Malware Prediction

Pratik Pawar
IIITB (MT2020121)
Pratik.Pawar@iiitb.org

***Abstract*—The goal of this assignment is to predict the probability of the machine being infected by malware depending upon nearly 80 parameters.**

***Index Terms*—Introduction, Feature Engineering, One-hot encoding, Train and test split, Cross Validation, Logistic Regression, Decision tree, LGBM, XGBoost, Weighted average method, Model Scores, References**

I. Introduction

The malware industry is a well-organized market dedicated to evading traditional security measures. A computer infected by malware, can be handled by criminals to hurt consumers and enterprises in several ways. The goal of this assignment is to predict the probability of the machine being infected by various families of malware beforehand, based on different properties of that machine, in order to protect them from damage.

II. Feature Engineering

A. Data Preprocessing and feature engineering

Our training data consists of 57730 rows * 83 columns. The 'HasDetections' column which is binary 0 (malware not detected) and 1(malware detected) is our ground truth. We have converted all object columns into category columns. As we can see from data there are all sorts of columns like categorical, numerical and binary. Our main goal of EDA is to determine which columns are important in predicting a device's malware probability.

.

B. Data Cleaning and handling null values

There are some columns in data having null values more than 40%. We have deleted these columns. Deleted columns are as follows

1) DefaultBrowserIdentifier
2) PuaMode
3) Census_ProcessorClass
4) Census_InternalBatteryType
5) Census_IsFlighingInternal
6) Census_ThresholdOptIn
7) Census_IsWIMBootEnabled

There were a few ways we could impute these values. Imputing with the mean did not make sense, because these values are discrete and specific. They are model numbers, identification numbers, etc. Mode seems to be a good choice for imputing the columns. For null values we have assigned mode value of the particular column computed by non-null values.

III. One hot Encoding

Each row in this dataset corresponds to a machine, uniquely identified by a MachineIdentifier.

We have used one-hot encoding for handling categorical columns, after doing this we got a feature matrix of size (567730, 8062). Similar kind of transformation is done on a test dataset.

IV. Train and test split

We have split train data to train(85%) and cross validation set(15%) and their corresponding ground truths. This helps us find nearly the best hyperparameters.

V. Models

A. Logistic Regression

Logistic regression basically gives output probability from 0 to 1 (both inclusive) , which tells about what are the chances of the machine being infected. Logistic regression is a simple model having nearly no hyperparameters to set. We got a public score of 0.54168 using this model. It was comparatively less than other models. Hence we decided to discard this model for predicting final output.

B. Decision Tree

Decision tree is a tree-structured classifier, where internal nodes represent the features of a dataset, branches represent the decision rules and each leaf node represents the outcome. It is a graphical representation for getting all the possible solutions to a problem/decision based on given conditions. We got a public score 0.55057, which is not good enough compared to other models.

C. Light GBM

Despite using Logistic Regression and Decision tree we haven't got better results. So, we have moved on to light GBM. LGBM is a gradient boosting algorithm which uses decision tree and expands in vertical fashion. Following are the generated parameters after hyperparameter tuning:

```
params = {'num_leaves': 60,
          'min_data_in_leaf': 100,
```

```
'objective':'binary',
'max_depth': -1,
'learning_rate': 0.1,
"boosting": "gbdt",
"feature_fraction": 0.8,
"bagging_freq": 1,
"bagging_fraction": 0.8,
"bagging_seed": 1,
"metric": 'auc',
"lambda_l1": 0.1,
"random_state": 133,
"verbosity": -1}
```

The public score improved to 0.71447 after using this model.

D. XGBoost

XGBoost is a decision-tree-based ensemble Machine Learning algorithm that uses a gradient boosting framework. It improves the error produced by model in subsequent iteration. Generally, XGBoost is fast when compared to other implementations of gradient boosting. XGBoost and Gradient Boosting Machines (GBMs) are both ensemble tree methods that apply the principle of boosting weak learners using the gradient descent architecture. However, XGBoost improves upon the base GBM framework through systems optimization and algorithmic enhancements.

```
params = {learning_rate=0.03,
          n_estimators=3000,
          max_depth=11,
          min_child_weight=9,
          gamma=0.2,
          subsample=1,
          colsample_bytree=0.4,
          objective= 'binary:logistic',
          nthread=-1,
          scale_pos_weight=1,
          reg_alpha = 0.6,
          reg_lambda = 3,
         seed=42}
```

We got the best public score for XGBoost implementation which is 0.71618. As we can see it is pretty decent as compared to logistic regression implementation.

VI. Weighted Average

As for this method, we should have model scores which are comparable and good. We will use both LGBM and XGBoost, as these are the top two models we got and their score is also similar. Weighted average increases generalizabillity property of overall model as it is better to rely on more models than single one.

We tried a couple of weighted predictions and we have got best results for 60% xgboost results and 40% lgbm results after parameter tuning.

Final output = 0.4 * lgbm prediction + 0.6 * xgboost prediction

Model Scores

Model Name	Public Score	Private Score
Logistic Regression	0.54168	0.54346
Decision Tree	0.55057	0.55310
Light GBM	0.71447	0.71332
XG Boost	0.71618	0.71532
Weighted LGBM (0.5) and XGB (0.5)	0.71787	0.71761
Weighted LGBM (0.4) and XGB (0.6)	0.71789	***0.71751***

References

[1] https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html

[2] https://towardsdatascience.com/decision-trees-in-machine-learning-641b9c4e8052.

[3] https://lightgbm.readthedocs.io/en/latest/Parameters.html .

[4] https://medium.com/@pushkarmandot/https-medium-com-pushkarmandot-what-is-lightgbm-how-to-implement-it-how-to-fine-tune-the-parameters-60347819b7fc

[5] https://towardsdatascience.com/https-medium-com-vishalmorde-xgboost-algorithm-long-she-may-rein-edd9f99be63d

[6] https://www.analyticsvidhya.com/blog/2017/06/which-algorithm-takes-the-crown-light-gbm-vs-xgboost/

[7] https://medium.com/swlh/microsoft-malware-prediction-using-classical-machine-learning-algorithms-5ade962ca73a.