Springboard - Data Science Track
Jacqueline (Xuan) Guo

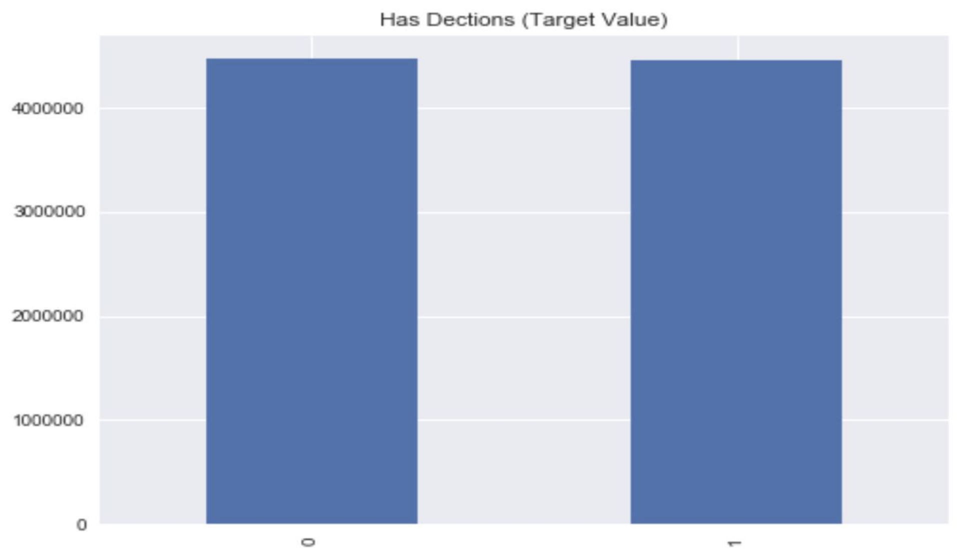
Capstone Project 2 - Final Report

Title:
Microsoft Malware Prediction (Kaggle Link)

Introduction:
Right now we are in a computer based society. Microsoft Windows is one of the most popular computers for people to use. However, once a computer is infected by malware, criminals can hurt consumers and enterprises in many ways. Thus we could predict if the computer will soon be hit with malware, we can reduce the risk of malware infection.
Computer services companies such as Microsoft or any windows users would like to apply this prediction to their system to keep their user's using experience safe. Most people or companies who use computers would care as well, for example, institutions like financial service companies or banks would love to keep their company information safe.

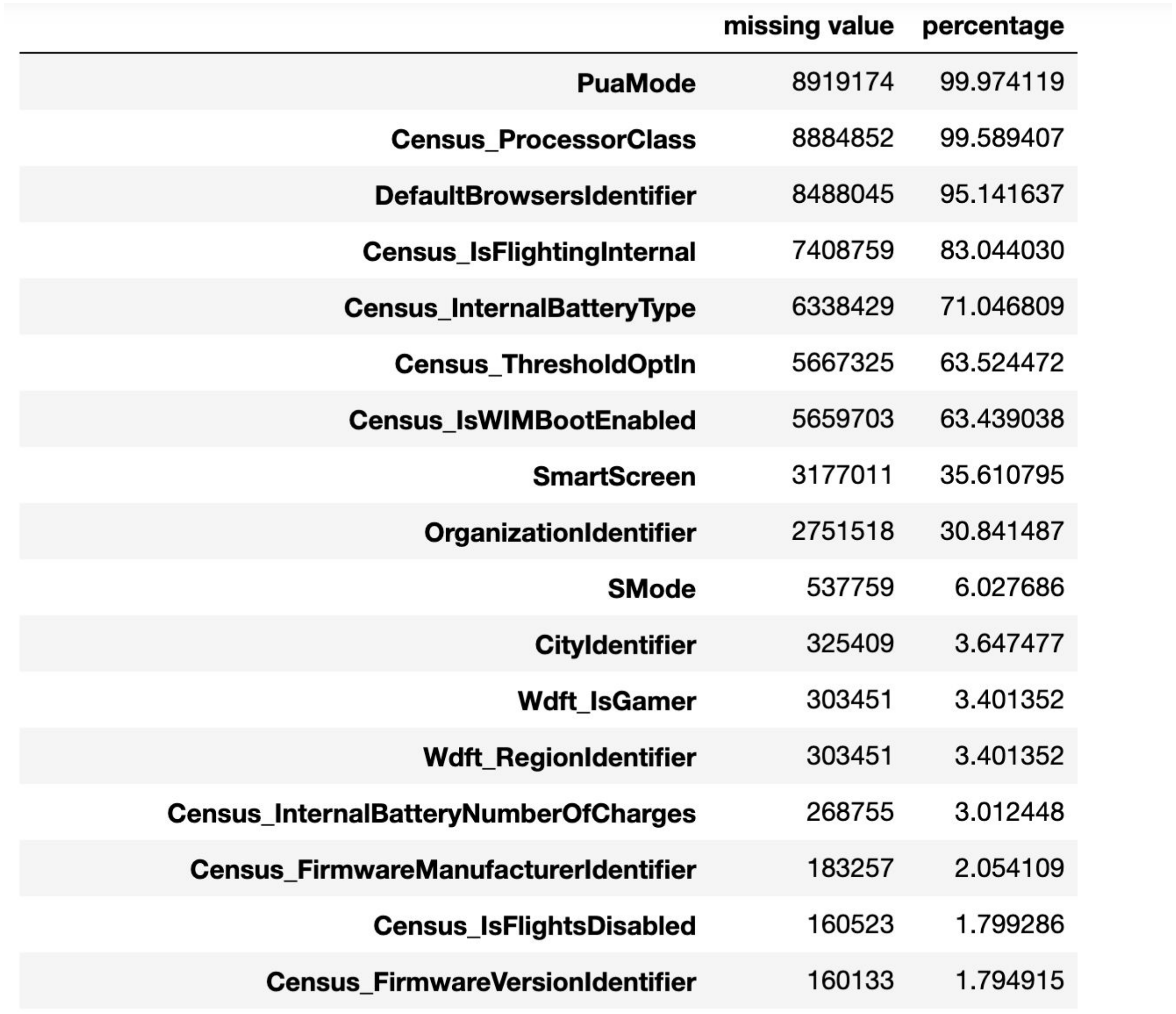
Dataset:
Dataset is from Microsoft of Kaggle competition. The size of the data is 7.89 GB with 167 columns and 8921483 rows. Microsoft provides one training dataset and one testing dataset which include the information about product name, different system versions, or different engine versions. This information could help us to predict a Windows machine's probability of getting infected by various families of malware, based on different properties of that machine.

Preprocessing:
My second capstone project is available to view at: Capstone Project 2. First, I've checked if our dataset has a balanced target value distribution. I've drawn a histogram as following:



As the graph states, our labeled value is pretty balanced.

Then I want to check if there are a lot of missing values in our dataset:

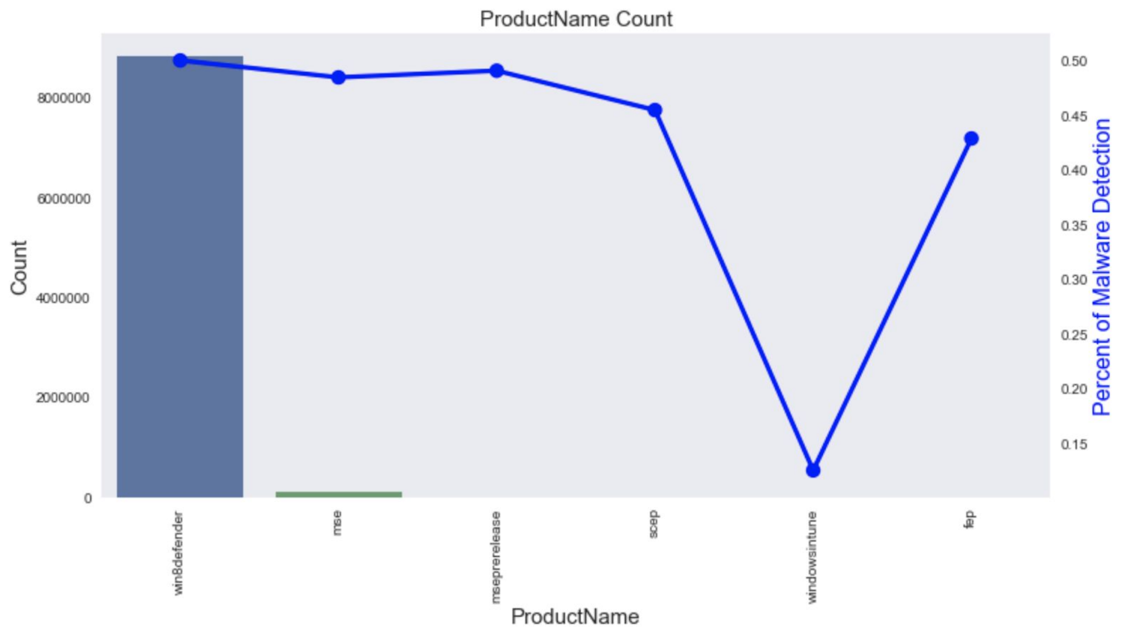


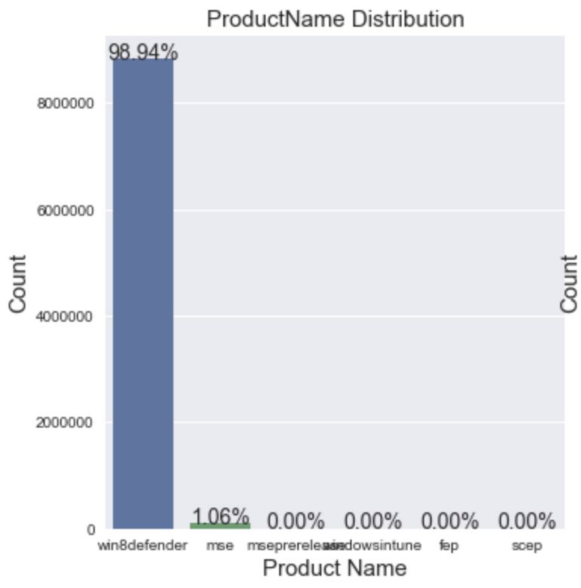
	missing value	percentage
PuaMode	8919174	99.974119
Census_ProcessorClass	8884852	99.589407
DefaultBrowsersIdentifier	8488045	95.141637
Census_IsFlightingInternal	7408759	83.044030
Census_InternalBatteryType	6338429	71.046809
Census_ThresholdOptIn	5667325	63.524472
Census_IsWIMBootEnabled	5659703	63.439038
SmartScreen	3177011	35.610795
OrganizationIdentifier	2751518	30.841487
SMode	537759	6.027686
CityIdentifier	325409	3.647477
Wdft_IsGamer	303451	3.401352
Wdft_RegionIdentifier	303451	3.401352
Census_InternalBatteryNumberOfCharges	268755	3.012448
Census_FirmwareManufacturerIdentifier	183257	2.054109
Census_IsFlightsDisabled	160523	1.799286
Census_FirmwareVersionIdentifier	160133	1.794915

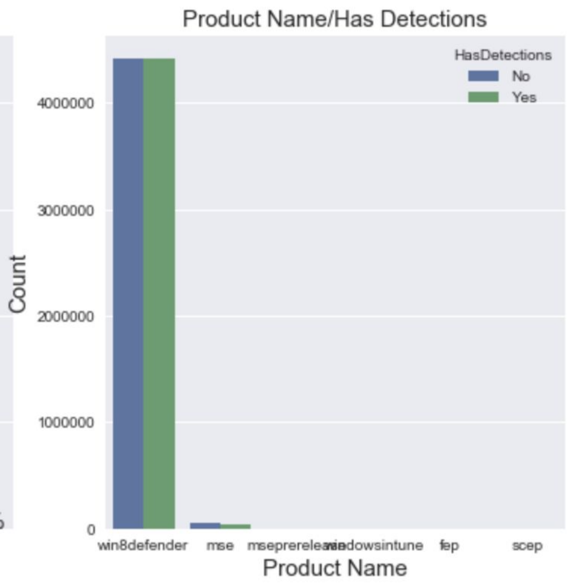
I found that I could drop four features by analysis which include: 'DefaultBrowsersIdentifier', 'PuaMode', 'Census_IsFlightingInternal', and 'Census_InternalBatteryType'.

Next step is to do the Exploratory Analysis. First, let’s focus on categorical features.

I. Product Name

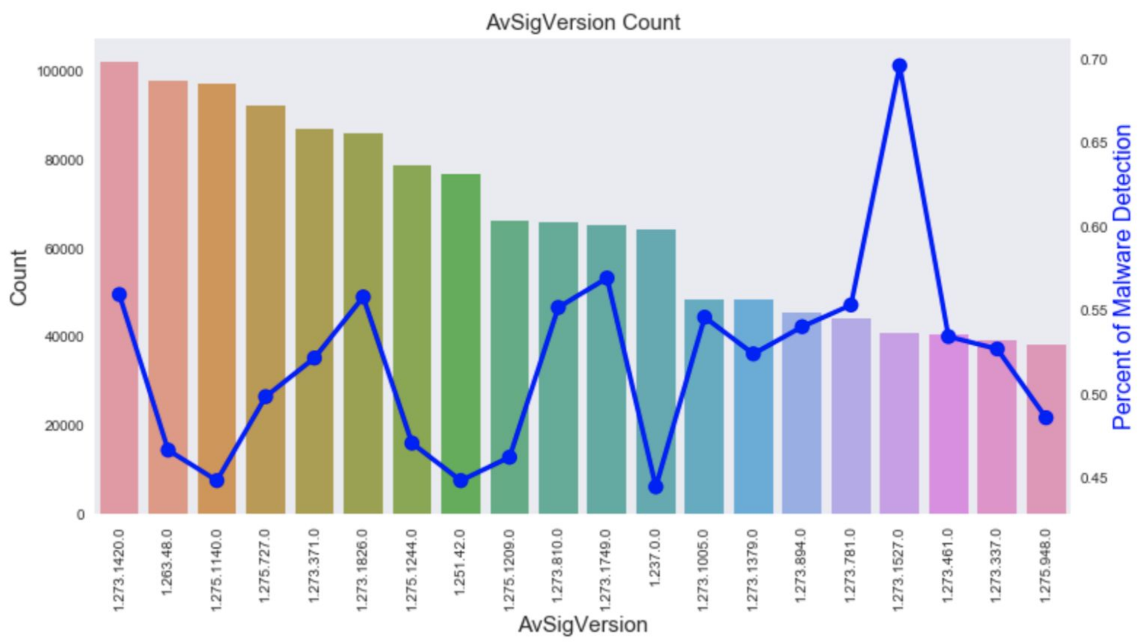






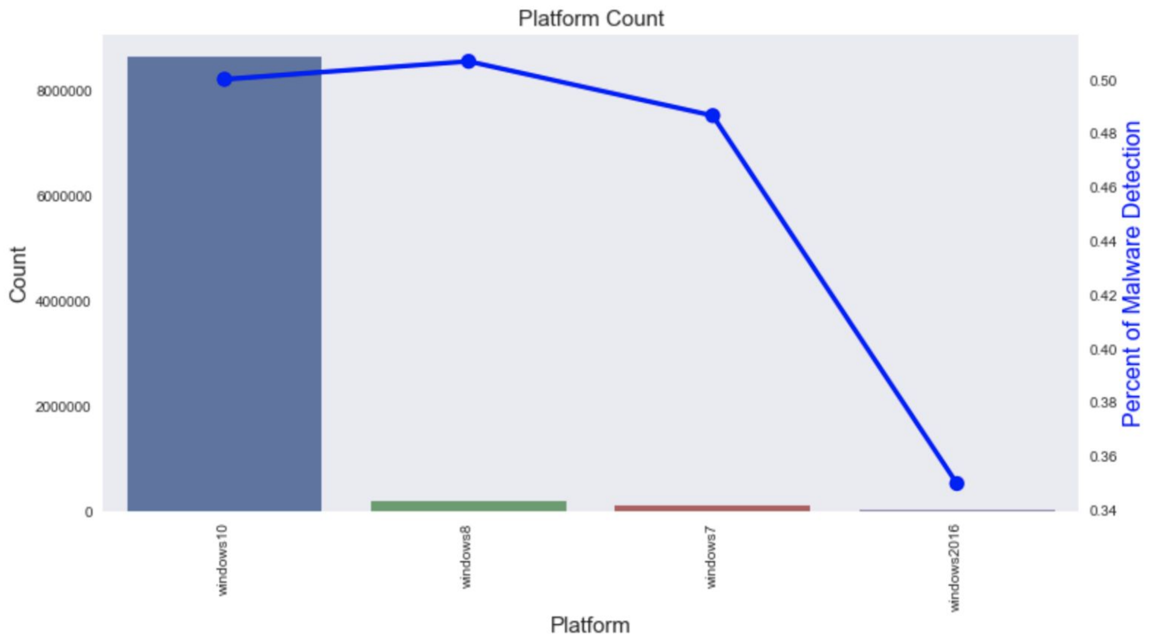
And above graphs we can see that windows8defender is the most common product and it has detection rate of around 50 %. We can also see that windowsintune has a very low detection rate. It's possible that malware is less likely to be detected in windowsintune.

II. AvSigVersion



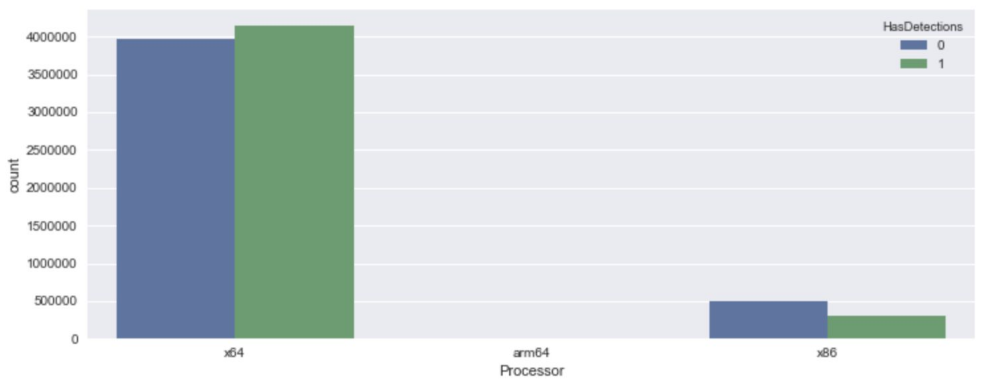
According to the graph, we can notice that there's an unusual peak of detection rate at version '1273.337'. Thus we should pay more attention to this feature.

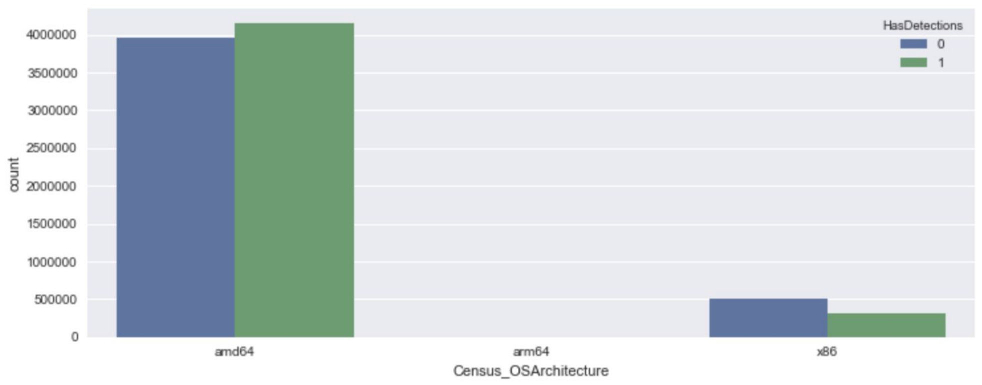
III. Platform



We can see that windows 10 is the most common platform and it has a detection rate of around 50%. And windows 2016 has a low detection rate.

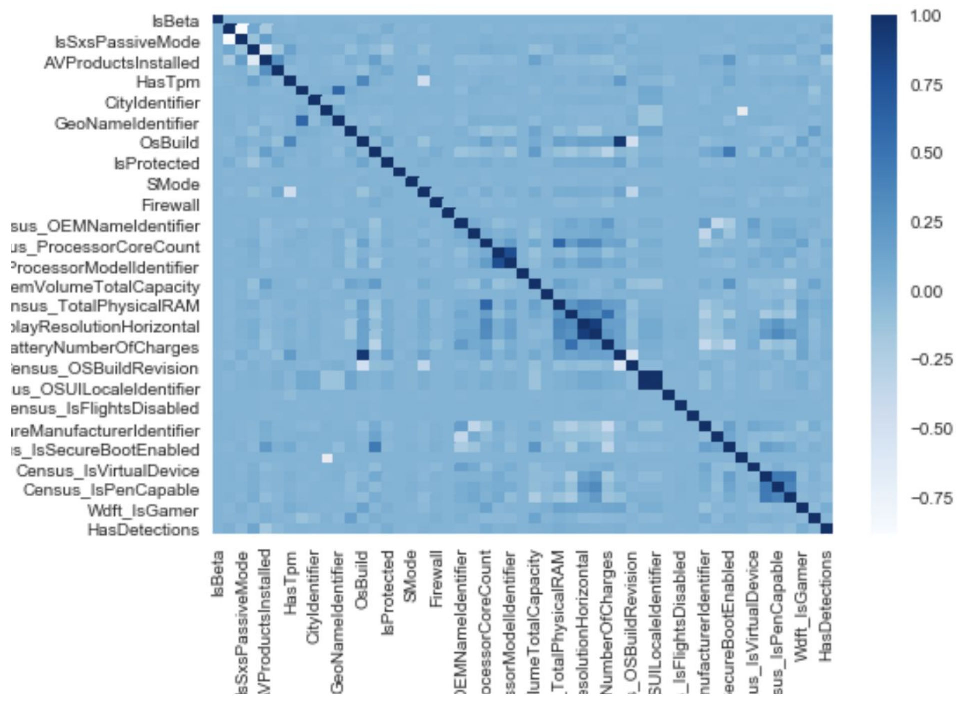
IV. Processor and Census OSArchitecture





We can see that for x86 processors, the detection rate is lower than other processors.

I draw a heatmap to measure the correlations between numerical features as following:



We could see from the heatmap that some of the features highly related to each other, thus we could apply feature selection to produce higher accuracy when fitting the model.

Limitations:

I. **Unknown Data**

I found some of the features are hard to analyze because we could not know the feature's meaning due to security reasons. Also, there are some features that have very complicated expressions such as 'MachineIdentifier' that contains a series of combined numbers, letters, and special characters. I removed those features to make it easy to fit in the models.

II. **Outliers and Missing Data**

There are lots of outliers and missing data in our data set. What I did is remove some features that are missing over than 90% of data, leave some of the missing data as missing, or replace the missing value with median value.

Feature Engineering and Models:

I. **Objective:**

Utilize supervised learning techniques to build predictive models for the malware detection data. So far I have gone through the data, did some exploratory analysis on the dataset. With the understanding gained, the next step is feature engineering, selecting models, and evaluating models.

II. **Feature Engineering**

Based on exploratory data analysis, here are the feature engineering I did (aside from minor data cleaning such as lowercase all device info):

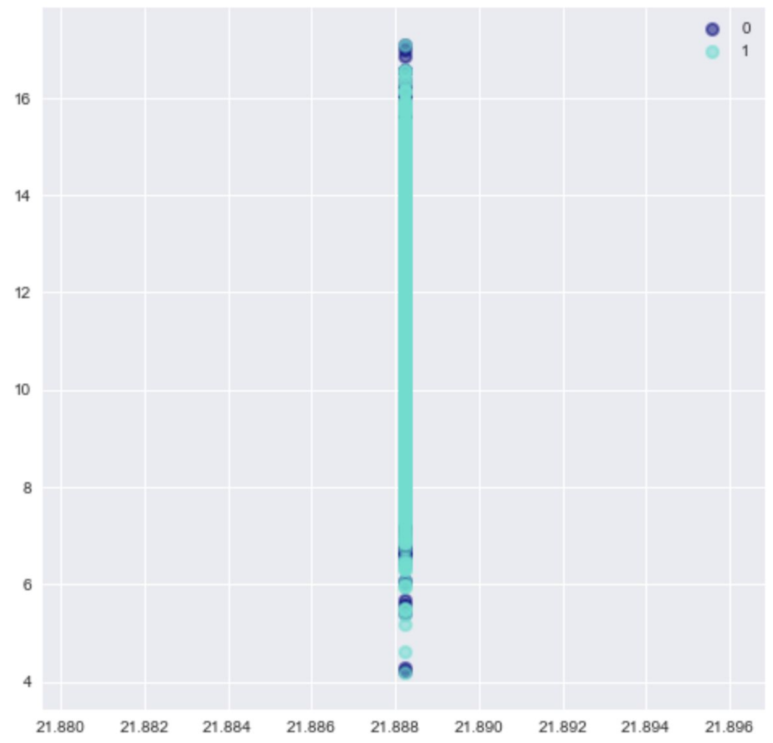
- ❏ Removed some features that have lots of missing data.
- ❏ For other features which only lack a few data, I filled missing values with the median.
- ❏ Converted categorical values into a numerical representation by Pandas method-get dummies.
- ❏ Applied PCA to reduce feature dimension to visualize the target value.
- ❏ Used Scikit Learn to do feature selection which selected 50 out of 277 features to fit in the model.

III. Model Selection

I have selected three models to perform malware detection prediction which are: Logistic Regression, Random Forest, and Decision Tree.

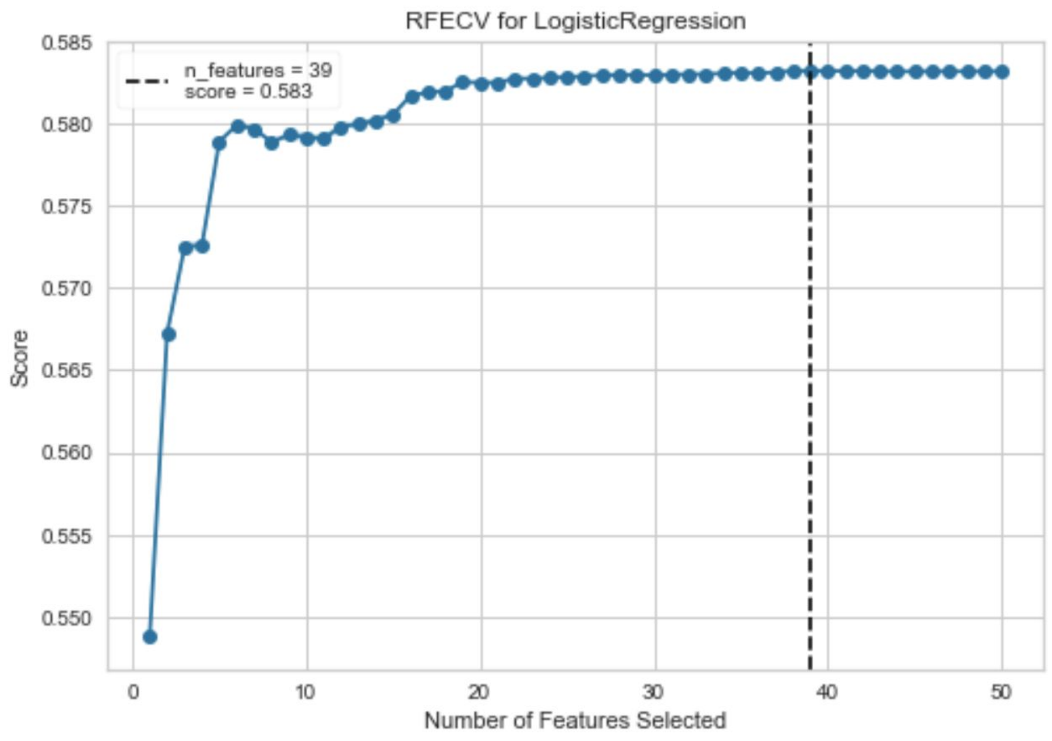
1. Logistic Regression

   Logistic regression is a statistical model that in its basic form uses a logistic function to model a binary dependent variable. After preprocessing, I've applied Logistic Regression using Scikit Learn. I got an accuracy score of 0.49 which is not a good prediction. This means our model only correctly predicted half of the result. Then I used PCA to reduce the dimension of our features to visualize the distribution of test set target value as below:



   My intention is to see if there's a clear distinction between detections and nondetections. However, the target values are overlapped together which probably means we lost feature information during dimensional reduction.
   On the other hand, I've normalized our data to avoid if there are some features that have more weight than others. After fitting the model, the accuracy score improved to 0.59 which is higher than the first time.

Furthermore, I’ve also applied feature selection to select 50 features out of 227. Then the accuracy score of Logistic Regression is increased to 0.61. I still want to know if I should abandon more, so I’ve used RFECV from Scikit Learn to draw a selected feature graph as following:



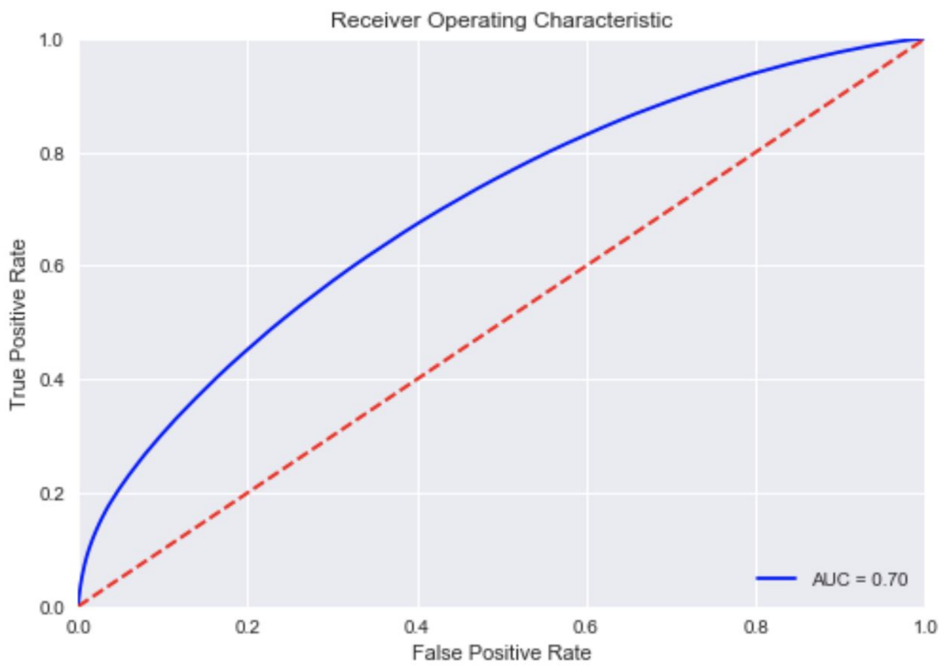
However, I found the accuracy score is approximately the same.

2. Decision Tree

   Decision Tree Learning is one of the predictive modeling approaches used in statistics, and it uses a tree-like model of decisions to category our data. The original roc score of Decision Tree is 0.57. After feature selection and normalization, the roc score is 0.549.

3. Random Forest

   The Random Forest is a classification algorithm consisting of many decision trees to avoid overfitting. Before feature selection, the roc score of Random Forest is 0.6958. After feature selection and normalization, the roc score is 0.6960. Apparently, feature selection did not help to improve the prediction for Random Forest. The ROC curve as following:



The combined model result as following:

	Accuracy Score	A-score after normalization	A-score w/ feature selection	ROC score
Logistic Regression	0.499	-	0.618	-
Random Forest	-	0.696 (ROC)	-	0.696
Decision Tree	-	0.549	-	0.549

Discuss and Conclusion:

In conclusion, I've tried three models to find the best model for our prediction. According to the accuracy score of each model, we can conclude that Random Forest is the best model to use in this problem because it has a 0.696 roc score. The prediction result is much more accurate of the random forest model compared to others. I've also found that sometimes 2 dimensional visualization cannot clearly visualize the pattern between target values. Moreover, we could apply feature selection to improve our model's performance and normalizing our data could make the result more accurate.

In this capstone, I spent most of the time on improving the model performance. I've learned a new way to visualize the performance of feature selection. I've also learned more about the

Logistic Regression and how to improve model performance by feature selection and feature normalization.

Future:

This Kaggle competition data set has lots of features of Microsoft malware detection. To make the prediction more accurate, sometimes we should abandon some weak features and only use features that are strongly related to malware detection rate. However, I've only applied the Scikit Learn feature selection package. For further prediction, I think I should go back to some of the removed features, and try to learn more about their meaning and relationship between detection rate. I've improved the accuracy score of Logistic Regression from 0.49 to 0.61, but I believe there is still some way to increase the accuracy better. I need to learn more about Machine Learning knowledge to support the better model performance.

Deliverables:

1. Code notebooks
2. Report on the capstone project
3. Presentation or poster on the capstone project: PPT