MICROSOFT MALWARE PREDICTION KAGGLE COMPETITION ATTEMPT REPORT

Jatin Gupta, z5240221
Nishant Chokkarapu, z5242882
Marhadiasha Kusumawardhana, z5226934
Yash Tamakuwala, z5248584

INTRODUCTION

Microsoft held a Kaggle competition in an effort to improve the security of its products. Microsoft is the biggest operating system provider for personal computers. Its Windows operating system has approximately 1 billion users. Due to its large userbase, malicious hackers tend to develop malware for Windows as they have a higher chance of success in attacking Windows users. Microsoft needs to be steps ahead of these malicious hackers to protect its customers. In an effort to do that, Microsoft held a Kaggle competition, in which the participants are challenged to present a machine learning solution that can predict the probability of a Windows machine being infected with a malware given a set of attributes about the machine.

In the competition, the participants are challenged with a **binary classification problem**. Each example in the given dataset represents a machine and each machine can only be in one of the two classes. The class is represented by a target variable, `hasDetections`, whose value is either 1 or 0. If the value is 1 (positive class), it means that malware has been detected on the machine. On the other hand, the value is 0 (negative class) if no malware has been detected on the machine.

The participants of the competition are provided with a large and balanced dataset. The dataset was generated by combining heartbeat and threat reports gathered by Windows Defender. The dataset includes the training set and test set. The training set has 8 million rows, and its size is 4.38GB. The training set is balanced as approximately half of the examples is in a positive class and the other half is in the negative class. The test set, whose size is 3.8 GB, does not have the target variable, which itself has to be predicted. Each example in the dataset has **82 attributes**. We have classified each of these attributes into categorical, continuous, or Boolean attributes. There are 50 categorical attributes, 13 continuous attributes, and 19 boolean attributes.

Area under the ROC (receiver operating characteristic) curve metric is used to evaluate submissions to this competition. The participants are expected to come up with a machine learning model that predicts the probability of each example in the test set belonging to the positive class. Hence, submission in this competition is a list of the probability values for each example in the test set. Each submission is then evaluated by the area under the ROC curve between the predicted probability and the actual class of the examples.

We chose this competition as the topic of our project because it gives us a huge opportunity to gain valuable experience in machine learning and the problem is very interesting and well-defined. The fact that it is a binary classification problem gives us a valuable opportunity to learn by trying and comparing different machine learning techniques because most machine learning algorithms are designed to solve this problem. Indeed, our approach in this project is to compare and try multiple machine learning algorithms to find the one that produces a model that achieves the highest test score. Moreover, the given dataset, which is balanced and contains more than 80 features and millions of rows, gives us a opportunity to gain valuable experience in feature engineering and data cleansing. The semantics of the problem itself is very interesting because by working on this project,

we might be able to know what are the circumstances that cause a Windows machine to be more likely to get infected with a malware, which is an interesting knowledge to have.

SOLUTION FINDING STRATEGY

Our roadmap for coming up with a machine learning model that achieves a satisfactory performance for this competition consists of four parts: sampling, feature selection, encoding selection, and algorithm selection. Sampling is the process to select a fair, balanced, and representative subset of the whole dataset. Feature selection is an objective process of removing features that are not helpful in predicting the class. Encoding selection is the process of selecting the method to encode categorical features to numerical features that produce the highest test score. Algorithm selection is the process of experimenting with different available machine learning algorithm with different parameters and choosing the algorithm and its parameters that produce the highest test score, which is the area under the ROC curve.

SAMPLING

It is inescapable that we must work with a subset of the dataset provided by the competition instead of working with the whole dataset. As mentioned, the size of the training set is 4.38 GB and it consists of 8 million rows. On the other hand, our machines have limited memory and computing resources. Also, training a model with a huge dataset takes an unacceptable amount of time considering the limited time that we have for this project. Therefore, we decided to work with a representative subset of the dataset, instead of working with the whole dataset.

For that purpose, we conceived a scheme to produce a fair, balanced, and representative sample of the dataset. First, we decide on the sample size, let us denote it to be S. Then, we split the dataset into 2 files: one that contains all rows of positive class, and the other contains all rows of negative class. Next, we randomly take a subset of size $S/2$ of rows from each class from those 2 files and combine them into a single dataset with size S. The result of this process is the dataset that we will be working within the project. Because we have the same number of rows for each class, $S/2$, then this sample is balanced. Also, because we take the subset of each class randomly, then this sample is a fair representation of the dataset.

FEATURE SELECTION

The feature selection scheme that we apply consists of four parts. First, we remove features with the undesirable distribution of values, e.g. features that are dominated by a single value. Then, we cleanse the dataset to prepare it for statistical tests. After the dataset has been cleansed, we remove irrelevant features based on statistical significance tests. Lastly, we remove features that are highly correlated with other features. The result of this process is a sorted list of features to be used for model training. Figure 1 illustrates this selection process.

We consider three categories of undesirable value distribution of a feature. First, the value of each row for that feature is unique. Second, at least 95% of the dataset have the same value for that feature. In other words, the feature is dominated by a single value. Third, the majority of rows in the dataset have no value (null/missing) for that feature. If a feature is in one of these categories, we drop that feature.

Data cleansing and feature classification are done to prepare the dataset for statistical significance tests. The data cleansing operations include imputing missing values and lowering the case of string values. After cleansing, we classify each feature to be categorical or continuous based on its value and our domain knowledge. This classification is needed because different statistical significance test is applied to a different type of features.

We also drop features that are considered irrelevant by using statistical significance tests. For each categorical feature, we test its significance using the chi-squared test. For each continuous feature, we test its significance using the F-test. If the p-value of the result of the test of a feature is more than 0.05, we drop that feature.

Because we need to encode categorical features to numerical features in the training of our models, we also test encoded categorical features with the F-test. We also drop encoded categorical features that have the p-value more than 0.05 from the result of the F-test. The statistical tests and the selection process are automatically conducted using a Python script.In the last step, we drop features that are highly correlated with other features. The absolute value of the Pearson correlation coefficient is calculated for all pair of features in the dataset. If a pair of feature has value more than 0.9, we drop one of the features in that pair. The feature that is dropped is the feature has less F-test score than its counterpart. This process is also automatically conducted in the Python script.

The result of this process is a sorted list of features, which is a subset of all of the features, that we will use to train machine learning models. This list is sorted by the F-value of the features that we get from the F-test of each feature. As mentioned before, we also do F-test on categorical features after they are encoded. Hence, we can sort them together with continuous attributes in a single list. Note that different encoding method might result in a different sorted list of features because the F-value of each encoded feature depends on its encoding method.

We need to sort the features by significance because we need to limit the number of features when training some models. Some of the algorithms, such as KNN and SVM, take a lot of time for training, especially when the dataset has many features. Hence, we must explicitly limit the number of features (k) that are used in training those models. Because we have sorted the features based on statistical significance, we can easily take k most significant features to be used in training those models.

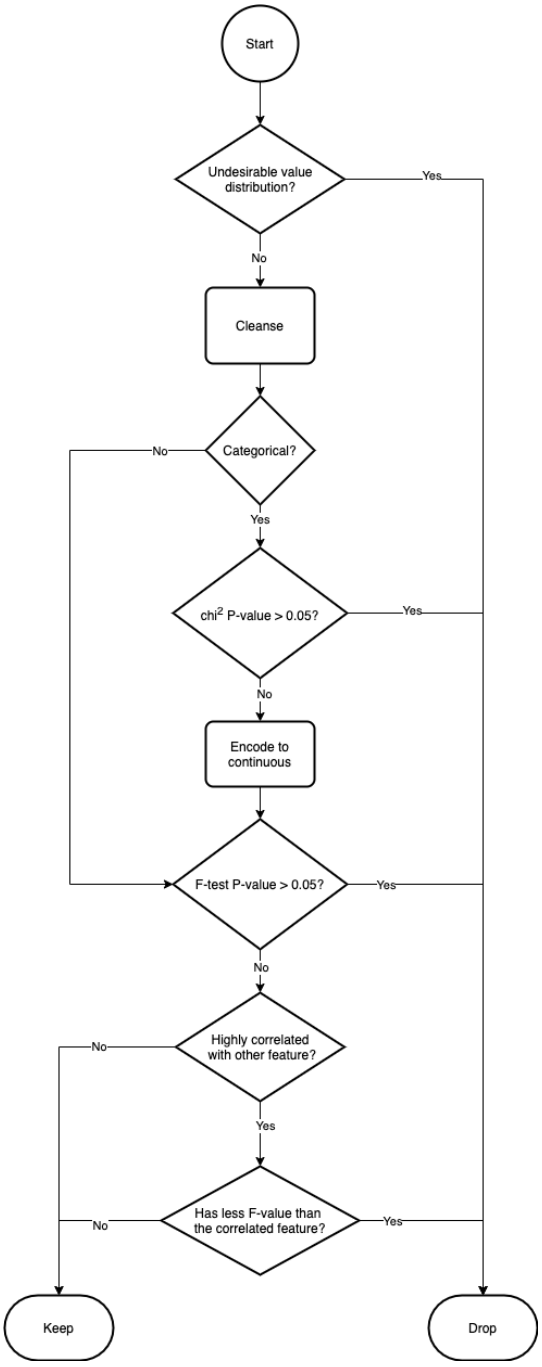

FIGURE 1. FLOWCHART DESCRIBING THE FEATURE SELECTION PROCESS

ENCODING SELECTION

The performance of the model is dependent on the choice of the encoding method of categorical features. The machine learning algorithm implementations that are available, especially from scikit-learn, require all features to be numerical. Hence, the encoding of categorical features to continuous numerical features is required. The ideal encoding method encodes a categorical value to a continuous value that correctly represents the category such that the value is relevant and useful in prediction. There are various methods to encode categorical features and not all methods can achieve this ideal because their usefulness depends on the nature of the problem, the features, and the target variable. Hence, the choice of the encoder is important and has an impact on the test score.

Some encoding methods are taken out of our consideration because of various reasons. Ordinal encoding is not suitable because most of the categorical features do not have an order, e.g. `CityIdentifier`. In addition,

for some features that might have an order, like `SkuEdition`, we cannot accurately map each category to a number that represents its position in an ordering that is relevant to the target variable. Hashing also does not help because the output of that encoding is arbitrary. We also dismiss encoding methods that increase the dimension of the dataset, such as one-hot encoding and binary encoding. We dismiss them because our dataset dimension is already high (more than 40) and the cardinality of some of the categorical features, such as the `CityIdentifier` and `CountryIdentifier`, is very high.

Encoding methods that we consider to be suitable to represent categories with numerical values are target-based encoding. The basic target encoding method uses a Bayesian probabilistic framework to replace a categorical value with the probability of an example in that category belongs to the positive class. There exist variations of this target-based encoding method that modify the encoding process in order to reduce overfitting. Target-based encodings that we consider are the **basic target encoding** itself, leave-one-out encoder, **James-Stein encoder**, and **CatBoost encoder**. However, at the time of this writing, the widely available leave-one-out encoder library is faulty when used with the scikit-learn Pipeline pattern. Hence, we have to take leave-one-out encoder out of the consideration.

We also consider the **frequency encoding** method as suitable to be used to encode categorical features. The frequency encoding method encodes each category to be the population size of that category in the dataset. We consider this encoding because there is a possibility that there is some relation between the population size of the category with the target variable. In addition, we can fit the encoder with both the training set and the test set because it does not need the label of each example. Fitting the frequency encoder with the test set is permitted in the competition because the competition gives access to the label-less test set. Hence, we consider this encoding method as one of the candidate encoders to be used in our solution.

The encoding selection process is conducted together with the algorithm selection process. In our model benchmark, we will test models that are produced by each combination of algorithm and encoding that we consider. In other words, each candidate algorithm will be combined with 4 different encodings to produce different models. The combination of the learning algorithm and encoding method that produces the model with the highest test score would be selected as our solution.

ALGORITHM SELECTION

We consider various classification machine learning algorithms as candidates to be used as our solution. The algorithms that we consider include **decision tree, logistic regression**, **SVM, and k-nearest neighbour**. In addition to experimenting with those base algorithms, we will also try ensemble algorithms such as **bagging**, **random forest**, **AdaBoost**, and **gradient boosting**. All of the ensemble algorithms that we try are ensembles on top of the decision tree algorithm. We use the **scikit-learn** implementation of each algorithm except gradient boosting. For gradient boosting, we use the **LightGBM** implementation of the algorithm.

We conduct a benchmark test on each algorithm that we consider in order to pick the algorithm that produces the model with the highest test score (area under the ROC curve), as our solution. First, the dataset is cleansed with the same operations as the cleansing in the feature selection process and each categorical feature in the dataset is encoded with an encoding method. Then, we split the dataset into a training set and test set. Then, we conduct a grid search with cross-validation on the training set to tune the algorithm parameters. After that, we train a model using the algorithm with the tuned parameters on the training set. Lastly, the test score of the model is produced by calculating the ROC AUC score of the model's prediction on the test set. The combination of the learning algorithm and encoding method that produces the model with the highest test score would be selected as our solution.

Note that the grid search with cross-validation that we conduct on each algorithm is not exhaustive. We use scikit-learn's `GridSearchCV` function to conduct this process. However, because of limited time and

computing resources, we cannot afford to perform parameter search with a large parameter grid. Hence, we consider a limited number (5 to 6) of combinations of parameters in the grid search.

We use different sample size and number of features in model training depending on the training time or prediction time of the algorithm. Prediction time of K-nearest neighbour and training time of SVM algorithms increase considerably as the number of feature increases. The scikit-learn implementation of SVM, especially when it is tasked to predict probability, has considerably longer training time compared to other algorithms. Because we only have limited time and computing resources, we cannot afford to use all of the features or a large sample size when training models using these algorithms. Hence, for these algorithms, we have to explicitly limit the number of features or use a smaller sample size compared to the sample size used in other algorithms.

RESULTS

This section is divided into two subsections, feature selection results and model benchmark results. In the first subsection, we will report the result of the feature selection process. As mentioned, different encoding method produces a different list of features. Hence, in that subsection, we will report the feature selection results for each different encoding method. In the second subsection, we discuss and compare the results of benchmark tests on models produced by each combination of algorithm and encoding method in order to pick the combination that will be our solution to the problem.

FEATURE SELECTION RESULTS

The feature selection process leaves around 45 to 49 features to be used for training the model. Feature selection processes using base target encoding and CatBoost encoding produces 48 features. Feature selection using James-Stein encoding results in 49 features. Lastly, feature selection using frequency encoding produces only 45 features. Table 2 breaks down the number of features that are dropped at each step of the feature selection process for each encoding method. Note that a sample of 100,000 examples is used for the feature selection process. Detailed results of the feature selection processes are stored in the `feature_selection_results` directory in the submitted code repository. Link to the sample that we use in this process is provided at the end of the report.

TABLE 1. BREAKDOWN OF NUMBER OF FEATURES THAT ARE DROPPED AT EACH STEP OF THE FEATURE SELECTION PROCESS.

	Base Target	James-Stein	CatBoost	Frequency
Original number of features	82	82	82	82
Unique values	-1	-1	-1	-1
Majority value is null	-7	-7	-7	-7
Dominated by single value	-16	-16	-16	-16
Statistically insignificant	-5	-5	-5	-6
Highly correlated	-5	-4	-5	-7
Final number of selected features	**48**	**49**	**48**	**45**

Different choice of encoding also affects the statistical significance measure of encoded categorical features. Table 3 lists the 5 most statistically significant selected features for each encoding method. As we can see, some of the top 5 most significant features for each encoding method are different. However, the top 5 features of base target encoding and James-Stein encoding methods are the same but in a different order. Meanwhile, 4 of the top 5 features of CatBoost and frequency encoding are the same.

There are some interesting observations that we get from the feature selection results. First, the `SmartScreen` feature is in the top 5 most significant features for all encodings. Hence, those features might be very decisive in determining the target variable. Second, during feature selection using frequency encoding, `CityIdentifier` is dropped because of statistical insignificance, although it is in the top 3 most significant features when it is encoded using base target encoding and James-Stein encoding. Lastly, `Census_OEMModelIdentifier` and `Census_FirmwareVersionIdentifier` do not even rank in the top 20 of the most significant features when they are encoded using CatBoost encoder or frequency encoding method, even though they are in the top 3 most significant features if they are encoded using the base target encoding and James-Stein encoding methods.

TABLE 2. TOP 5 MOST STATISTICALLY SIGNIFICANT FEATURES FOR EACH ENCODING METHOD.

	Base Target	James-Stein	CatBoost	Frequency
1	`Census_OEMModelIdentifier`	`Census_OEMModelIdentifier`	`SmartScreen`	`SmartScreen`
2	`Census_FirmwareVersionIdentifier`	`CityIdentifier`	`AVProductStatesIdentifier`	`AVProductsInstalled`
3	`CityIdentifier`	`Census_FirmwareVersionIdentifier`	`AVProductsInstalled`	`AVProductStatesIdentifier`
4	`SmartScreen`	`AVSigVersion`	`EngineVersion`	`EngineVersion`
5	`AVSigVersion`	`SmartScreen`	`AppVersion`	`Processor`

MODEL BENCHMARK RESULTS

We conducted a benchmark test for models trained by each combination of the machine learning algorithm and encoding methods that we consider. We find that the model trained using the **LightGBM implementation of gradient boosting algorithm** with categorical features encoded by the **frequency** encoding method achieves the highest test score, which is **70.68%.** The model parameter values that achieve this result are 100 for `min_child_samples` and 5 for `reg_alpha`. Hence, this combination of algorithm and encoding method is picked as our solution. Table 4 shows the test scores of each combination, with the highest test score shown in bold. Figure 2 compares the test scores of each combination in a bar chart. Detailed results of the benchmark tests are stored in the `benchmark_results` directory in the submitted code repository.

TABLE 3 THE TEST SCORES OF EACH COMBINATION OF TESTED MACHINE LEARNING ALGORITHMS AND ENCODING METHODS

	Base Target	**James-Stein**	**CatBoost**	**Frequency**
Decision Tree (CART)	62.50%	59.30%	67.14%	67.25%
Logistic Regression	63.39%	59.67%	68.98%	66.30%
SVM	59.11%	54.10%	67.62%	65.15%
K-Nearest Neighbour	61.44%	58.80%	63.49%	62.49%
Bagging	62.31%	60.39%	66.97%	65.63%
Random Forest	66.30%	62.93%	69.99%	69.51%
AdaBoost	63.11%	60.28%	69.54%	69.17%
Gradient Boost (LightGBM)	64.02%	60.18%	70.51%	**70.68%**

As mentioned, we use different sample size and number of features in model training of different algorithms. The training set and the test set for benchmarks of the decision tree, logistic regression, KNN, random forest, AdaBoost, and the gradient boost algorithms are taken from a random balanced sample with 100,000 examples. On the other hand, because benchmarks tests of SVM models that are able to predict probability are far slower and more computationally expensive, the training set and test set for benchmarks of SVM models are taken

from a random balanced sample with only 10,000 examples. Moreover, because a large number of dimensions slow down the benchmark tests of SVM and KNN algorithms considerably, models trained by them are limited to consider only the top 10 most statistically significant features. Links to samples that we use in these tests are provided at the end of the report.

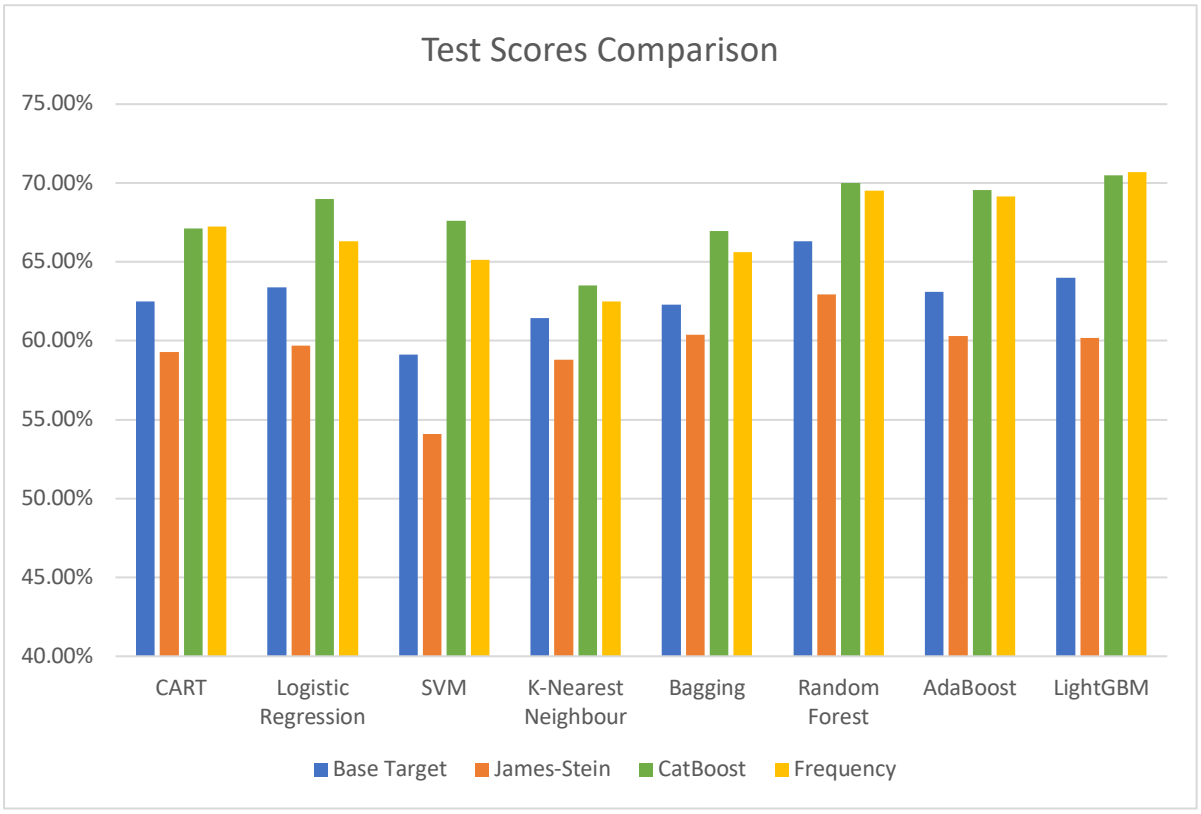

FIGURE 2. COMPARISON OF TEST SCORES OF EACH COMBINATION OF MACHINE LEARNING ALGORITHM AND ENCODING METHOD.

There are some interesting observations that we get from the results of the model benchmark tests. First, models that are trained using CatBoost encoder and frequency encoding method perform considerably better than models that are trained using the base target encoder and James-Stein encoder. Second, models with categorical variables encoded by the James-Stein encoder always perform worse than models with categorical variables encoded by the other three encoders. Lastly, models trained by some ensemble algorithms, such as random forest, AdaBoost, and gradient boost, indeed perform better on average than models trained by the base algorithms.

CONCLUSION

After conducting feature selection and model benchmark tests, we are able to come up with a solution to the Microsoft Malware Prediction Kaggle competition. Before training a model, we removed features from the dataset that have undesirable value distribution, are statistically insignificant, and are highly correlated with other features. Then, we conducted benchmark tests on models trained by different combinations of machine learning algorithms and categorical feature encoding methods in order to find the combination that produces the highest test score. From the results of these processes, we conclude that our solution to the problem presented by the competition is to train a machine learning model using the **LightGBM implementation of the gradient boosting algorithm**, whose parameters decided using grid search, on features that we selected from the feature selection process, with categorical features encoded using the **frequency encoding method**.

LINKS TO DATASET AND SAMPLES

Full dataset:
https://www.kaggle.com/c/microsoft-malware-prediction/data

Random balanced samples of 100,000 and 10,000 examples:
https://drive.google.com/file/d/1OMqvuSKF1jZbwf641xby1HYkb3BOfwMy/view?usp=sharing