**Microsoft Malware Prediction**

**Prepared for:**

Professor Xin Tong

**Prepared by:**

Siqi Chen

Yadi Gong

Hadeer Hammad

Liana Liang

Xiaoyan Lin 5800503221

Contact Person:

Xiaoyan Lin 5800503221 xlin@usc.edu

April 19, 2019

**Table of Content**

[1.0 INTRODUCTION.. 1](file:///E:/STUDY/USC/Spring%2019/DSO%20562%20Fraud%20Analysis/Data%20Quality%20Report%20.docx#_Toc536085219)

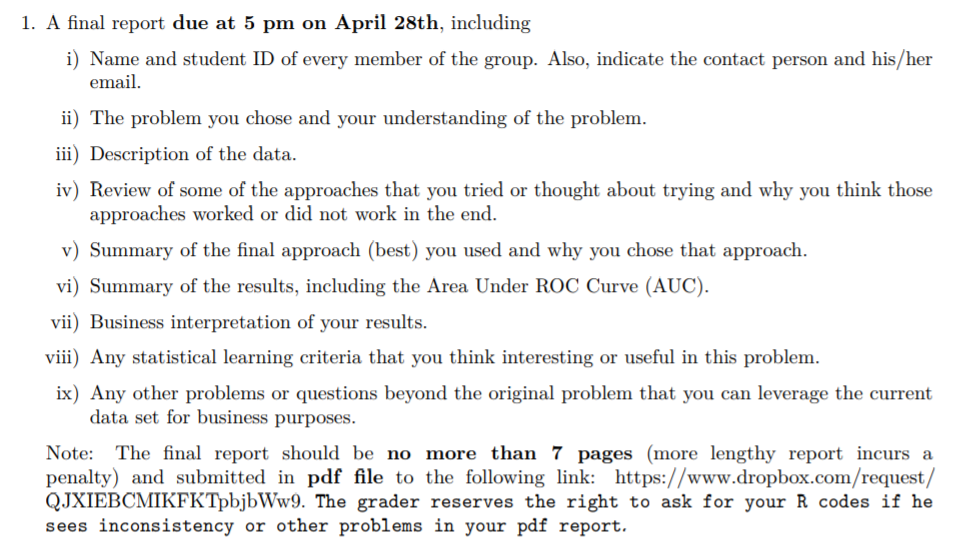
[1.1 Description. 1](file:///E:/STUDY/USC/Spring%2019/DSO%20562%20Fraud%20Analysis/Data%20Quality%20Report%20.docx#_Toc536085220)

[1.2 Dataset Information. 1](file:///E:/STUDY/USC/Spring%2019/DSO%20562%20Fraud%20Analysis/Data%20Quality%20Report%20.docx#_Toc536085221)

[2.0 FIELD SUMMARY.. 2](file:///E:/STUDY/USC/Spring%2019/DSO%20562%20Fraud%20Analysis/Data%20Quality%20Report%20.docx#_Toc536085222)

[3. FIELD ANALYSIS. 5](file:///E:/STUDY/USC/Spring%2019/DSO%20562%20Fraud%20Analysis/Data%20Quality%20Report%20.docx#_Toc536085223)

**MICROSOFT MALWARE PREDICTION REPORT**



# 1.0 INTRODUCTION

## 1.1 Description

The problem is to predict whether a computer is likely to be attacked by malware based on the machine’s various properties. The cost of malware infection could be very detrimental to consumers and enterprises and there are billion enterprise and consumer customers who use Microsoft’s Malware product. Therefore, it is crucial to develop predictive models in order to improve the cyber-security aspect of the Malware product. The data contains 83 variables and 8,921,483 observations. However, we used 100,000 random sample of the original data due to our limited computational power. The dataset entails a labeled field, indicating whether Malware was detected or not for each machine. We had built four supervised classification models and 3 classification tree-based models in order to predict Malware infection. (Only to be mentioned in the presentation and not report): *Although our intuition was guiding us to a certain few models, we had still applied all those seven models for our educational goal, which is developing a more robust understanding of the models’ application.*

UNDERSTANDING OF THE PROBLEM – Hadeer (See above 🙂 )

Microsoft provides a training dataset with 8,921,483 observations of 83 variables. We used a 100,000 random sample of the original data due to limited computing power. In this dataset, “HasDetections” is our response variable, which is what we want to predict. The response variable is balanced. (**Hadeer’s Note:** it’s great to mention that in the data cleaning portion).

## 1.2 Steps Performed

To build and select an optimal supervised model, we first cleaned the data, and then used feature selection methods to reduce the number of variables to 34. After that, we built seven supervised classification models: Logistic Regression, LDA, QDA, KNN, Bagging, Random Forests, and Boosting Trees, and found that XXX outperforms the rest six.

1.3 Conclusions

XXX

2.0 DATA CLEANING

1. Removed “MachineIdentifier” that are made as row index and are all distinct, since it cannot help with our analysis.
2. Remove 8 fields with more than 50% missing values, since there wouldn’t be a way to accurately filling in so many missing values.
3. Converting categorical fields into numeric forms:
   1. For categorical fields with less than 5 levels, we used dummy encoding.
   2. For categorical fields with more levels, we chose not to use dummy encoding because there would be too many variables for our models to handle and there would be infinite solutions to the coefficients. Instead, we used Mean Encoding. We computed the mean value of the response for each level, then replaced the value cells of that field with the result we got.
4. Filled in the NA or blank entries with numbers using function preProcess in Caret package with method = “medianImpute”, which replaced the missing values with the median of those observed values in each column.

3.0 FEATURE ENGINEERING

After the data cleaning, there are 83 variables in the cleaned dataset. In order to improve the model accuracy and the model interpretability, we tried different feature engineering methods to see their performances.

1. Feature Selection by RandomForest

We tried the very powerful train function in caret to train a random forests model (method = “rf”) with 5-fold cross-validation repeated 5 times. However, this method turned out to be computationally exhaustive and took too much time to run and we also found during the later model-building process, it even took a long time to run on the selected variables by using RadomForest, let alone 83 variables.

1. Best Subset Regression

After the random forest, we also tried the best subset regression, to try to find the most optimal combination of variables, which also turned to be pretty computationally inefficient.

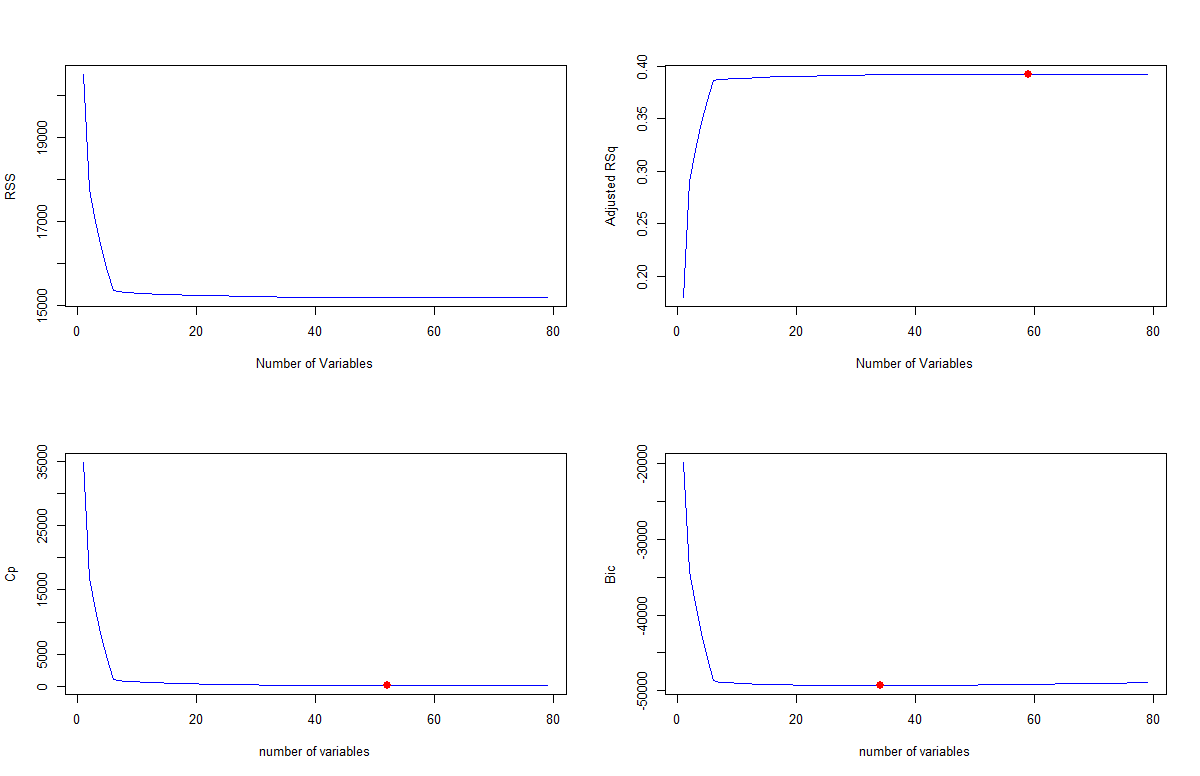
1. Lasso

As with ridge regression, the lasso shrinks the coeﬃcient estimates towards zero. However, in the case of the lasso, the *l*1 penalty has the eﬀect of forcing some of the coeﬃcient estimates to be exactly equal to zero when the tuning parameter λ is suﬃciently large, while Ridge will not set any of them exactly to zero and finally include all p predictors in the final model. Hence, much like best subset selection, the lasso performs variable selection. As a result, models generated from the lasso are generally much easier to interpret than those produced by ridge regression.

We applied Lasso with 5-fold cross validation, filtering 73 variables out of 83 variables, whose performance was even not as good as filtering out all near zero variance features. Even if we applied Lasso on the 58 variables selected by filtering out all near zero variance features, Lasso actually only removed 5 variables out of them and had no obvious improvement on the model interpretability.

1. Forward Stepwise Selection

Forward stepwise selection is a very computationally eﬃcient method even compared with backward selection. We always believe it’s better to improve the model simplicity as much as possible without the sacrifice of the model accuracy. We used cross-validated prediction error, Cp (AIC), BIC, or adjusted R2, to help us determine the number of variables to keep.



We chose the number of variables which get the lowest BIC as our final number of features, since BIC has a stronger penalty for including additional variables to the model compared with Cp, adjusted R^2, and BIC.

Therefore, we finally chose to keep 34 out of 83 variables.

# 4.0 MODELING

The purpose of our predictive models is to serve as auxiliary tools, meaning that they act as “second channels” for malware detection and therefore focusing on false positive is critical. From business standpoints, we do not want to waste our company’s resources on false alarms and instead dedicate them to real malware infection.

We had applied four classification models: *Logistic Regression, Linear Discriminant Analysis, Quadratic Discriminant Analysis, and K-Nearest Neighbors.* Additionally, we had builtthree tree-based classification models: *Random Forest, Bagging, and Boosting.* Bagging has performed the best in terms of yielding the lowest FPR and Random forests has performed the best in terms of yielding the lowest overall classification error rate.

## 4.1 Logistic Regression

We used logistic regression to model Windows machine’s probability of getting infected by various families of Malware. Logistic regression estimates the coefficients using the maximum likelihood method. In other words, it seeks estimates for the coefficient estimates as well as the intercept such that the predicted probability of Windows machine getting hit with Malware corresponds as closely as possible with the machine’s Malware infection. We set the threshold as 0.4747, as we found it to be the optimal point for minimizing FP without coming at a lot of expense of the FN rate. Furthermore, machines that have an 0.4747 probability of Malware detection or higher is classified as 1, which represents having Malware. Similarly, machines that have less than 0.4747 probability are classified as 0, which means they have no Malware.

|  |  |  |  |
| --- | --- | --- | --- |
| Threshold =0.4747 |  | Actual |  |
|  |  | Malware | NoMalware |
| Predict | Malware | 31136(TP) | 23124(FP) |
|  | NoMalware | 18845(FN) | 26895(TN) |

False Positive Rate **= 46.23%**

Some of Significant Coefficients’ Interpretation:

* A machine having an optical disk drive (CD/DVD) and gamer device increase the machine’s risk to be hit with malware, holding other factors constant.
* Various geographic locations put machines at different risks of Malware, holding other factors constant.
* There’s negative and significant association between the probability of a machine getting hit with Malware and its number of pixels in the vertical direction of the internal display. Specifically, a one-unit increase in the number of pixels in the vertical direction of the internal display decreases the log odds of machine’s risk to be hit with Malware by -0.0002519, holding other factors constant.
* There’s a negative and significant association between the probability of a machine getting hit with Malware and its number of logical cores in the processor. Specifically, a one-unit increase in the number of logical cores in the processor decreases the log odds of machine’s risk to be hit with Malware by 0.0297, holding other factors constant.

## 4.2 Linear Discriminant Analysis (LDA)

Like Logistic Regression, LDA is a linear function of the predictors; hence, it produces a linear decision boundary. LDA models the distribution of the predictors X separately in each of the response class: “has detection” and “has no detection.” LDA classifier results from assuming that the observations within each class come from a Gaussian distribution with a class-specific mean vector and a common covariance. If these Gaussian assumptions are met, then LDA outperforms Logistic Regression and vice-versa. Furthermore, when the classes are well-separated, the parameter estimates for the logistic regression model are surprisingly unstable; nevertheless, LDA and QDA do not suffer from this problem. It is interesting to note that when we set the threshold of logistic regression at 0.5, the results of LDA and logistic regression were almost similar.

|  |  |  |
| --- | --- | --- |
| LDA |  | Accuracy |
| No Detection Classification Error Rate: | 38.7% | 61.3% |
| Overall Classification Error Rate: | 41.1% | 58.9% |
| Detection Classification Error Rate: | 43.5% | 56.5% |

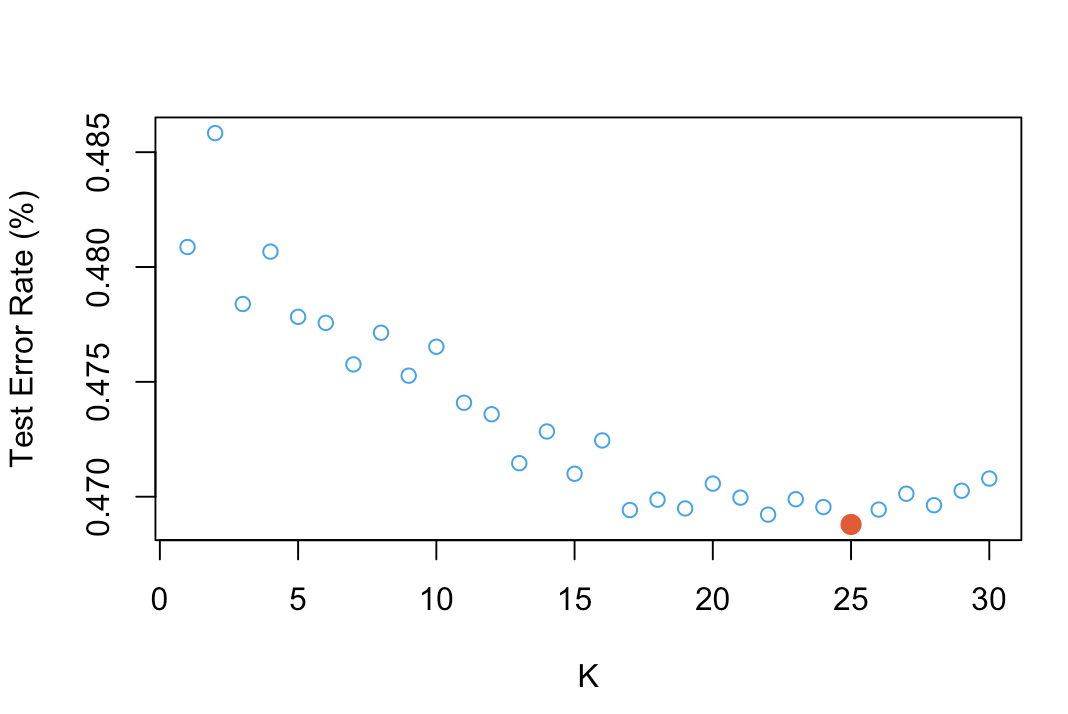
## 4.3 Quadratic Discriminant Analysis (QDA)

Like LDA, QDA results from assuming that the observations from each class are drawn from Normal Distribution. Nevertheless, QDA assumes that each class has its own covariance matrix. When the decision boundaries are moderately non-linear, QDA may give better results than logistic regression and LDA. However, QDA performs poorly relative to the other models that we had just discussed.

|  |  |  |
| --- | --- | --- |
| QDA |  | Accuracy |
| No Detection Classification Error Rate: | 22.6% | 77.4% |
| Overall Classification Error Rate: | 40.8% | 59.2% |
| Detection Classification Error Rate: | 58.9% | 41.1% |

## 4.3 KNN

K nearest neighbor is a classifier that estimates the conditional distribution of Y given X, and then classify a given observation to the class with highest estimated probability. We built our model as K changes from 1 to 30 and plot the test error rates.



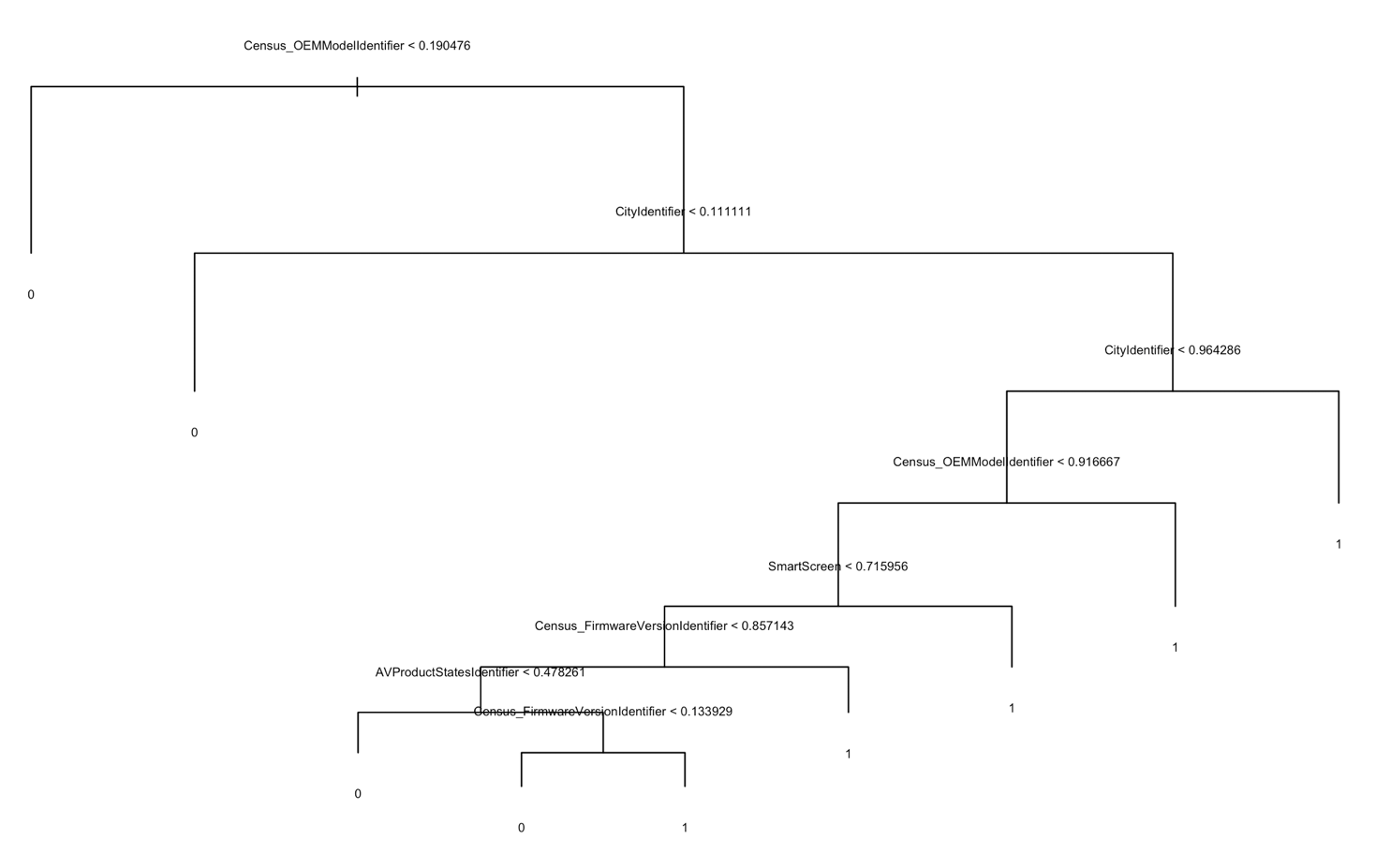
When K is greater than 17, the test error rate is decreasing slowly, and when K equals to 25, it has the lowest error rate. That’s why we set K = 25. The confusion matrix shows KNN doesn’t perform as well as the others. The overall accuracy is only 53.10%.

|  |  |  |  |
| --- | --- | --- | --- |
| K=25 | | Actual |  |
| Malware | No Malware |
| Predict | Malware | 27549 | 24472 |
|  | No Malware | 22432 | 25547 |

The false positive rate is 48.93% which means around 49% of the machines that don’t have a malware are labeled by our model as having a malware. Similarly, the false negative rate is 44.88%. Both of these two rates are higher than others so KNN would be the last choice to predict if a machine has a malware.

Tree based model

We built three tree-based models to fit training data set and drew a classification tree (below) using five variables: "Census\_OEMModelIdentifier" ,"CityIdentifier", "SmartScreen", "Census\_Firmware VersionIdentifier" , "AVProductStatesIdentifier". The number of terminal nodes is 9. The misclassification error rate for training dataset is 0.2818.

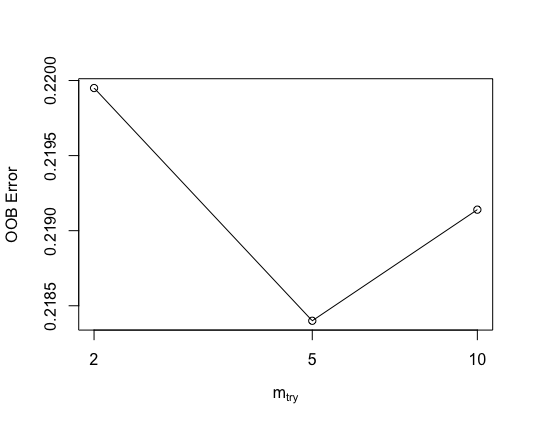


**Random Forest**

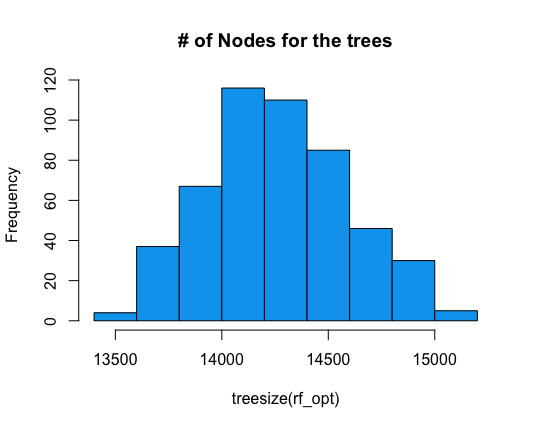
Tune parameters

Growing a random forest we use a smaller value of the mtry argument than bagging. By default, randomForest() uses p/3 variables when building a random forest of regression trees, and square root of p variables when building a random forest of classification trees. Here we

use mtry = 5 for it makes OOB error lower. Besides, we select number of trees as 500 because when trees are greater than 500, there is no significant reduction on error (picture with high quality is in our file).



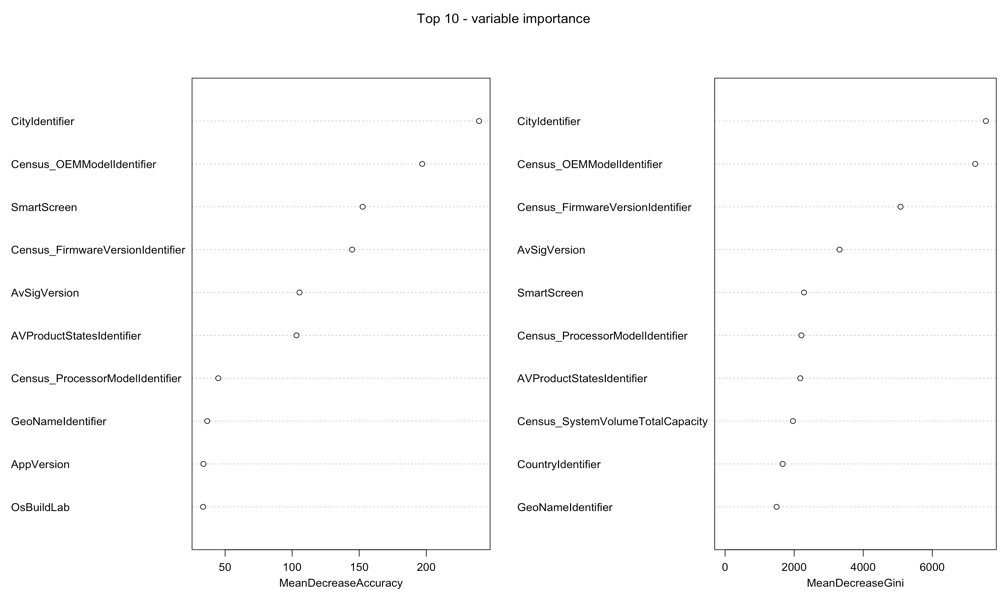
The plot below shows number of nodes distribution. The bottom axis shows the number of terminal nodes and top shows frequency. Most of the trees have more than 14000 nodes.



As for the performance, we ran the model 10 times and get the confusion matrix showing average result. The overall accuracy is 59.25% and false positive rate is 41.92%.

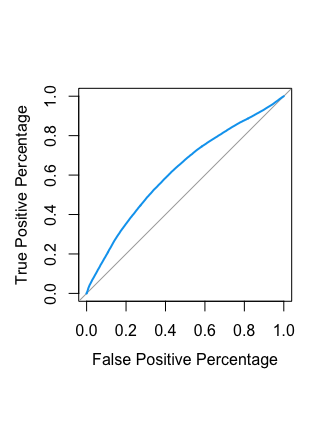
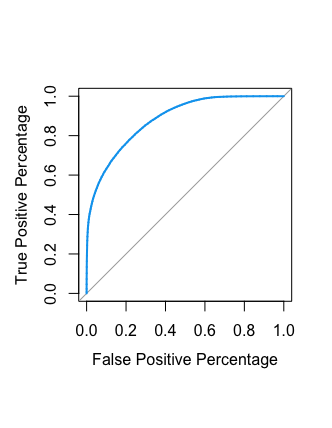
|  |  |  |  |
| --- | --- | --- | --- |
| Random Forest |  | Actual |  |
|  |  | Malware | No Malware |
| Predict | Malware | 30203 (TP) | 20970 (FP) |
|  | No Malware | 19778 (FN) | 29049 (TN) |

In addition, according to importance of each variable, the plot is figured out displaying top 10 importance variables. Cityidentifier, Countryidentifier and geonameidentifier are included in.



ROC curve for best model—random forest

The roc curve displays the combination of false positive rate and true positive rate under all possible thresholds. The left one is for training dataset and the one on the right-hand side is based on testing data. Apparently, training data has a better curve which is more towards to top left corner. The AUC for each plot are 0.8821 and 0.6193 respectively.



**Bagging**

Like random forest, bagging is also a tree-based model, but bagging use all 34 features at every split. Because trees in bagging are closely related to each other, bagging has lower accuracy compared to random forests. However, among all our models, bagging has the lowest False positive rate, 41.82%

|  |  |  |
| --- | --- | --- |
| Bagging |  | Accuracy |
| No Detection Classification Error Rate: | 40.96% | 59.04% |
| Overall Classification Error Rate: | 41.39% | 58.61% |
| Detection Classification Error Rate: | 41.82% | 58.18% |

**Boosting**

In this case, the performance of boosting is not as good as random forest and bagging.

|  |  |  |
| --- | --- | --- |
| Boosting |  | Accuracy |
| No Detection Classification Error Rate: | 38.29% | 61.71% |
| Overall Classification Error Rate: | 41.54% | 58.46% |
| Detection Classification Error Rate: | 44.79% | 55.21% |

**Insights and Business Recommendations**

There’s a recent test that shows some well-known threat detection products were not able to detect custom-written malware samples as well as advanced persistent threats. Therefore, we recommend that Microsoft builds customized Malware Detection products for enterprise by building additional protective features that are particularly resistant to the top threats for each geographic region as well as industry type. The reason why we believe customizing the product by geographic region could be a good idea is because geographic region was selected by Random forests and logistic regression models as a statistically significant predictor. Additionally, the reason we think customizing the product by industry type to be meaningful is because there is a research that shows that the Financial services industry is a main target for malware attacks since it operates software that tracks ownership for monetary assets. The costs of malware attack for financial institution are extremely high, as they could be monetary, damage to reputation and brand, reduction in customer assets, data breaches, and many more. An additional recommendation to Microsoft is to track the open-source social networks that cybercriminals use to discuss existing malware and how it can be developed to perform more evil tasks. We recommend that Microsoft build clustering models for key words in those sites pertaining to geographic region, industry type, and other relevant key words to improve its Malware detection capability.