Kaggle Project Report

STAT 841 2019W

Team name: EV1M31, Bingying Xia (20773904), Yuhao Wu (20662478)

0. Abstract

Malware industry has long been a serious problem for computer users. One of the major challenges that anti-malware software faces today are the large scale data that needs to be evaluated to predict if a machine will soon be hit by malware before damage happens. Therefore, the goal of Microsoft Malware Prediction Challenge was to build sophisticated models to automatically predict the probability of a Windows machine being infected by malware, based on different properties of that machine. In this project, we used 3 statistical learning models (LightGBM, Artificial Neural Network, and Random Forest), and generated 6 versions of dataset with different encoding techniques (frequency, one-hot, numeric, label encoding) to solve this problem. The results are evaluated by AUC between the predicted probability and the observation label. Stacking and blending finally is used to boost predictive AUC.

The results show that LGBM with dataset of 18 interactions and 8 dropped features produced the highest score on private leaderboard (0.658) compared to other single model, while LGBM with one hot encoding dataset hits the peak on public leaderboard. Stacking model developed by logistic regression outperforms all single models. Additionally, feature engineering generally improves a model's AUC more than hyper-parameter tuning.

Finally, our team get 0.640 on private leaderboard, ranking at 635 (top 27%).

1. Variable feature engineering

To encode the train and test datasets with 81 variables, 6 dataset versions were generated using different encoding and feature engineering method shown in table 1. Since the number of unique values in each variable is far less than the number of observations, in dataset V1, all variables were considered as categorical variables using labeling encoding methods so that the missing value was encoded as a category in dataset V1.

Table 1. List of dataset versions using different variable encoding method

Dataset name	Description
V1	Label encoding for all variables
V2	42 categorical variables; 18 frequency variables; 5 version variables; 16 continues variables
V3	V2, add 7 interaction variables
V4	V2, add 18 interaction variables, drop 8 variables
V5	V2, add 31 interaction variables, drop 8 variables
Vohe	One hot encode for all variables

According to the meaning of all variables, we classified the 81 variables into four types in dataset V2. The variables with multi-categories, such as *Processor Model Identifier*, *Processor Class*, and binary variables have been encoded as categorical variables by converting the labels into numeric form. According to Microsoft Security Intelligence Report (Microsoft , 2018), hackers went for the easy marks to reduce the attack cost with potentially high returns. Machines in different regions have different risk of being attacked because of varying network operators and internet traffic censorships. For example, the region network operator lose attention to cyber safety, and then a mass of machines could be attacked successfully in this region. Therefore, frequency encode method was adopted to encode the

variables with geospatial information, such as *Country Identifier*, *City Identifier*. The frequency of the category occurrence replaced the original label. In addition, there are five version variables, like *Defender App Version*. Those variables were converted to continuous numbers based on the ordinal relationship. The remaining variables were regarded as continuous variables, and their missing data have been substituted by average or mode.

According to feature importance computed by tree models and the practical meaning of variables, we further developed dataset V3 ~ V5, on the basis of V2, by adding different number of interaction variables with high importance values and significant practical characteristics. For example, *Smart screen* is a function that helps keep Windows safe by checking downloaded files and web content within apps. Features like *Smart screen* were selected to make interactions with other variables. After interaction, frequency encoding was implemented for each unique values because we want to know how many users are in that particular property combination. Moreover, we took 4 non-importance value variables out from datasets V4 and V5. Additionally, the original dataset has 4 variables with above 99% missing values. Thus, datasets V4 and V5 discarded the variables with batch-missing values.

In addition, dataset Vohe was generated by one-hot-encoding method, which can be applied to categorical variables where no ordinal relationship exists, because using numeric encoding and allowing model to assume a natural ordering between categories might result in unexpected results. Furthermore, the sparse-matrix strategy was implemented to conduct one-hot-encoding inspired by a kernel (Bogorod, 2019) to reduce memory usage.

2. Model fitting

2.1. Random forest

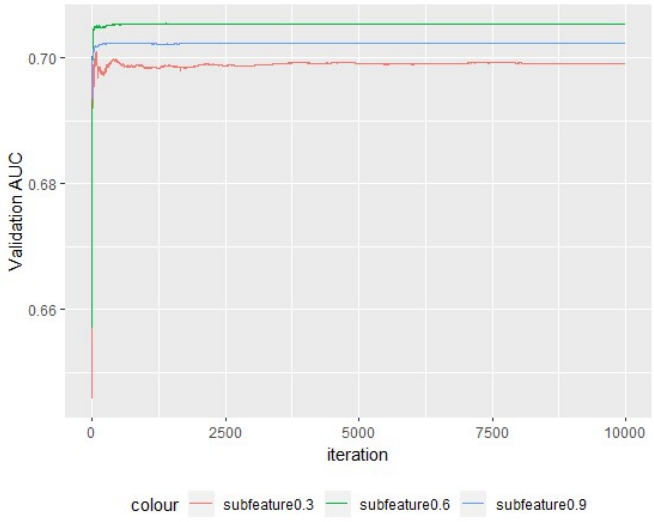


Figure 1.Line plot of CV AUC as a function of the number of iterations

Random Forest (RF) model is developed by building a number of decision trees on bootstrapped training samples. For a single decision tree building, a random sample with m predictors is chosen as split candidates from the full set of p predictors. The split uses only one of the m predictors to define the criteria of the bifurcation (James, Witten, Hastie, & Tibshirani, 2013). The number of predictors considered as split candidates and the number of trees are two important parameters for RF model. As figure 1 shows CV AUC as a function of the number of iterations with three variable fractions (30%, 60%, 90%). According to the

result, after 1000 iterations, the AUC is going to be stable and the AUC with 60% of feature fraction is higher than that of other feature fractions. Though different iteration numbers (1000, and 10000) demonstrates similar CV AUC, the public score with 10000 iterations (0.65) is higher than that of 1000 interactions (0.623). Thus, we selected the 60% feature fraction and 10000 iterations as the optimal parameters for RF.

2.2. Gradient boosting tree

Table 2. GBT parameter tuning

Shrinkage	Feature fraction	Depth	CV AUC
0.1	0.3	10	0.7204
	0.9	10	0.7219
	0.6	10	0.7294
0.05		10	0.7384
	0.3	5	0.7321
		-1	0.7316

Gradient boosting tree (GBT) is an ensemble model of trees trained in sequence (Friedman, 2001). GBT develops a decision tree in each iteration via fitting the negative gradients. The library with Python named LightGBM was implemented to develop GBT models and predict the result. LightGBM is able to train a traditional gradient boosting tree model but with a novel instance sampling technique (Gradient-based One-Side Sampling) and feature reduction method (Exclusive Feature Bunding) to improve the efficiency of computation.

As table 2 shows, the three model parameters were tuned by K-fold cross validation. Shrinkage is a regularization parameter to decelerate learning process to avoid overfitting. Feature fraction is the ratio of variables that is selected on iteration (Microsoft, 2016). Depth is the number of splits performed on a tree. According to the parameter tuning result, shrinkage of 0.05, feature fraction of 0.3 and depth of 10 are the best combination, despite the differences of CV AUC are slight among the posted combinations.

2.3. Artificial Neural Network

Table 3.

Num. of layers	Num. of neurons in each layer	Dropout rate	Bias	CV AUC
4	200/120/60/1	0.3	No	0.640
6	50/100/200/100/50/1	0.1	No	0.707
6	50/100/200/100/50/1	0.1	Yes	0.708

Neural network is a statistical learning method which model the target as a linear function by extracting linear combinations of the input features (Hastie, Tibshirani , & Friedman, 2001). We used a library named Keras in Python to test different neural structures based on CV AUC and public leaderboard score. finally built a fully-connected network using 6 hidden layers respectively with 50, 100, 200, 100, 50, and 1 neurons. The six layers in the model is able to provide enough flexibility to capture the nonlinearities in the data. ReLU activation, 0.1 dropout, Adam Optimizer, and decaying learning rate were implemented to train model using dataset with one hot encoding technique (Vohe). In order to deal with the internal covariate shift, we also set Batch-normalization and add bias value on each layer.

3. Result interpretation

3.1. Public score comparison

The intermediate plot (b) of figure 2 shows the public scores of three models trained by different dataset version. LGBM with the dataset developed by one-hot-encoding technique

(Vohe) outperforms the rest, with 0.693 public score. Moreover, the public score (0.674) of ANN with Vohe much higher than that of ANN trained by V1 and V2. RF trained by dataset Vohe has better performance compared to models trained by other datasets. A study demonstrated the same result where one-hot-encoding preceded ordinal encoding in ANN (Potdar, Pardawala, & Pai, 2017) .

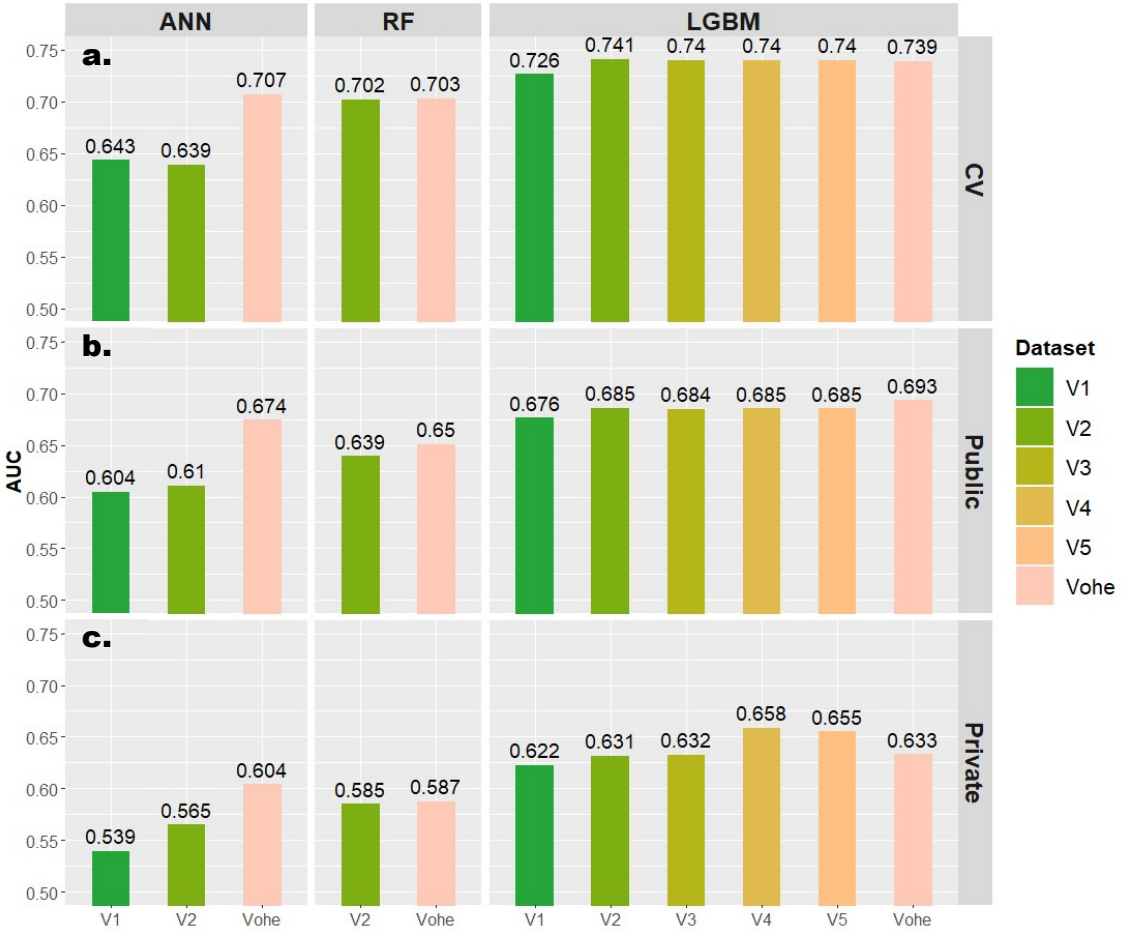


Figure 2. CV (a), Public (b), Private (c) scores of different models with datasets

3.2. Private score comparison

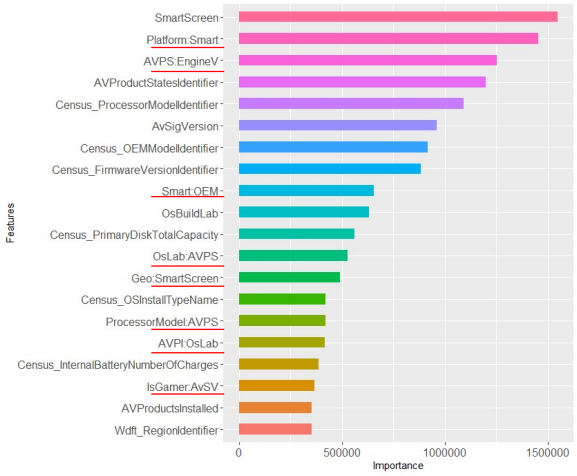


Figure 3 Feature importance of LGBM model with V4

As the bottom plot (c) of figure 2 shows, the LGBM model trained by V4 and V5 (0.658 & 0.655) preform preferably compared to LGBM models with other dataset even Vohe (0.633) in terms of private score. In the *Kaggle* Leaderboard, the two scores are severally in rank no.14 and 17. As mentioned in section 1, dataset V4 and V5 have respectively 18 and 31 interaction variables that were generated based on the feature importance and practical experience of malware. As the feature importance of LGBM with V5 shown in figure 3, 8 of top 20 significant variables are the interaction variables. Thus, the interaction variable contributed significantly to the model developments to improve AUC in terms of private score. However, V4 &V5 were not used to train RF and ANN, the private scores of single ANN and RF models with Vohe are higher than that of two models with datasets V1 and V2.

3.3. Model stacking

Table 4.

Coeff.	Est.	Std. Er	z-val.	P-val.
Intercept	-2.176	0.0056	- 386.73	$<2e^{-16}$
lgbm_Vohe	2.194	0.0106	206.09	$<2e^{-16}$
lgbm_V2	3.042	0.0102	295.95	$<2e^{-16}$
ANN	0.260	0.0082	31.41	$<2e^{-16}$
RF	-1.119	0.0169	-66.16	$<2e^{-16}$

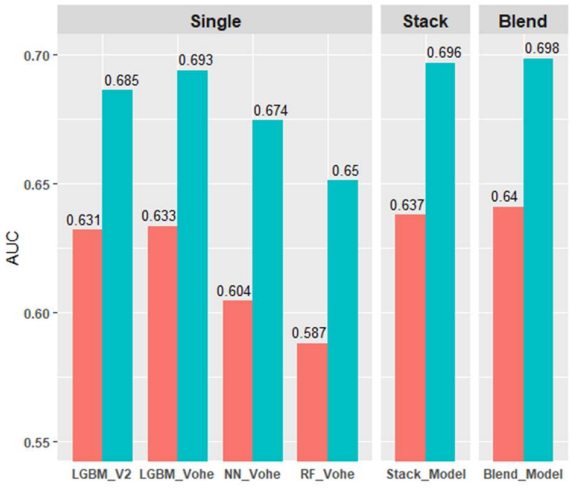


Figure 4 Public and Pricate scores of single models, stacked model and blended model

A stacking framework, using logistic regression, was applied to combine four single classifiers, such as *lgbm with Vohe*, *lgbm with V2*, *Neural network*, and *RF with Vohe*. K-fold cross validation procedure was employed to compute the label probability for each observation of training dataset. Then, the label probability predictions of each single classifier have been input into a logistic regression model as X-variables, and actual label data was

considered as y-variable. Meanwhile, single classifiers predict testing observations during K-fold validation. Finally, the generated logistic regression model was applied to predict the label probability of testing data using original testing predictions computed by the four single models.

As figure 4 shows, the stacked model with 0.696 public score and 0.637 private score outperforms each single classifier in both public and private scores. According to the *p-values* in table 4, the input training predications of all model are significant to the stacking model. Then, all testing predictions have been blended with an equal weight. With the blended prediction, the public and private scores are 0.69827 and 0.64070 respectively.

3.4. Overfitting in training, private testing and public testing data

According to the three bar plots in figure 2, we can draw a conclusion that, in all combinations of models and datasets, CV scores are higher than public scores and public scores are higher than private scores. CV scores were calculated based on AUC of K-fold cross-validation using the whole training data. In each combination of models and datasets, the greatest value difference among AUC values of K-folds is less than 0.001. However, the range of the score gap between CV and Public LB is from 0.04 to 0.06, and the score gap between Public and private is from 0.03 to 0.07. Thus, we do believe training, private testing and public testing data were not separated based on random sampling.

In all scores (public, private, CV), ANN and RF have similar result where models with Vohe are better than that trained by other dataset. However, in LGBM model, V4 and V5 (0.658 & 0.655) are better than that with Vohe (0.633) in terms of private LB, despite that Vohe (0.693) outperforms V4 and V5 (both 0.685) in public score. Moreover, the four datasets, V2 V3, V4, and V5, have mostly same values in both CV and public scores (0.740 & 0.685). Nevertheless, the greatest difference of their private scores is 0.0271, as the figure 2c shows, thus they are distinct according to the private score.

4. Conclusion and discussion

According to the result and overfitting problem in the training, public testing and private testing dataset, there are three conclusions and two limitations summarized below.

4.1. Conclusion

4.1.1. LGBM outperforms RF and ANN

GBM updates the observation and classifier weights in each interaction compared to RF where each observation and classifier has equal weight in the whole model. The reason why ANN cannot achieve the performance that LGBM presented might be that the structure of ANN proposed in this study is not the optimal one for the given Malware data. Thus, LGBM is superior to RF and ANN in both CV, Public score, and Private score in the result (figure 2).

4.1.2. Model stacking can improve the scores in Leaderboard

Based on the result (figure 3), the ensemble models by stacking and blending multiple model types are more accurate than models with a single algorithm. However, due to limited by the number of single models, this project only used the logistic regression to conduct model stacking. With the number of models increasing, random forest regression model could be adopted to stack models.

4.1.3. Feature engineering is more important than parameter tuning

Six dataset versions were proposed using different feature engineering methods in this project. As the result shown in section 2 and 3, parameter tuning only can increase AUC

values by around 0.01, but AUC difference of models with different dataset could be the range of 0.02 to 0.07. Hence, featuring engineering improves the predication accuracy than parameters tuning.

4.2. Limitations

4.2.1. Finding best dataset to do cross-validation

Based on the overfitting problem discussed in the previous section, a proper dataset for cross-validation is significantly important to this project. The public dataset and the private dataset were not split randomly from the original whole dataset, and testing dataset is quite different to training dataset. In 39 of 81 variables, the numbers of unique values between training and testing dataset are different. For example, figure 5 shows the histogram of number of observations in defender state versions (*AvSigVersion*) with training and testing datasets. Because the variable has the temporal property, we conducted ordinal encoding to convert the original values to continuous values. As figure 5 shows, the training dataset lacks observations with last versions, furthermore testing dataset only has a handful of observations with old versions. It could be one of reasons to result in the overfitting.

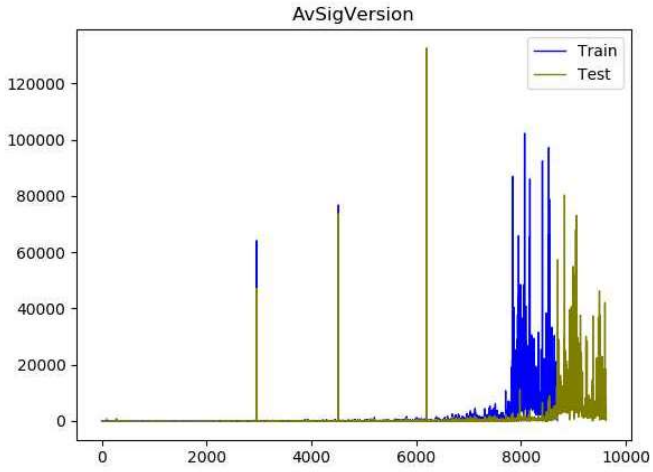


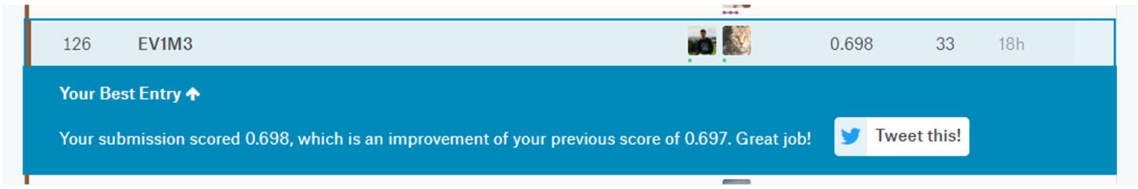
Figure 5. The histogram of defender state versions (AvSigVersion)

Hence, we were not able to depend on the CV AUC or the public score to choose our best model that has the highest private score. Meanwhile, it is more difficult to evaluate model performance comprehensively based on CV and public score. However, based on the variable distributions, we might employ advance sampling and feature selection methods to extract a validation dataset, which mimics the characteristic of the testing dataset, from training dataset. Then, the CV score based on the validation dataset is closer to the private score so that we can select the optimal model and feature engineering method.

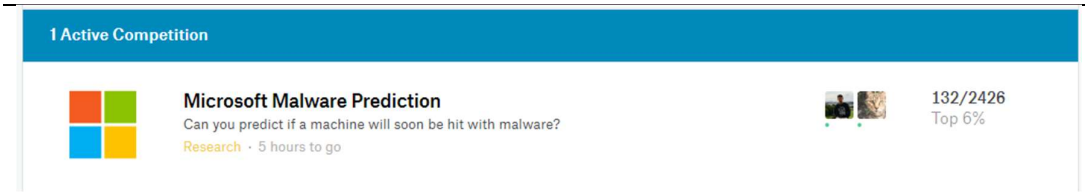
4.2.2. Using prior knowledge to do feature engineering

Due to lack of the background knowledge in Malware detection and absence of variable interpretation from the competition guide, we cannot process data to obtain extra variable information to encode a better dataset. In the future work, thought understanding and interpreting the variables more deeply, more useful interaction variables and information could be obtained to improve the predication accuracy of the testing data.

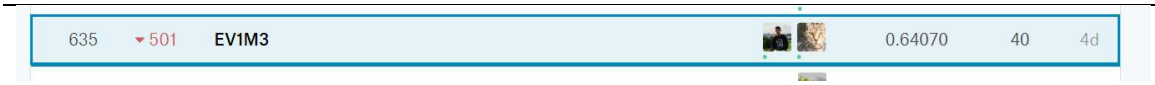
Appendix



a. The highest public score



b. The final public rank



c. Final private score and rank

d. The highest private score but without submission for participitation

References

Bogorod, V. (2019, Feb.). *LightGBM. Baseline Model Using Sparse Matrix*. Retrieved from Kaggle: https://www.kaggle.com/bogorodvo/lightgbm-baseline-model-using-sparse-matrix

Friedman, J. H. (2001). Greedy Function Approximation: A Gradient Boosting Machine. *The Annals of Statistics*, 1189-1232.

Hastie, T., Tibshirani , R., & Friedman, J. (2001). *The elements of statistical learning.* New York: Springer series in statistics.

James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). *An Introduction to Statistical Learning.* New York: Springer Science+Business Media.

Microsoft . (2018). *Microsoft Security Intelligence Report.* Seattle: Microsoft . Retrieved from https://info.microsoft.com/rs/157-GQE-382/images/EN-US_CNTNT-eBook-SIR-volume-23_March2018.pdf?mkt_tok=eyJpIjoiWmpVMVpHVTNNR1k0WkRrNSIsInQiOiJlVEZDdk0wSkd6SXd2Nzcrb3JlTjRZSHFQSUQzd3A4Tm94OHoyN3lSUVBSODhLXC9ZVjR6bTNWU051R1RnN2xuN3A3M1dNU3VWVDNreUduOU9DOHEz

Microsoft. (2016). *LightGBM Documetation.* Microsoft.

Potdar, K., Pardawala, T. S., & Pai, C. D. (2017). A Comparative Study of Categorical Variable Encoding Techniques for Neural Network Classifiers. *International Journal of Computer Applications*, 7-9.