Microsoft Malware Prediction

DSO 530 Final Project

Level-II

Mengtian (Monica) Hu

Executive Summary

Nowadays, we are easily connected with each other because of the internet. Artificial intelligence generates huge financial benefit and improves our life quality. However, the security of data becomes a crucial problem in today's society. Malware is one of the reasons. In order to solve this problem for it's one billion enterprise and individual customers, Microsoft publishes a challenge on predicting if a machine will be soon hit by malware.

In this Microsoft Malware Prediction dataset, we have 83 predictors recording the machine information. Also, there is a label with 1/0 value that represents if the machine was hit by malware. 1 means yes and 0 means no. To simplify the problem, we randomly sample 100,000 observations for both training and testing dataset. Before building the model, we first cleaned the dataset and created 2000+ interactive variables and 6 expert variables.

Next, based on the correlation ranking of all variables with the target objective HasDetections, we picked 72 variables which have correlation no smaller than 0.05 for further feature selection. We applied two different approaches to select variables: Lasso logistic regression and best subset selection. To predict the probability of getting attacked by malware for each machine, we used 6 types of supervised models. We chose AUC as an evaluation matrix to measure model performance. After tuning the parameters of these models, XGBoost generated over 70% AUC on the testing dataset. By checking the training and testing accuracy at 0.5 thresholds, the XGBoost model had 66.75% for training and 64.11% on testing.

Data Description

Datasets: We randomly split 200,000 observations sampled from the raw dataset provided by Microsoft into equally two sets: training and testing. For training dataset, it is balanced on attacked vs. non-attacked. There are 83 predictors, which record the information on the machine. "HasDetections" is the label of this dataset. It contains binary value 0 or 1. 1 represents the machine was attacked and 0 represents the machine is secure so far.

Data cleaning and processing

It is crucial to clean and process the data before doing any data analysis. Here are the steps we cleaned and processed the data in our project.

1. Drop fields

In our data exploration, we found that some fields may not be beneficial to build models upon, so we decided to remove these fields. These fields include:

a) Fields with a missing value ratio of more than 50%, these fields include:

- DefaultBrawsersIndetifier
- PuaMode
- Census_ProcessorClass
- Census_InternalBatteryType
- Census_IsFlightingInternal
- Census_ThresholdOptIn
- Census_IsWIMBootEnabled

b) Categorical fields with thousands of classes and most groups contain few items. This is because it finds a good way to recode these fields. These fields include:

- MachineIdentifier
- Census_OEMModelIdentifier
- Census_FirmwareVersionIdentifier
- OsBuildLab
- AvSigVersion
- CityIdentifier

c) Categorical with extremely imbalanced classes. These fields include:

- Census_IsFlightsDisabled
- IsBeta
- AutoSampleOptIn

d) Numerical field with seemly mistaken values:

- Census_InternalBatteryNumberOfCharges

2. Filling Missing Values in Categorical Fields

We treat records with missing value in categorical fields as a new class. We believe that data are missing because there is a reason or there is not a reason. If there is a reason, then the class with missing values will behave differently, making treating it as a new group a reasonable choice. If there is not a reason, then we can treat this group as a sample of the whole population. After mean encoding, this group will have a value close to the probability of detecting malware in the whole population.

3. Correct and transform values

Some of the values are typed wrong, and some values can be reasonably merged into one value. The detailed method can be found in the Appendix.

4. Merge small categorical classes and mean encode

The way we process categorical fields is to mean encoding. We treat mean encoding as a statistical inference of mean. However, some classes under categorical fields have too few items, making statistics inference not accurate. As a result, we decide to merge classes with small items in them into one big class. Because different fields have a different situation, we set different thresholds for different fields. Classes with the number of items lower than the thresholds will be merged into one class. Thresholds are listed in the Appendix.

After we merged the small classes, we conducted mean encoding to transform all categorical fields into a numerical field.

5. **Fill Missing value in numerical fields**

In this step, we treat numerical fields with missing value as the predicted variable, use other fields without missing values to train predicted models. Here we believe the relationship is not linear, so we use the bagged decision tree as a prediction method. Cross-validation is used to check the predicting quality. The R^2 measure of cross-validation result is listed in the Appendix.

Variable creation

First, we create all possible interaction variables (about 2000+) as we lacked the knowledge about these variables and could not decide which variables should be interacted and which could not.

Second, we used our knowledge and business insights to create 6 expert variables:

primary_drive_c_ratio (Census_SystemVolumeTotalCapacity/ Census_PrimaryDiskTotalCapacity)

non_primary_drive_MB (Census_PrimaryDiskTotalCapacity- Census_SystemVolumeTotalCapacity)

aspect_ratio (Census_InternalPrimaryDisplayResolutionHorizontal/ Census InternalPrimaryDisplayResolutionVertical)

ram_per_processor (Census_TotalPhysicalRAM/Census_ProcessorCoreCount)

new_num_0 (Census_InternalPrimaryDiagonalDisplaySizeInInches/ Census_ProcessorCoreCount)

new_num_1 (Census_ProcessorCoreCount* Census_InternalPrimaryDiagonalDisplaySizeInInches)

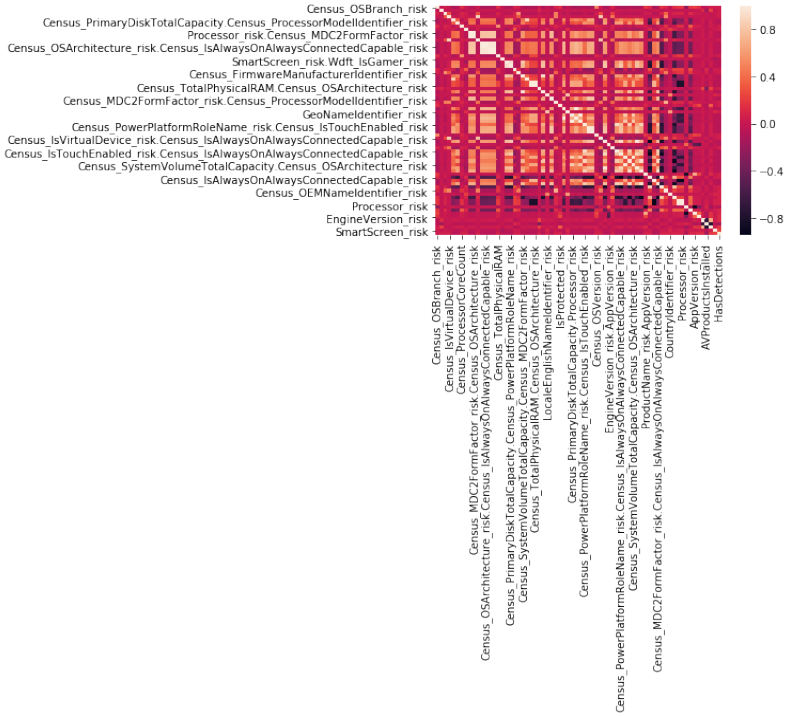

*Figure SEQ Figure * ARABIC 1 Correlation*

Third, in order to increase the model predicting ability and get rid of the collinearity among predictors, we calculated the correlation between predictors and the target variable "HasDetections" and the correlation among all predictors. We first removed highly correlated predictors from the model. We computed the variance inflation factor (VIF), which is the ratio of the variance of $\hat{\beta}_j$ when fitting the full model divided by the variance of $\hat{\beta}_j$ if fit on its own and removed ones with high VIF (larger than 5). Then we ranked the correlation between remaining

predictors and the target variable, and we picked top 72 variables which have the correlation larger or equal to 0.05.

Feature selection

After we created variables, we applied 2 ways to select our final features to predict detection:

1. Lasso logistic regression
   We applied Lasso logistic regression by R to select 66 final predictors
2. Best subset selection
   We applied best subset selection, which went over all possible models and finally selected 44 predictors

Model Building

In order to solve this classification problem, we experimented six different types of model and tuned parameters for each type model to improve the testing AUC. In this case, since we cannot determine which type of error is more important, we think that testing AUC will be the reasonable metric for evaluating the model performance. Our goal is to find a robust model with the highest AUC on the testing dataset.

The six types model we tried are logistic regression, Ada boosting, XGBoost, random forest, KNN and neural network. For logistic regression, since we used two types of approach to select the variables: Lasso and best subset, we built both Lasso logistic regression and simple logistic regression. For the other five types of model, we used the group of variables selected by the best subset approach.

1. Lasso Logistic Regression

In lasso regression, we first used cross-validation to select the best lambda value from a lambda list. As Figure 1 shows, the best lambda value is 0.000267, with the highest AUC. Then we use the whole training dataset train the model, using the best lambda. As the ROC curve is shown in Figure 2, the AUC is 0.6898781, with a threshold as 0.5, the testing accuracy is 0.63225 and training accuracy is 0.63113.

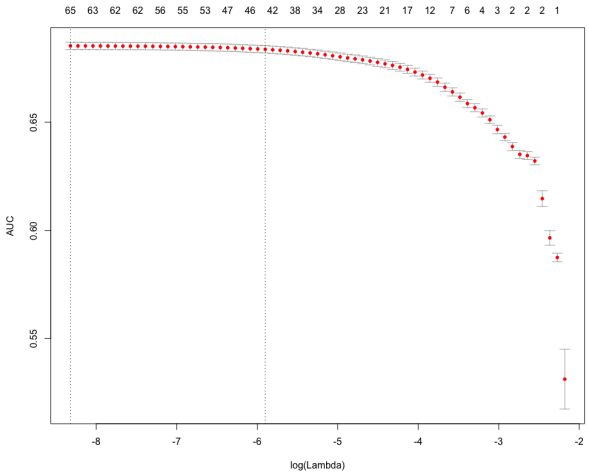

Figure 2 AUC vs. Lambda.

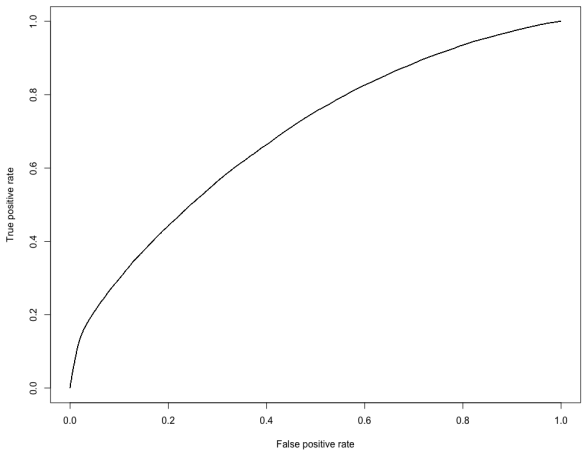

Figure 3 ROC Curve

2. Simple Logistic Regression

We implemented a simple linear regression as a baseline model. The model performance is 57.73% test AUC.

3. Ada Boost

We implemented Ada Boost Classifiers using default parameters. The maximum number of estimators at which boosting is terminated is 50. The learning rate is 1 and we do not use a pre-set best predictor. The model performance is 57.73% test AUC.

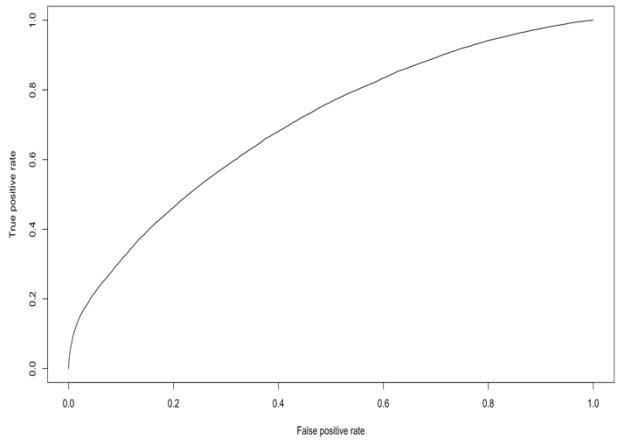

Figure 4

4. XGBoost

For XGBoost, we set the model had a maximum 30 iterations with 10 parallel trees each time and 8 maximum splits of each tree. For every subtree, by randomly selected 80% of observations and 80% of variables, the model resulted in the testing AUC at 70%. With the threshold at 0.5, the accuracy for training dataset is 66.75% and for the testing dataset is 64.11%. This model generated the best performance, compared to other models we tried.

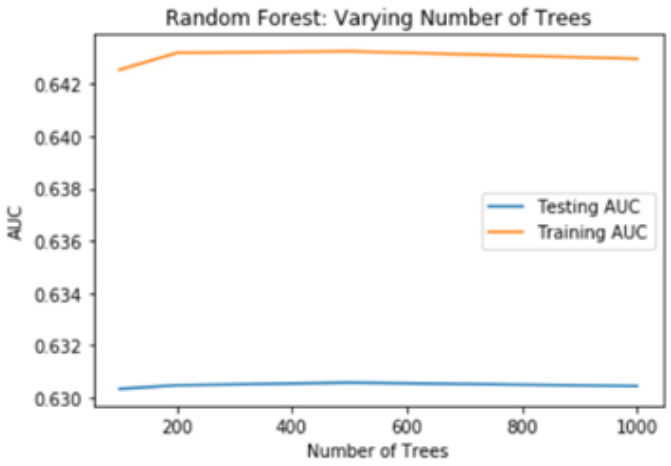

Figure 5

5. Random Forest

For Random Forests, first we tried the number of trees equals to 100 and maximum interaction depth equals to 8. The test AUC is 63.03%. Then we tried the different number of trees and summarized the testing AUC below. The random forest that yields the best result is when the number of trees is 500, and the best AUC is 63.06%.

6. K-nearest Neighbors

When the number of neighbor K equals 6, the model performance is 52.14%. When varying n, the training AUC dropped significantly while the testing AUC does not vary too much. The results are summarized in the chart below. The n which gives the highest testing AUC is 7.

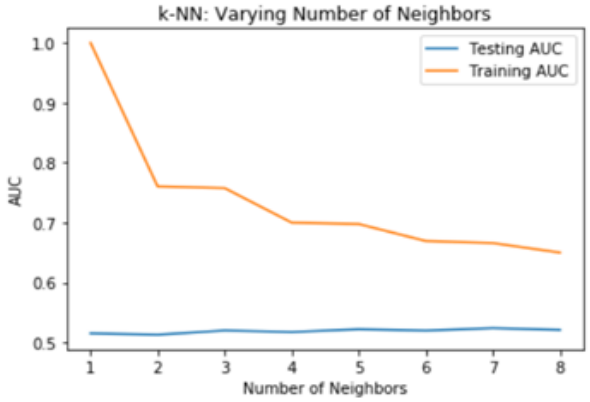

Figure 6

6. Deep Neural Network

We used the 44 features after the best subset to build the model. In this neural network, we build 3 hidden layers, the 1st hidden layer with 30 units, the 2nd hidden layer with 20 units, and 3rd hidden layer with 10 units, using 'Relu' activation function. For the output layer, we used the 'sigmoid' activation function. The loss function we used is 'binary_crossentropy'. To avoid overfitting and after several times attempts, we choose to randomly drop 30% neurons for each hidden layer.

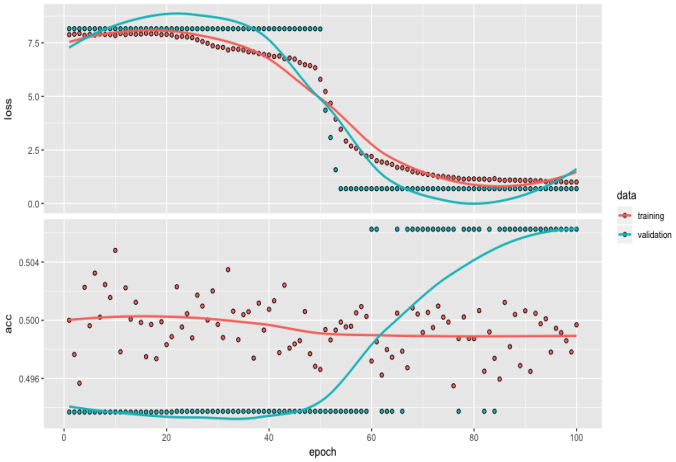

Figure 7

After 100 epochs, we select the best model from the 68th epoch, for the loss and accuracy in this epoch is the best. The training accuracy is 0.50156, the test accuracy is 0.49981, and the AUC is 0.5. This AUC and accuracy for test data showed that using a very complicated model is not very useful in this case. The training and validation loss history is as follows.

Summary Table

Model	Training Accuracy	Testing Accuracy	AUC (Test)
Lasso Regression	63.11%i	63.22%	0.69
Logistic Regression	53.53%	53.69%	0.56
Ada Boosting	64.07%	63.27%	0.63
XGBoost	66.75%	64.11%	0.70
Random Forest	64.27%	63.03%	0.63
6 Nearest Neighbors	67.07%	52.15%	0.52
Deep Neural Network	50.16%	49.98%	0.50

Business Interpretation

After obtaining the optimal XGBoost model, we summarized the characteristics of the top two most risky groups:
1. Machines with the following Defender Version "4.10.14393.0", "4.10.14393.1198", "4.10.14393.1593", "4.10.14393.1794", "4.10.209.0" have higher probability to be detected with malware. 2. If a Virtual Machine is installed, the computer has higher probability be detected with malware. 3. Computers with Monitor Size larger than 11 inches have higher probability be detected with malware. 4. Machines located in Country "104", "190", "95", "214", "100" have

higher probability to be detected with malware. 5. Computers with Census_OSArchitecture as “amd64” have a higher probability to be detected with malware.

The insight 3 may be linked to other factors such as the usage of the computer. We may not conclude causality due to correlation.

Statistical learning insights

In the data preprocessing part, we found that creating new variables based on business insights is better than creating all the possible interactive variables. Building models to predict missing values such as screening resolution is better using the median value. Before using mean encoding, it’s better to take a look at the categorical variables and merge low frequent categories.
In the modeling part, we found that simple model (logistic regression or KNN) and complicated mode (deep neural network) are not good choices in this case.

Extended question

With this dataset, we can also work on which kind(s) of the machine is safer for customers and enterprise to use. By solving this question, Microsoft can provide a constructive and compelling suggestion for its consumers. Moreover, based on the features of the safer machine, Microsoft is able to adjust and modify the current design of its products. By implementing both these two actions, the company can effectively improve the security of its customers’ devices and benefit the company self in the long run.

Appendix

1. Value correction and transformation:

Field	Original Value	Value after modification	Reason
OsVer	Values start with 6	6	Merge group
OsVer	Values start with 10	10	Merge group
SmartScreen	On, on	On	Correction
SmartScreen	Off, off	Off	Correction
SmartScreen	&#x02;, &#x01;	ExistsNotSet	Correction
SmartScreen	Unknown, NaN	Unknown	Correction & Merge
Census_ PrimaryDisk TypeName	UNKNOWN, Unspecified, Unknown, NaN	unknown	Correction & Merge

Census_ ChassisType Name	UNKNOWN, Unknown, NaN	UNKNOWN	Correction & Merge
Census_ PowerPlatfor mRoleName	UNKNOWN, Unspecified, NaN	UNKNOWN	Correction & Merge

2. Small classes merge thresholds:

Field	Threshold
Census_MDC2FormFactor	100
Census_OSInstallLanguageIdentifier	100
Census_ActivationChannel	110
Census_OSBranch	100
Census_OSEdition	100
Census_OSSkuName	100
Census_PowerPlatformRoleName	100
Census_OSVersion	100
Census_ProcessorModelIdentifier	100
SMode	6023
SkuEdition	60
LocaleEnglishNameIdentifier	100
CountryIdentifier	100
AVProductStatesIdentifier	100
EngineVersion	520
RtpStateBitfield	250
OrganizationIdentifier	105
GeoNameIdentifier	100

AVProductsInstalled	2500
AVProductsEnabled	2600
AppVersion	100
UacLuaenable	116
Census_OEMNameIdentifier	100
Census_ProcessorManufacturerIdentifier	465
Census_FirmwareManufacturerIdentifier	100
OsSuite	111
Census_OSUILocaleIdentifier	75
Census_ChassisTypeName	100

3. R^2 measure of cross validation for filling missing value in numerical fields

Field	R^2
IeVerIdentifier	0.99
Census_ProcessorCoreCount	0.79
Census_PrimaryDiskTotalCapacity	0.70
Census_SystemVolumeTotalCapacity	0.77
Census_TotalPhysicalRAM	0.54
Census_InternalPrimaryDiagonalDisplaySizeInInches	0.44
Census_InternalPrimaryDisplayResolutionHorizontal	0.97
Census_InternalPrimaryDisplayResolutionVertical	0.98