Microsoft Malware Prediction

Gurpreet Singh
2018csb1092@iitrpr.ac.in
Kanniganti Nagasrikar
2018csb1098@iitrpr.ac.in

Indian Institute of Technology, Ropar, Punjab, India

Abstract

We intend to find out if a given computer system has been/will be infected with a malware without scanning the personal files of a user. This is a Kaggle challenge posted by Microsoft a year ago.

Firstly, we do feature extraction and data exploration including graphical analysis. Then, we intend to use multiple classification algorithms and tabulate the performance. The effects of data wrangling are also studied.

1 Introduction

The main objective of this problem is to determine whether a computer system, given its hardware and software specifications and user habits, has been attacked by malware or not (output is boolean).

Most anti-viruses scan through users' personal files to see the presence of virus. However, users have growing concerns about their data being frequently scanned. What this method proposes to do is to be able to analyse through non-personal data such as versions of software and operating systems, and the amount of internet usage to predict if the computer is likely to be a victim of any malware.

Excessive use of internet, multiple file transfers and protection of confidential data makes malware detection and removal extremely important. Machine Learning and Data Science is proved to be very efficient in classification problems. Once we achieve good accuracy, the software can warn user that their usage habits are unsafe and with the user's permission, can conduct a deep scan.
Another significant advantage of this approach is that users will not need to perform repeated scans and that will improve the overall performance of their device.

Therefore, we have decided to pursue this project to determine if a malware attack has occurred/is about to occur on a system.

2 Literature Survey

Google Scholar was one of our two major sources to research about previously existing works done in this field. We found that several attempts have been made in this field both with and without machine learning. However, this problem statement differs from others in the nature of classification. Other works attempt to classify executable files using available data as malicious or benign. Those papers utilized techniques like GNB classifier, Support Vector Machines, Logistic Regression, Decision Trees and many other improved forms of existing algorithms. In contrast to it, we are using data associated with a computer extracted by Windows Defender. Our aim is to predict malware attack on a computer in future rather than classifying files.
However, a lot of solutions of this problem statement are available on Kaggle. In addition to these, we found articles available on https://towardsdatascience.com and https://medium.com very helpful. All these artcles contain many data exploration and memory saving techniques. They gave us the idea of using efficient data types and feature engineering.

All the specific resources we found useful are mentioned at the end of this document under the 'References' header.

3 Dataset

Dataset consists of eighty two columns, each indicating a different measure of the system security or user habits. This data was generated by combining heartbeat and threat reports collected by Microsoft's endpoint protection solution, Windows Defender. It contains 7,853,253 unique entries.

The sampling methodology used to create this dataset was designed to meet certain business constraints, both in regards to user privacy as well as the time period during which the machine was running. This means this dataset does not contain any personal information of the user. We are supposed to look at only the configuration, protection and usage habits to determine the likelihood of a virus. The data has 81 features, with 52 being categorical, 23 of which are encoded numerically to protect the privacy of the information.

The dataset can be found at the following link:

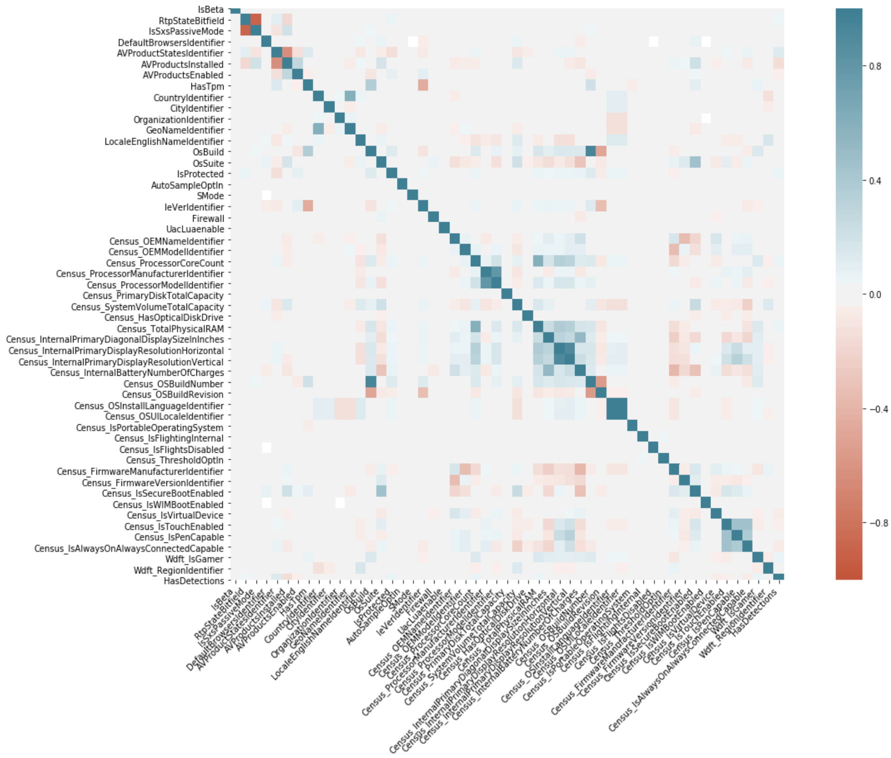
https://www.kaggle.com/c/microsoft-malware-prediction/data

4 Method

4.1 Data Cleansing

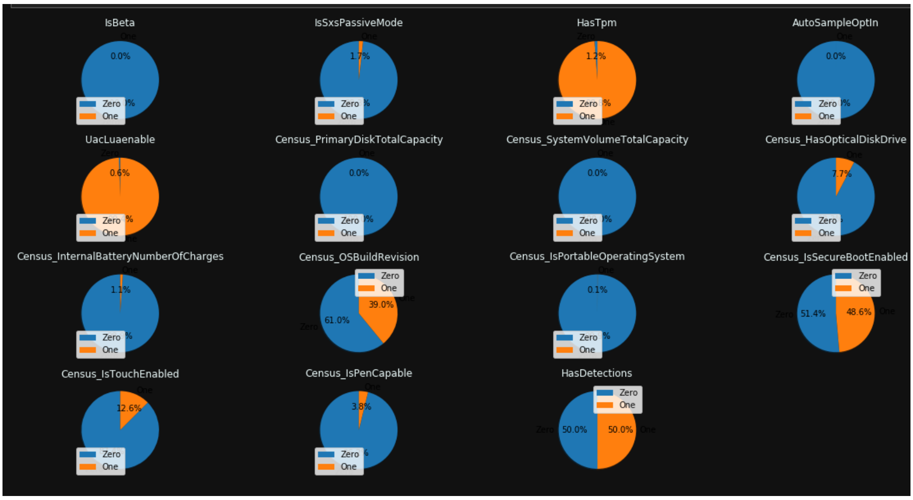
The dataset is collected over a varying timeframe and due to multiple constraints while collecting data (including protecting user privacy), there are a lot of missing values.

Here is a plot of the correlation matrix:



Any column where more than 30% percent values are missing has been dropped and the rest are filled with a median value. Median was chosen because most of the data was categorical and hence, is a better choice than mean. We have also looked at other approaches such as removing all the rows with any missing values. This compressed the dataset by about 20% but the data is much cleaner and easier to evaluate.

Another major issue with the data was it's enormous memory requirements. The training data was over 4 GB in size. To make it run quicker and use less RAM, we converted the data types of most columns to something more compact. For example, all the columns holding integers were initially 64 bit integers. Checking the min and max values that occur in a particular column, we have changed their data types to 8/16/32 bit integers for faster processing. Similarly, all float data types have been converted from float64 to float 32 or float 16. This had the surprising effect of decreasing the memory usage by nearly 70%!
Considering issues like limited internet availability, limited memory and processing resources, we decided to work on a smaller dataset consisting ten thousand rows and eighty three features. Three columns had more than 95% missing values. Moreover, some columns had almost all of their data belonging to same category. We decided to omit such features from our analysis. These features were: 'DefaultBrowsersIdentifier', 'OrganizationIdentifier', 'SmartScreen', 'Census_ProcessorClass', 'Census_InternalBatteryType', 'Census_IsFlightingInternal', 'Census_ThresholdOptIn', 'Census_IsWIMBootEnabled', 'PuaMode'. After removing rows with missing values, we were left with dataset consisting 4988 rows and 56 columns. Furthermore, aforementioned efficient assignment of data types lead to 70% reduction of size of pandas dataframe from 6.33Mb to 1.9Mb.



4.2 Data Wrangling

Most of the data was categorical in nature. To make it easy for algorithms to make sense out of this data, it was needed to be converted this categorical data to other forms. We have used one-hot encoding for most of the categorical columns. We had to manually go through all columns and select which could be appropriately be represented by one hot encoding. We also removed columns in which the number of columns was comparable to number of samples as they add no useful information to data. Note that the data was not ordinal and hence the order of values also contain no information.

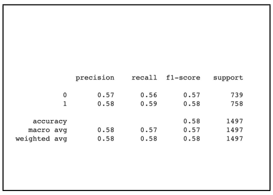
After this process, we ended up with 193 columns (increased due to one-hot encoding)

4.3 Classification Algorithms

Before Applying any classification Algorithms, we had to significantly reduce the dimensionality for optimized performance. Hence, we applied Pricipal Component analysis and reduced the dimensionality to 13 for Logistic Regression and Gaussian Naive Bayes. This retained 99.9% variance. As SVM was expected to take more time, three principal components were chosen for it. These three accounted for 99.7% of the total explained variance.
The performance achieved was in the order LR>GNB>SVM.
The run time of LR and GNB was almost similar, but SVM took exponentially large time compared to the other two.
In case of LR, the F1 scores of both the classes are almost identical. On the other hand, GNB and SVM identify class 1 better.

5 Experiments and Conclusion

Considering that the dataset is not highly skewed towards any of the classes, accuracy should be a good measure of performance of the algorithm. As it can be seen, the accuracy of of all the algorithms is close to 55%.
Checking other literature on this problem, given the resource constraints, this value of accuracy is satisfying enough. The $25,000 award winning solution on Kaggle had an accuracy

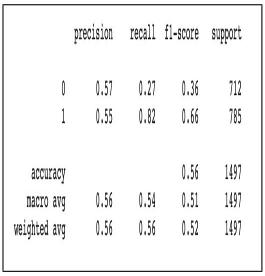


```
              precision    recall  f1-score   support

           0       0.57      0.56      0.57       739
           1       0.58      0.59      0.58       758

    accuracy                           0.58      1497
   macro avg       0.58      0.57      0.57      1497
weighted avg       0.58      0.58      0.58      1497
```

(a) LR

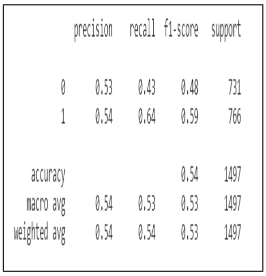


```
              precision    recall  f1-score   support

           0       0.57      0.27      0.36       712
           1       0.55      0.82      0.66       785

    accuracy                           0.56      1497
   macro avg       0.56      0.54      0.51      1497
weighted avg       0.56      0.56      0.52      1497
```

(b) GNB



```
              precision    recall  f1-score   support

           0       0.53      0.43      0.48       731
           1       0.54      0.64      0.59       766

    accuracy                           0.54      1497
   macro avg       0.54      0.53      0.53      1497
weighted avg       0.54      0.54      0.53      1497
```

(c) SVM

Figure 1: Classification Reports

of about 66%. Considering that we're are trying to predict the malware presence based just on user habits, this is very exciting!
The Classification reports of all algorithms are given above.

References

[1] Link1

[2] Link2

[3] Link3

[4] Link4

[5] Link5