Capstone Project

MICROSOFT MALWARE PREDICTION CHALLENGE

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.

Microsoft Malware Prediction Challenge

- Microsoft has issued a challenge to use a malware dataset to do the following:
- Use historical data of machines with different properties and malware rate and gain data insight on those properties.
- Predict a Windows machine’s probability of getting infected by various families of malware based on the properties of that machine.

Data source

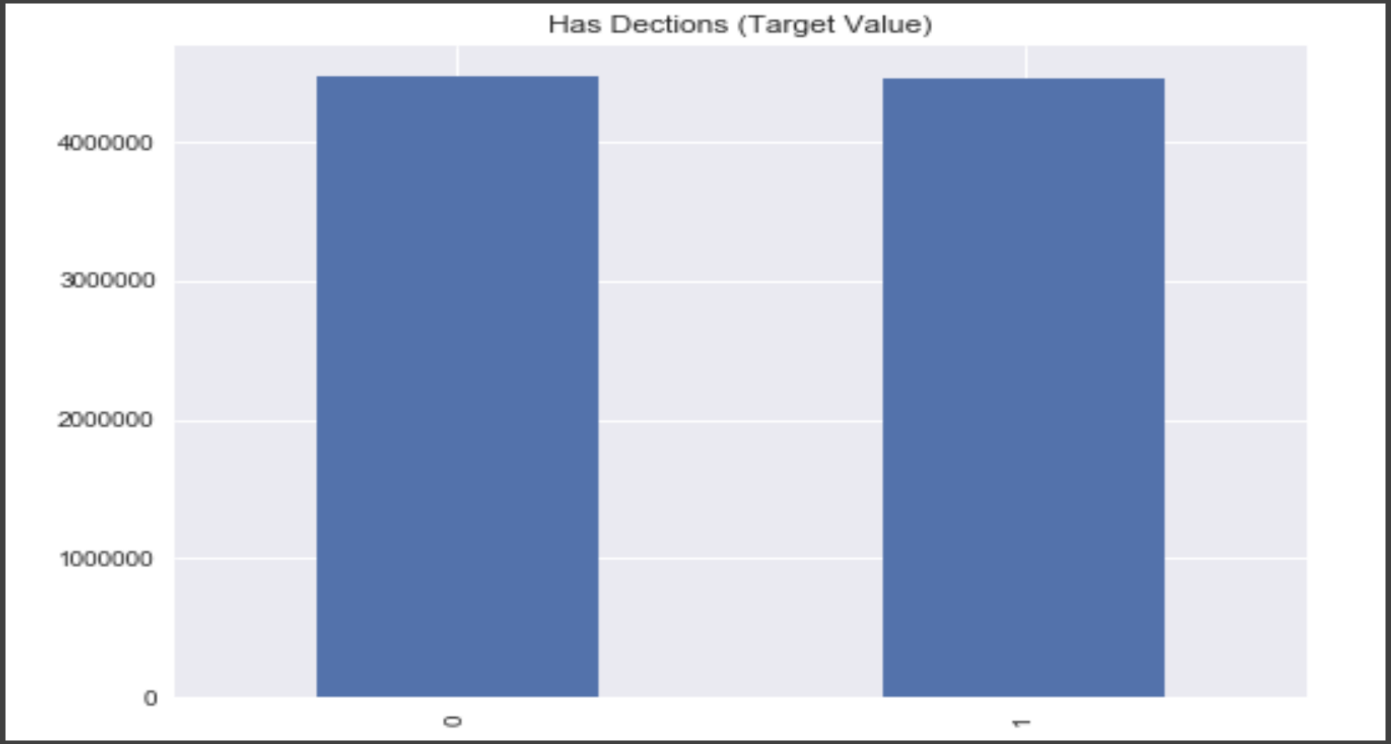
- Dataset is from Microsoft of Kaggle competition. The size of the data is 7.89 GB with 167 columns and 8921483 rows.
- Microsoft provides one training dataset and one testing dataset which include the information about product name, different system versions, or different engine versions.
- This information 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.

Data – First Glance

- The first steps in any project involve formatting the data as necessary
- In raw data, There are 8,921,483 observations of 83 features:
  - Number of numerical data: 53
  - Number of Categorical data: 30

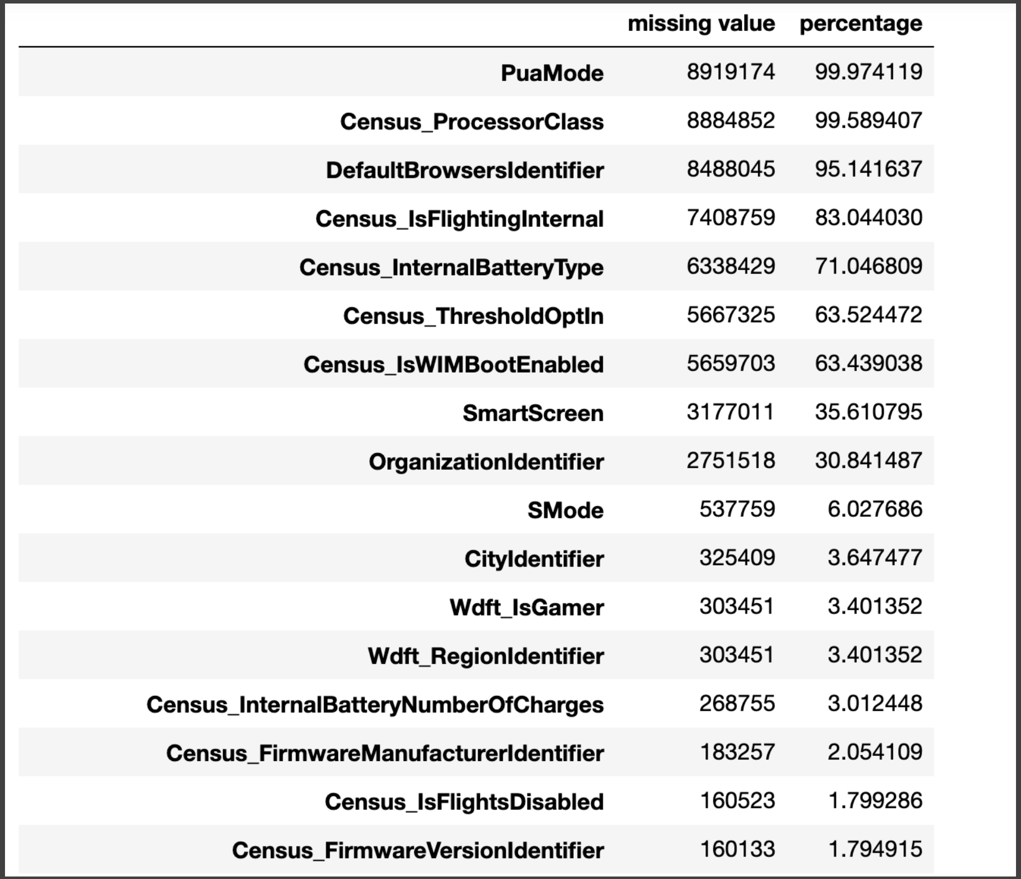
Data – First Glace Continued

- The label is balanced, which is nice:



Data Preprocessing

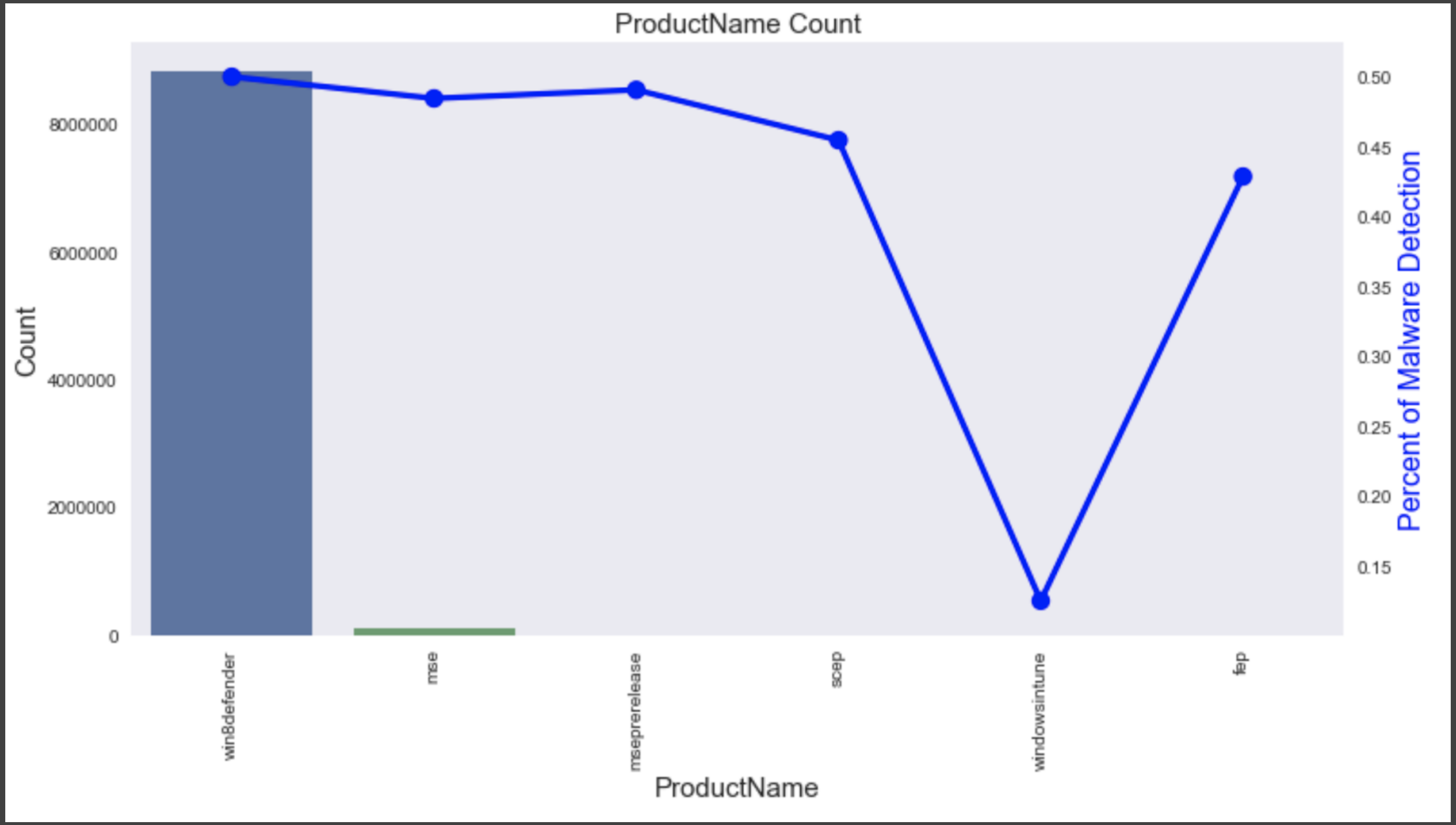
- I then checked missing values in the data
- There are four columns have over 83% missing values
  - PuaMode, Census_ProcessorClass, DefaultBrowsersIdentifier and Census_IsFlightingInternal
- I find them have too much missing values and very little unique values.
- So I think they are useless and dropped them



	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

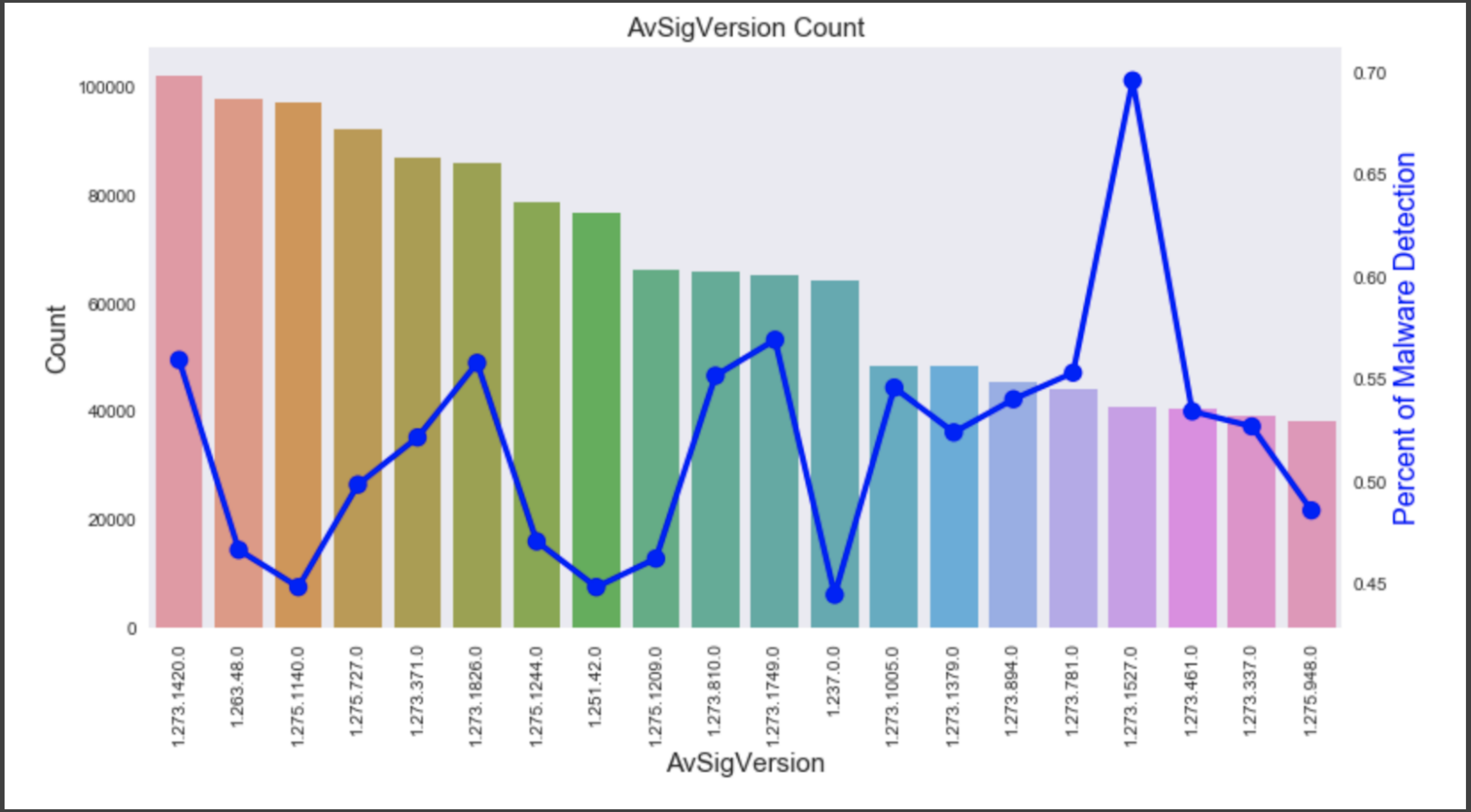
Data Preprocessing Continued

- I then examined the features that also have high missing value percentage from previous graph and there are plenty of features that have only one or very low number of unique features other than missing value, so I dropped them as well
- There are 26 columns in total in which one category contains 90% values. I think that these imbalanced columns should be removed from the dataset



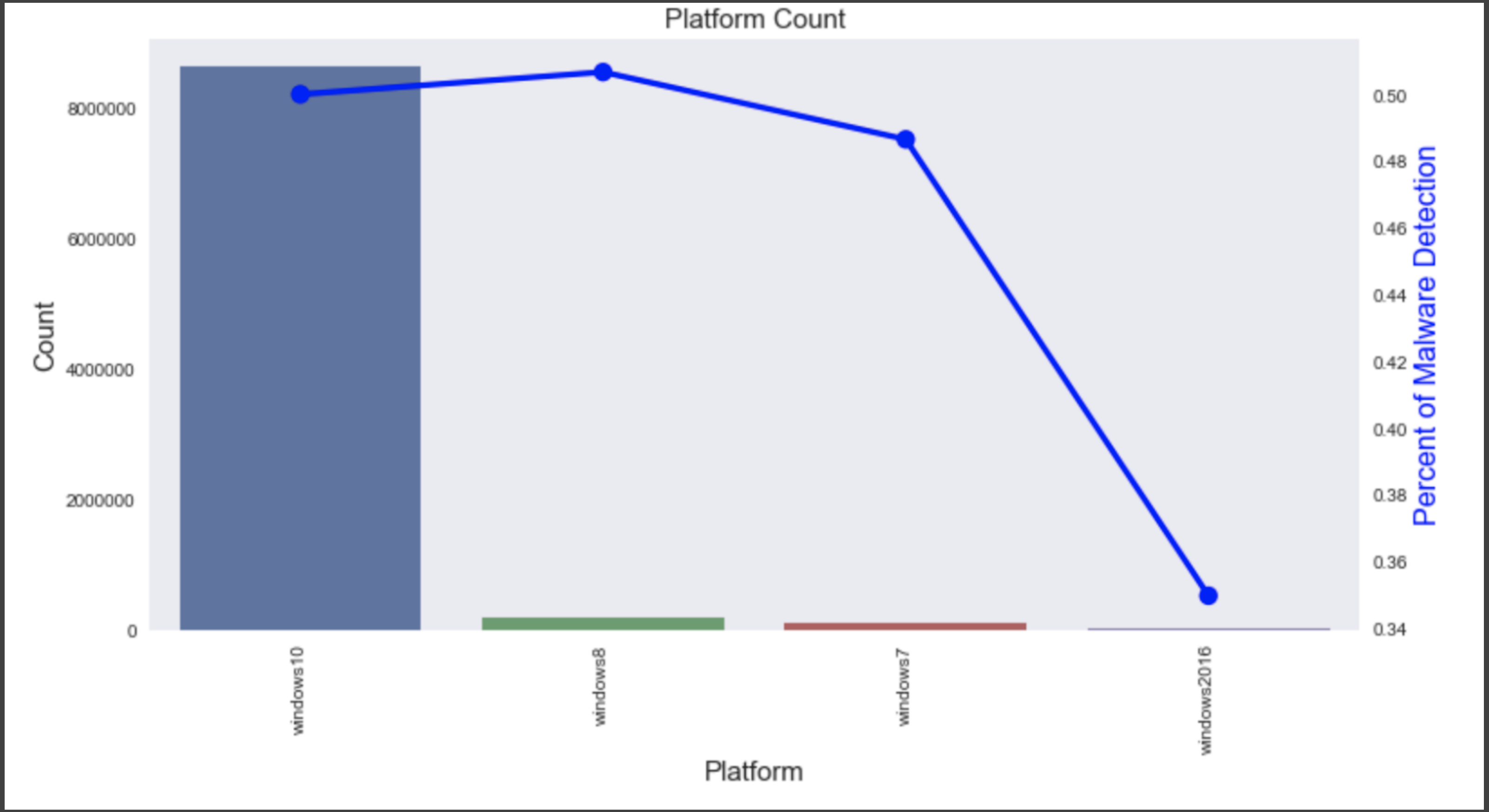
Data – Impressions

- 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.



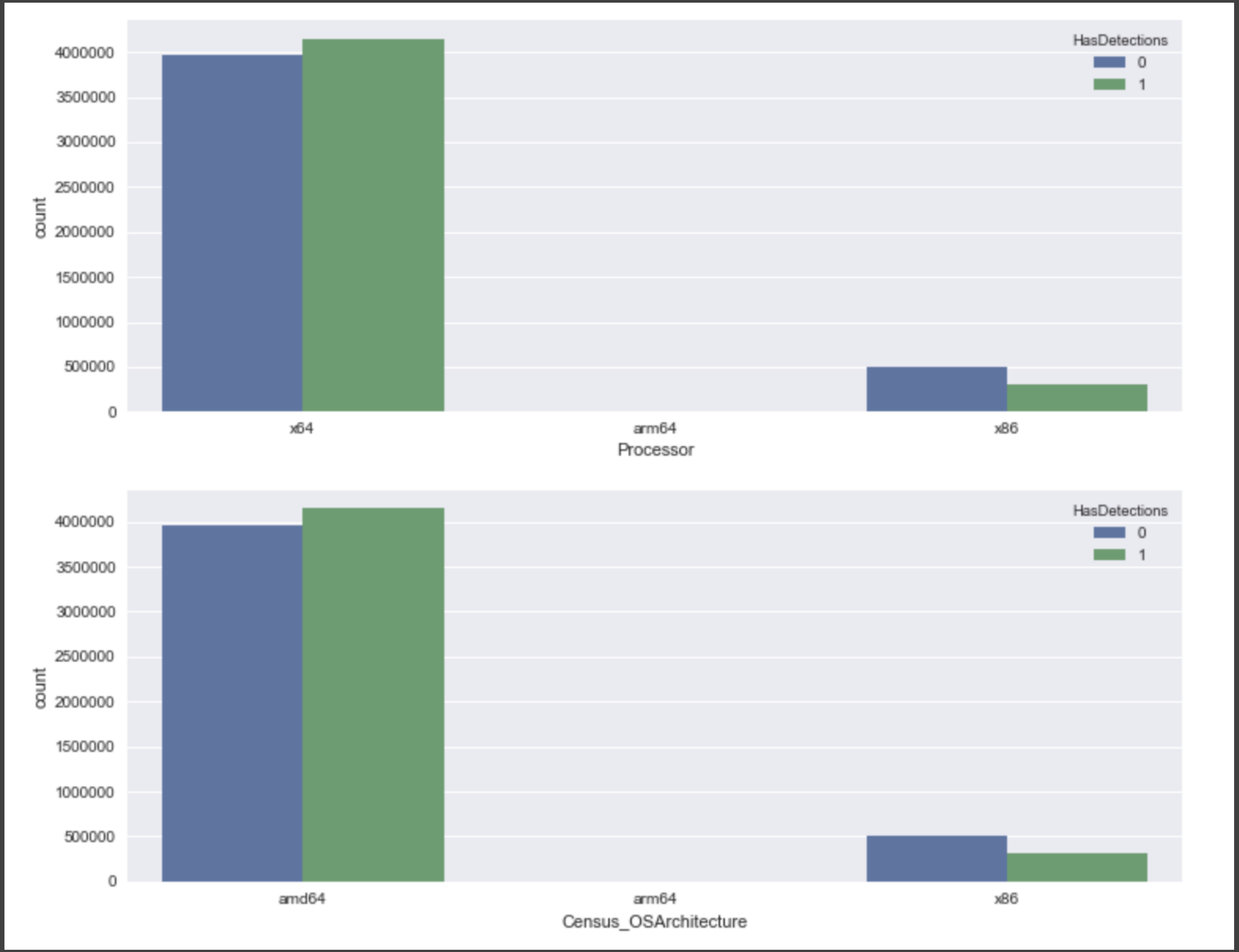
Data – Impressions

- 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.



Data – Impressions

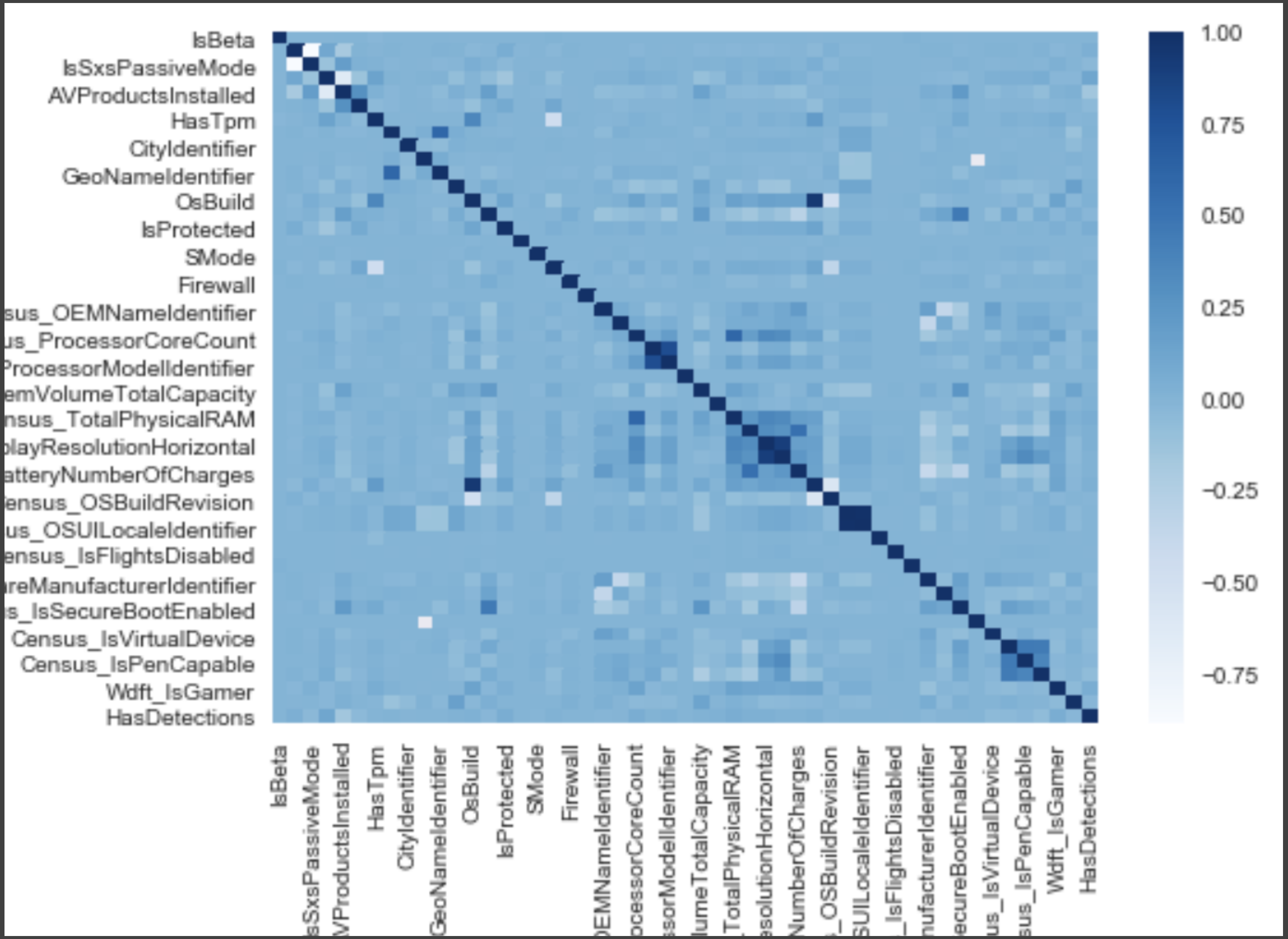
- We can see that windows 10 is the most common platform and it has a detection rate of around 50%. And windows 2016 has a low detection rate.



Data – Impressions

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

Data – Impressions



- I draw a heatmap to measure the correlations between numerical features
- 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.

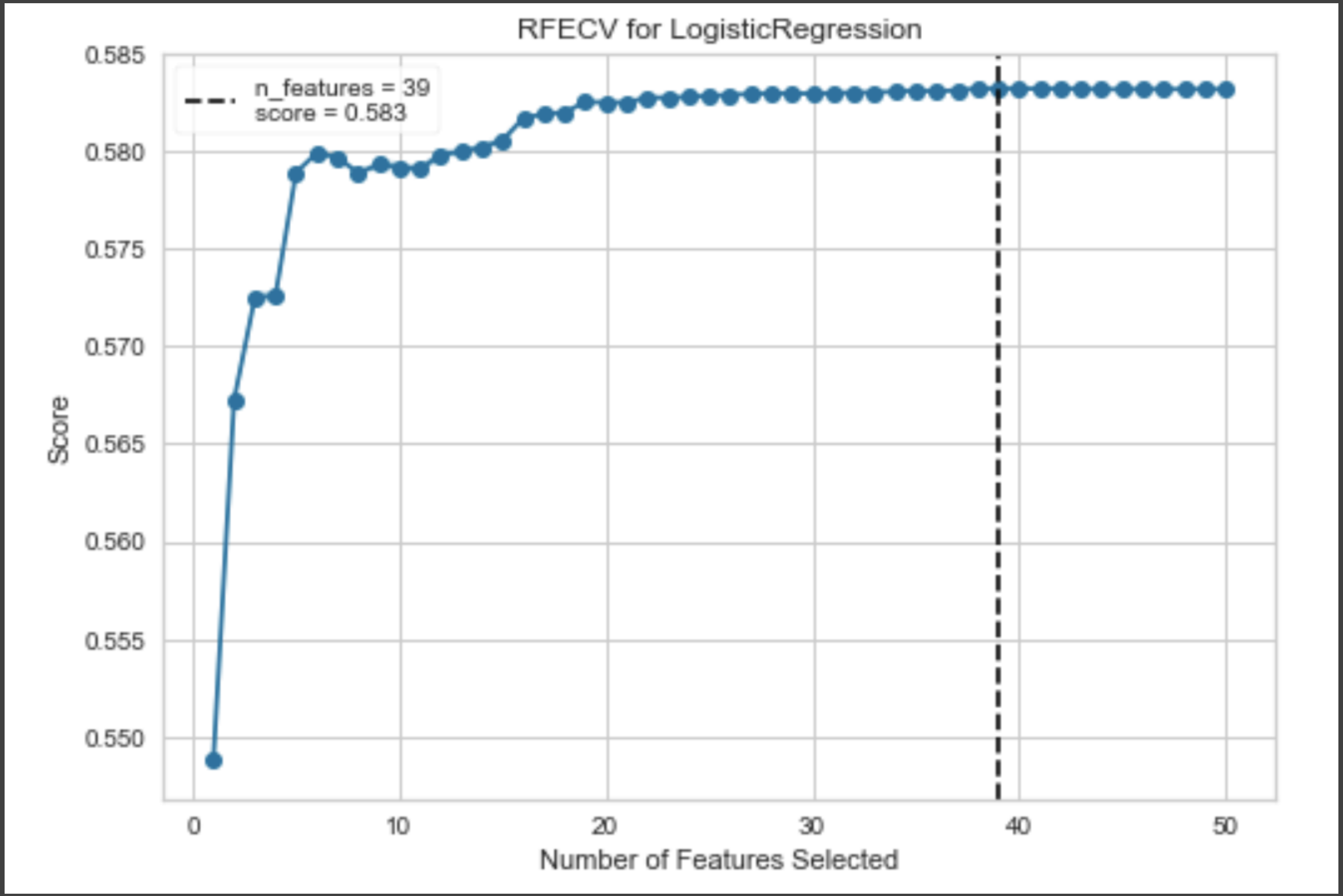
Data – Limitaion

- **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.
- **Outliers and Missing Data**
  - Aside from dropping columns with very high missing value precent, I also leave some of the missing data as missing, or replace the missing value with median value.

Feature Engineering summary

- 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.

Models – Logistic Regression

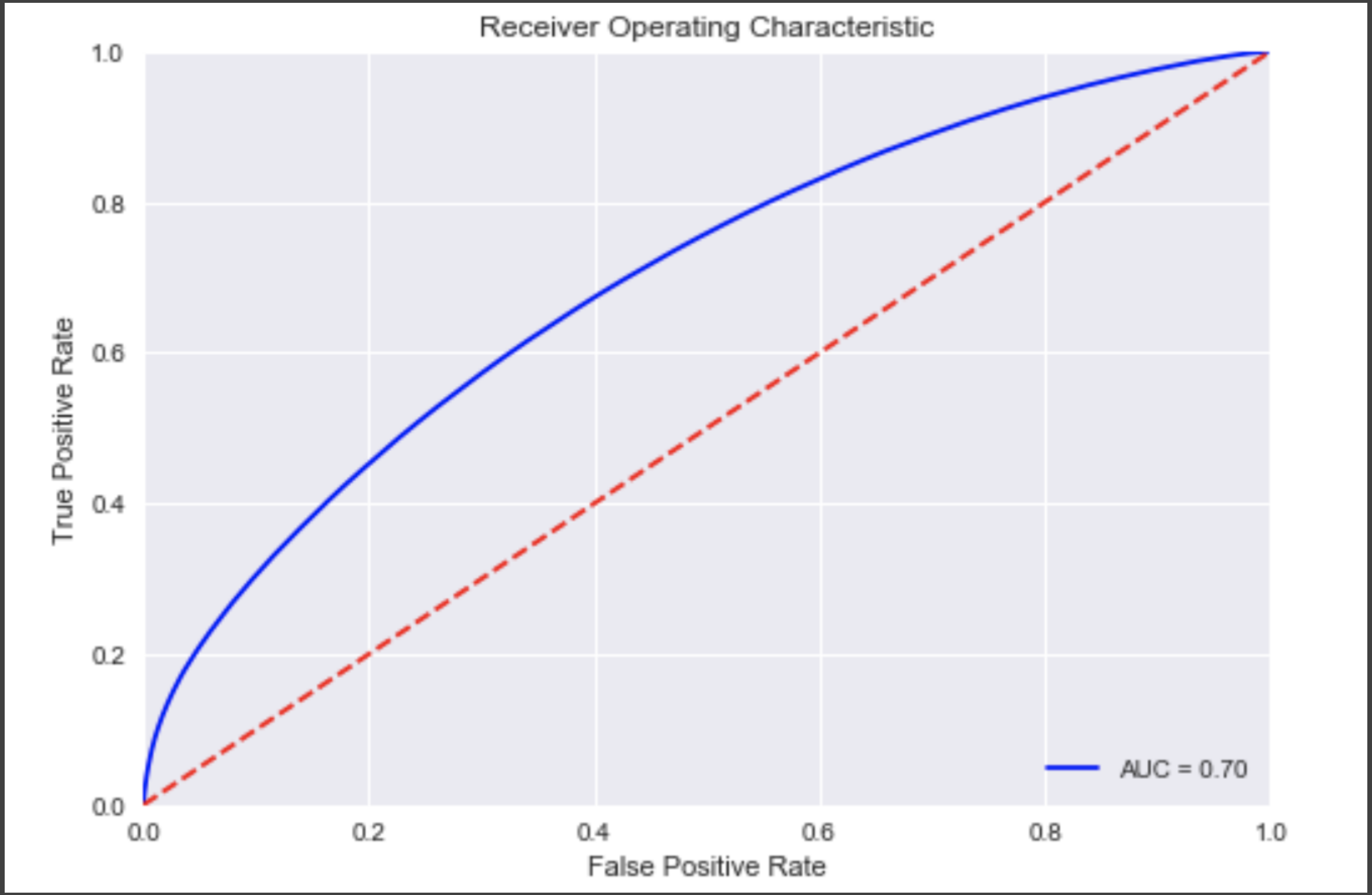


- First run gives a accuracy score of 0.49
- I normalized the data to avoid if there are some features that have more weight than others. After fitting the model, the accuracy score improved to 0.59 which is higher than the first time.
- I've also applied feature selection to select 50 features out of 227. Then the accuracy score of Logistic Regression is increased to 0.61

Models – 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.

Models – 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:

Conclusions

- 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.
- 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.