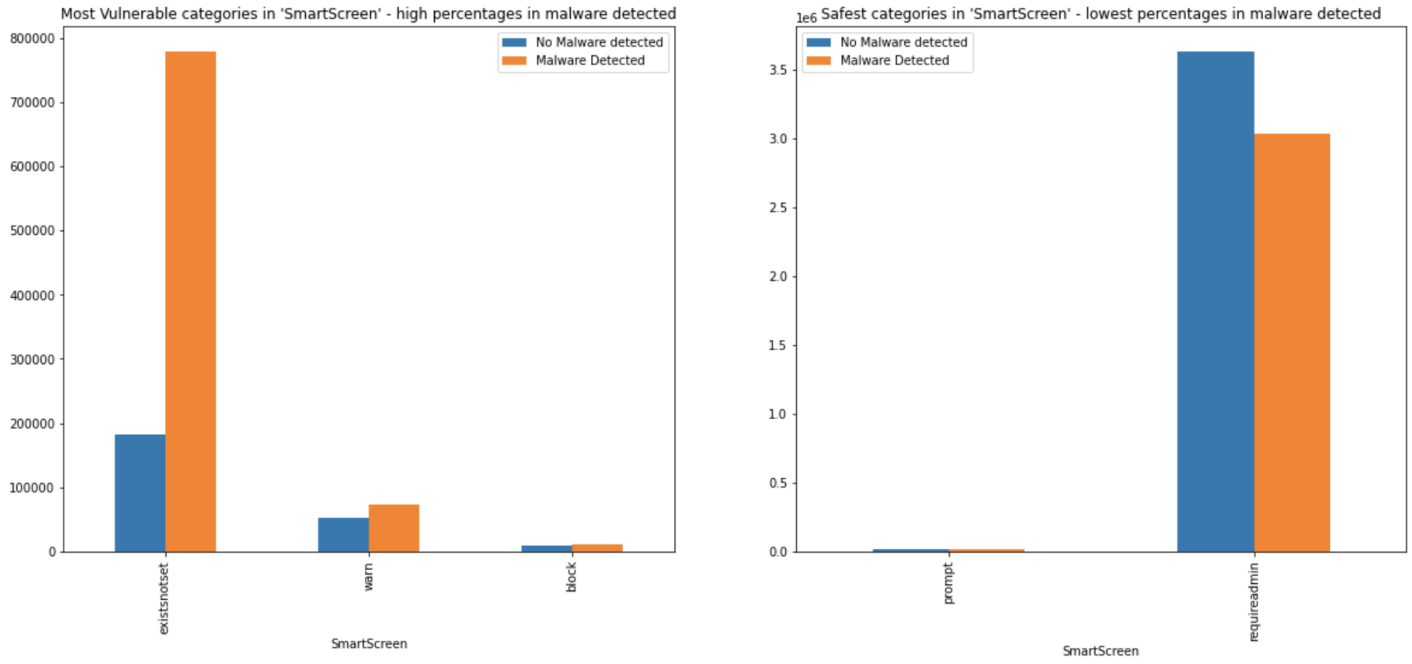
- When SMode is on (1), the chances of getting infected by a malware is reduced by 35 % (50 to 15%)
- When Census_IsFlights_Disabled then, only 5% is infected with malware. But the total number of systems that had the feature enabled is only 80 out of 8 million, it shows this feature is known to very few and used only by a handful of people. The reason for less malware attack % might be because the feature may not be known to the majority of the people.
- Enabling Census_IsAlwaysOnAlwaysConnectedCapable reduces chance of getting affected by a malware by 13% (50 to 37%)
- When a system is not protected (IsProtected) the chance of being affected by malware is less by 13% when compared to those systems that are protected. Ironically, this was the case but the scenario might be something like If a system is not equipped with malware protection software which detects some virus, one might not even be aware of all the virus that's present in one's system when compared to those which have malware protection software that checks for viruses and report once found.
- Machines running on virtual devices(Census_IsVirtualDevice) have only 19% of them being infected.
- When IsSXsPassiveMode is enabled, Malware detected percent came down from 50 to 37. It can be justified as Passive mode helps the system to run more than one antivirus program at a time. This multiple anti virus softwares makes sure the system is safe better and that explains the dip in malware detected systems.

- If the Is_Gamer column has value 1, then the malware detected percentage increased from 48 to 53. Stating that if we are gaming in our system there is slightly a higher chance of being attacked by malware when compared to non gamers.

3.2 Categorical Columns

The visualisation is similar to boolean but there are more than 2 categories unlike booleans. So, divided the categories into 2 types namely Most Vulnerable categories and Safest Categories in each column. Let me explain with an example. Take column "SmartScreen" which is a Microsoft Defender anti malware software and visualise it.

```
SmartScreen Vs HasDetections
```



SmartScreen	No Malware detected	Malware Detected	% of Malware detected
existsnotset	181909	779360	81.076161
warn	51934	73098	58.463433
block	9780	10936	52.790114
off	78217	79193	50.310018
prompt	15254	14051	47.947449
requireadmin	3635253	3033390	45.487365

Fig 3.2 - Categorical Columns visualisation - Smart Screen Example.

In the above figure we see 2 bar charts, one displays more vulnerable categories of column smart screen and other safest categories of smart screen. This split is done on malware

detection percentage. If the percentage is less than 48% it is added into safest and if more than 52, added into the Vulnerable graph. The inference here would be If SmartScreen is in “existsnotset” category then chance of being attacked by malware is more than 80% making that category one among the most vulnerable ones (from left chart) and if category is “requestadmin”, the percentage is 45% which is comparatively on the safer side (right chart).

Few inferences were made using this type of visualisation for all remaining categorical columns which are as follows:

- In feature Smart Screen if the status is “existsnotset”, the chance of a system being infected by a malware is as high as 81%. And “requestadmin” being the safest at 45% malware being detected.
- Systems in power platforms with role names “Slate”, “EnterpriseServer” and “AppliancePC” have less chance of vulnerability.
- Devices like Tablets,Servers and Detachables are less prone to malware when compared to Desktops,Notebooks and Convertibles.
- It looks mostly that with the number of battery charges increasing the system becomes less vulnerable to viruses.
- Census_ProcessorManufactureridentifier has very few entries in 3.0 and 10.0 versions but their attack percentage is less than 25% mainly version 10.0 with just 1% malware detected.
- Processor Core count 1 is most safest, followed by 2. Cores 12 and 16 are most vulnerable to malware.
- Anti virus products 3,4 and 5 having an average percentage of 27% of malware which being least and safe from malware. Version 1.0 with 55% making it most vulnerable. It explains because being the first version there will be many hacks which in turn made Anti virus teams to release new versions with more secured implementations.(AVProductsInstalled)
- If 2 or 3 anti virus products are enabled then the system is 70% safe from malware. (AVProductsEnabled).
- RtpStateBitfield which represents Real time transport protocol which helps in streaming audio and video over the internet and bitfield 8.0 is 76% vulnerable to virus and 5.0 is safest followed by 3.0.

- Census_OSBuildNumber and OSBuild contain almost the same information. OSBuild 17713,17672,17661,17682 are the most safest builds from malware. Alternatively, 7600 is most vulnerable.

Inorder to see all these visualisations you can visit the github report(mentioned in References and Links unit) and check for data visualisation and eda notebook in it.

3.3 Numeric columns

Next comes the numeric columns, for this I puzzled myself whether to check the bi variant or distribution but distribution of violin and histogram were helpful to see some insights.

Considering "AVTProductStatesIdentifier" column for example.

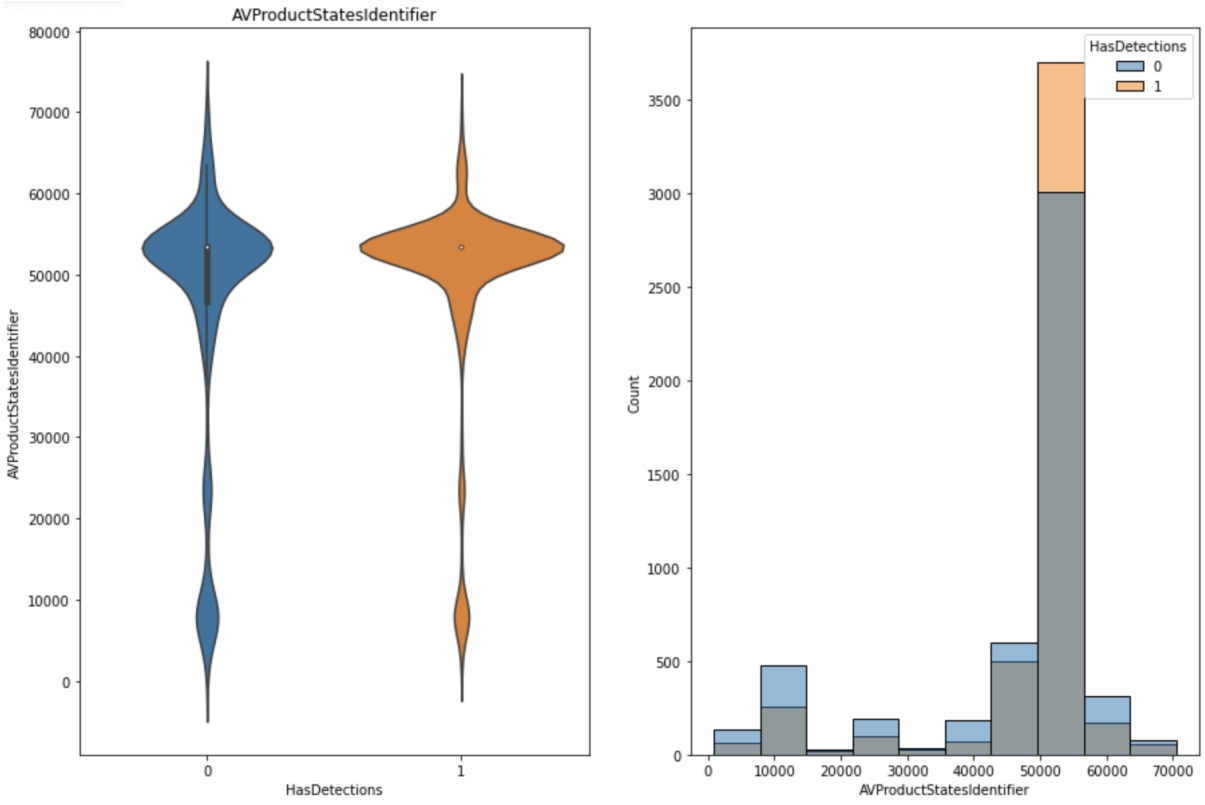


Fig 3.3 Numerical Columns visualisation - AVTProductStatesIdentifier Example.

In the above figure, We can see a violin chart that shows the distribution of AVTProductStatesIdentifier grouped over HasDetections. It can be inferred that almost the distribution looks the same but between 5000 and 6000 malware detected violin (Orange)

is wider than other. To confirm that we can see the right side graph where it clearly shows in the bin 5000-5500 there are more malware detected than not.

3.4 Spider charts

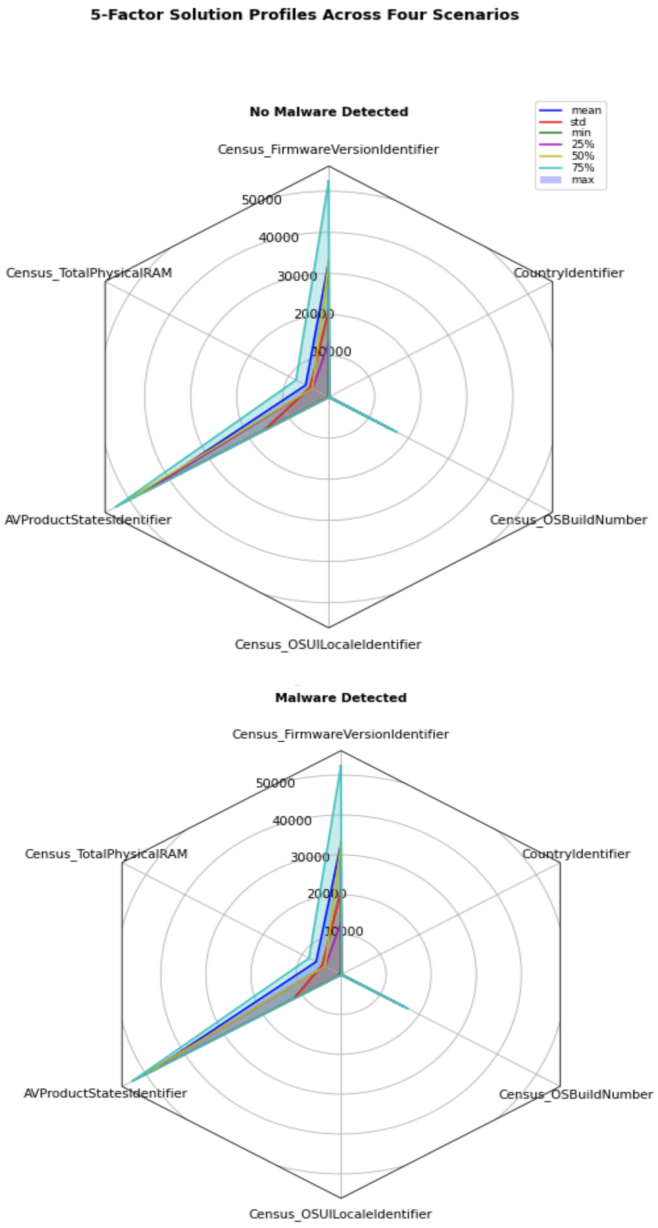


Fig 3.4 - Spider charts

Visualization confirmed that these numerical columns are affecting much on target variables as both the spider graphs look pretty similar on all statistical comparisons.

4. Data Modeling

With the knowledge of features and knowing the target, I first wanted to use Logistic Regression. To use it, I need to convert all features to numeric as most of the ML models won't take categorical variables. I tried one hot encoding but it gave 923 columns for training the logistic model. I was in search of a mechanism to convert categorical variables to numeric and also create fewer new columns. Landed onto a concept called "weights of evidences" which will actually give a numerical value for each category in a column according to its occurrence. I used this to convert all categorical columns into numeric so that dimensions (columns) will be very less. Trained Logistic model on previously mentioned dataset which is numeric and got an accuracy of 0.52.

This made me realise that Logistic Regression (drawing a line) won't work in this case of high dimensionality and then moved to the Decision tree to realise how time taking the process is. I have used the Random forest Decision tree model to train and test which fetched me 0.60 as accuracy.

I felt, this is progress and started searching for model that use Decision trees and stumbled upon Boosting Decision tree algorithms like Adaptive Boosting Decision tree (uses weights on dataset and brings of a strong learner from a set weak learners) and Gradient Boosting Decision tree (uses residuals/ errors of models as input for future models to make them strong). Read about the importance of LightGBM on large dataset and trained LightGBM to get an accuracy of 0.68 which is the highest till now.

In this process of reading about boosting techniques, I came across "CatBoost" that takes categories as inputs and XGboost a go to model for gradient boosting algorithms. They both gave an accuracy of 0.65 and 0.66 respectively on test data. Catboost was faster but XGboost took more time than all gradient models.

The following table will get an insight of models used and accuracies for each of them.It is understood that LightGBM was faster and more accurate among all the other models.

Model	Train Data Accuracy	Test Data Accuracy
Logistic Regression	0.525	0.525
Random Forest Decision tree	0.61	0.60
Light GBM	0.68	0.67
Cat boost	0.66	0.65
XGBoost	0.67	0.66

Fig 4.1 - Performance of Machine Learning models used.

But, we have a set of parameters to be provided to use LightGBM. This hyperparameter tuning needed to be done and made use of Bayesian optimiser hyper parameter tuning and got the best parameters for the LightGBM model. Using these parameters and iteration count for boosting, could reach to an accuracy of 0.70 on training data and 0.67 on test data with 0.77 as AUC (Area under the curve) for training data.

5. Test data submission

As we can see that lightGBM on parameters provided by Bayesian Optimiser parameter tuning provided highest accuracy for test data, decided to use this as the best model for test data provided to us to submit in a kaggle challenge. Running this model on the above mentioned data gave an output which is stored in a csv along with its MachineIdentifier column. This csv secured a public score of 0.639 where the competition highest score was 0.678.

6.Future Work

- Did use other models like catboost but didn't dig deep as we can better results in other models. Taking up from there looks good.
- I used CPU for modeling algorithms. We can extend it to GPU and check for any performance changes in them.

7. References and Links

- https://www.kaggle.com/c/microsoft-malware-prediction
- https://towardsdatascience.com/model-or-do-you-mean-weight-of-evidence-woe-and-information-value-iv-331499f6fc2
- https://lightgbm.readthedocs.io/en/latest/Parameters.html
- https://optuna.org
- https://github.com/fmfn/BayesianOptimization

You can see the work I have done on all stages of this data science project in the following Github Repo.

https://github.com/sankar95/Microsoft_Malware_Prediction