Google Analytics Customer Revenue Prediction

1 Data Preprocessing

Under the observation of the whole dataset, some data fields with constant value, which has no help to our prediction are deleted; According to data analysis, some columns' value is not relevant with revenue, so these features are not included in our final training set. After preprocessing operation mentioned above, there were still 34 features left for next preprocessing.

After parsing of the JSON data, so many missing values have been found in the data set. Then, such values as not available in demo dataset, (not set), unknown. Unknown, (not provided) can be treated as NA value. In addition, in terms of missing value processing, several strategies have been applied in our project:

- Useless data with high missing rate would be removed directly;
- As for those features with clear means, fill the empty by domain knowledge;
- Treat them as a new type;
- Clustering;
- Fill with 0 for those numeric features

For example, according to figure 1, it is obvious that those blocks with black color means that their missing rate is about 100 %, in which case we should drop them at all. What's more, in the case of categorical features, the sample number of different categories is imbalanced, which will cause overfitting problem. So, we manually aggregate those categories with low percentage as a new category, "others" .

For example, in figure 2, you can see in Sweden, United Arab, New Zealand and some other countries do not have any consumption records, so we manually aggregate these countries categories with low percentage as a new category, others" .

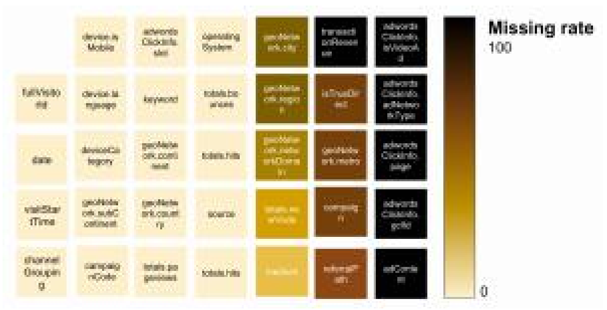

Figure 1: Missing Rate of Different Data

2 Feature Engineering

Now that the dataset is clean enough, we need to construct some statistical models to predict the result. When it comes to a statistical model, it serves as a function where both input and output are inevitable. To achieve a result with high confidence level, we need to find some inputs which make great contributions to the model rather than bringing noises to reduce the models generalization ability. So thats the meaning of feature engineering and we separate the feature engineering process into three parts.

2.1 Feature Extraction

First, we need to extract some features as the input. Though there are some features already given in the original data set, it is far too enough to construct a model only using these features. Because many of them have nearly no relevance with our target and thus make little contribution. What we focus on is to decompose these features and useful information through them. Apparently, the original data set can be separated into two dimensions, the one is temporal level and the other is user level. For

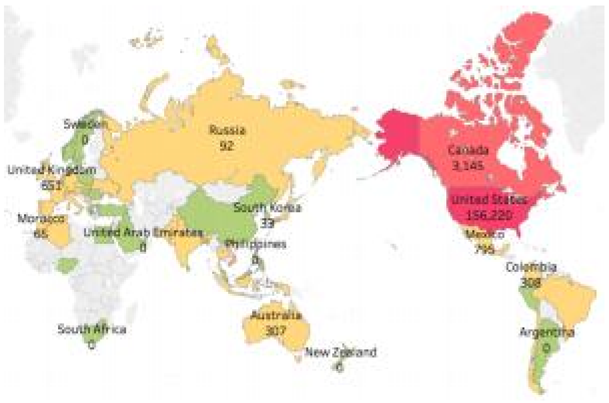

Figure 2: User DIstribution

the temporal level parts, we have the exact time of each transaction. And for the user level parts, we have user behavior information, geological information, device information, etc. When we plan to extract the feature, the auxiliary of figure is always helpful. Consider the below, we can see that the shopping behavior is totally different from morning to evening. In other words, customers tend to shopping online in the evening. So we can construct the feature hour of the day to make the predication. As the same, from figure 4, we can construct the feature day of week to extract the high relevance with our target. Besides, holiday is also a very important feature as the transaction numbers will rise sharply on that day.

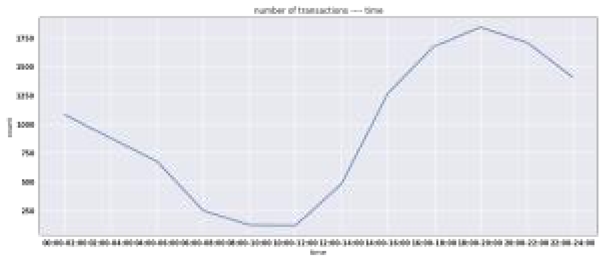

Figure 3: Temporal Features–Hours

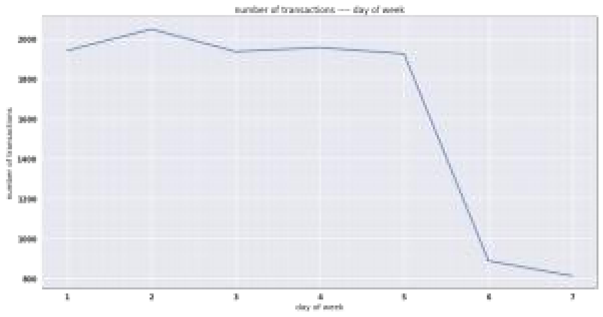

Figure 4: Temporal Features–Day of Week

For the user level part, we construct the feature like the time gap between two visiting, average transaction revenue, user location, and so on. All of them have their meanings. Take the example of the time gap between two visiting, if the time gap is small, that means this customer has a great interest on the shopping items and the probability of transaction will certainly grow up.

2.2 Feature Merging

In this part, we can merge some meaningful features into a more important feature. For example, we merge the district feature and operating system feature to form a new district-system feature. In this feature, people from the US who use the IOS system have great shopping interest. So this new feature has a great value. Through the feature merging, we can not only reduce the number of features and save time for tree models to split nodes, but also enhance the accuracy of predication.

2.3 Feature Selection

Now that we have plenty of features, we need to do the prune procedure. That is to say, we need to discard some features to reduce the noise and thus enhance our models generalization ability. Overall, there are totally four rules for us to select our features. The first rule is based on our understanding of the feature. we drop some geological feature like city and town as we already have the longitude and latitude information. As we all know, the latitude and longitude are the perfect representatives of the geological information. The second rule is based on the correlation coefficient. If the correlation coefficient between the feature and the target is nearly 0, that means this feature has nothing to do with the target and will make little contribution to constructing the model. So we will discard the features which has 0 correlation with the target.

The third rule is based on the feature importance. And the feature importance figure is generated by the baseline LightGBM model. From the figure, we can see that some of the features have very low importance, and that means these features have little contribution to construct the tree model. Consider the Occams Razor Theory, we discard this sort of feature.

The last rule is based on the actual performance. We use forward feature selection method to select the model. Briefly speaking, forward feature selection contains the following procedure.

1. Start with the null model;
2. Find the best one-variable model;

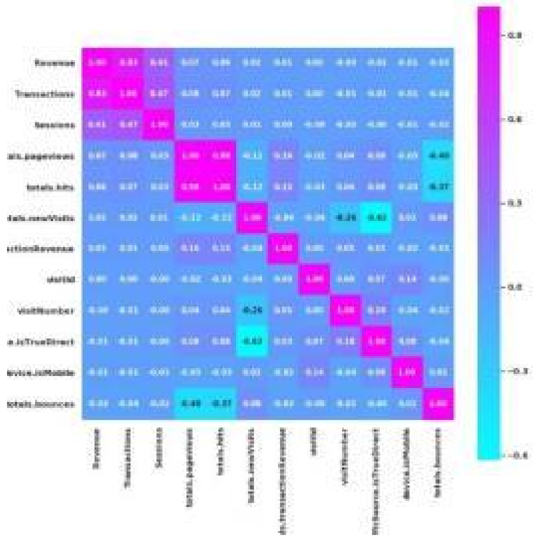

Figure 5: Heatmap of Correlation Matrix

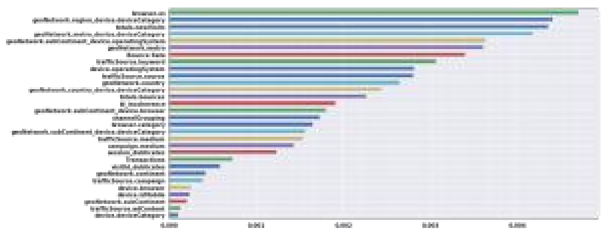

Figure 6: Feature Importance

3. With the best one variable model, add one more variable to get the best two-variable model;

4. ...

5. With the best (k- 1)-variable model, add one more variable to get the best k-variable model;

6. Repeat until the error no longer decrease;

7. Discard the features which do not participate in constructing the model

3 Model Construction

3.1 Random forest

Random forest is an extended variant of bagging. Based on the decision tree integration, RF further uses random attribute selection in the training process of decision tree. Specifically, the traditional decision tree selects an optimal attribute in the attribute combination of the current node when selecting the partition attribute; and in RF, for each node of the base decision tree, first A subset of k attributes is randomly selected from the attribute set of the node, and an optimal attribute is selected from the subset for division. The diversity of the basic learners of random forests comes not only from sample perturbations, but also from attribute perturbations, so that the generalization performance of the final integration can be further increased by the increase in the degree of difference between individual learners.

3.2 Gradient Boosting Decision Tree(GBDT)

Boosting tree actually uses an additive model (ie, a linear combination of base models) and a forward step-wise algorithm. A boosting method based on a decision tree is called a boosting tree. The decision tree for classification or regression problems is a binary classification tree. The lifting tree model can be expressed as the addition model of the decision tree: $f_M(x) = \sum_{m=1}^{M} T(x;\theta_m)$, where $T(x;\theta_m)$ represents the decision tree; $_m$ is the parameter of the decision tree; M is the number of trees.

The Boosting tree algorithm uses a forward stepping algorithm. First determine the initial lifting tree $f_0(x) = 0$, the model of the m-th step is $f_m(x) = f_{m-1}(x) + T(x;\theta_m)$, where $f_{m-1}(x)$ for the current model, the parameter θ_m of the next decision tree is determined by minimizing the empirical risk.

Assuming a squared error loss function,

$$L(y a f(x)) = (y - f(x))^2 = (y - f_{m-1}(x) - T(x;\theta_m))^2 = (r - T(x;\theta_m))^2$$

For the Boosting tree algorithm of the regression problem, it is only necessary to simply fit the residual of the current model. However, for the general loss function, it is often not easy to optimize each step. For this problem, Freidman proposed a gradient Boosting algorithm, which is an approximation method using the steepest descent method. The key is to use the negative gradient of the loss function at the current value of the model. Approximate values of the residuals in the regression tree as a regression problem fit a regression tree.

$$r_{mi} = -\left[\frac{\partial L(y, f(x_i))}{\partial f(x_i)}\right]_{f(x)=f_{m-1}(x)}$$

Compare random forest and gradient Boosting tree:

1. At the algorithm level, the random forest constructs training samples by randomly sampling the data set, and it is considered that the randomization is beneficial to the generalization performance of the model

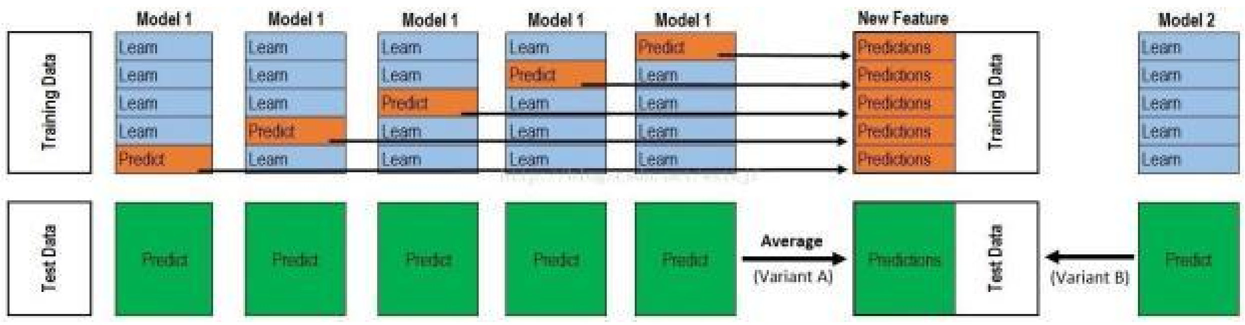

Figure 7: Results of Different Models

on the test set. The gradient Boosting tree finds the optimal linear combination of all decision trees based on the training data.

2. Random forests are easier to train than gradient-Boosting trees. Random forests only need to set a hyperparameter. The number of features randomly selected at each node is set to the total number of features of log or the square root of the total number of features in most cases. Made a good result. The gradient Boosting tree parameters include the number, depth and learning rate.

3. Random forests are more difficult to fit than gradient-boosting trees. The gbdt is sensitive to noise anomalies. From the perspective of deviation and variance, if the data is noisy, the boosting algorithm may exhibit a higher model variance. However, in other cases, the boosting algorithm can often achieve better results. However, random forests do not build model based on model residuals, and often achieve very low variance.

Because of the good performance of GBDT, we use lightbgm, xgboost and catboost as our models, and check the performance of the results. Then we use stacking method to combine the 3 models together.

3.3 Stacking (see Figure 7)

The upper part is a 5-fold cross-validation using a basic model. For example, catboost is used as the base model Model1. The 5-fold cross-validation is to take four folds as training data and another fold as testing data. Each cross-validation consists of two processes, first training the model based on training data; second predicting the testing data based on the model generated by training data training. After the first cross-validation is completed we will get a prediction about the current testing data(A1) and training data(B1).

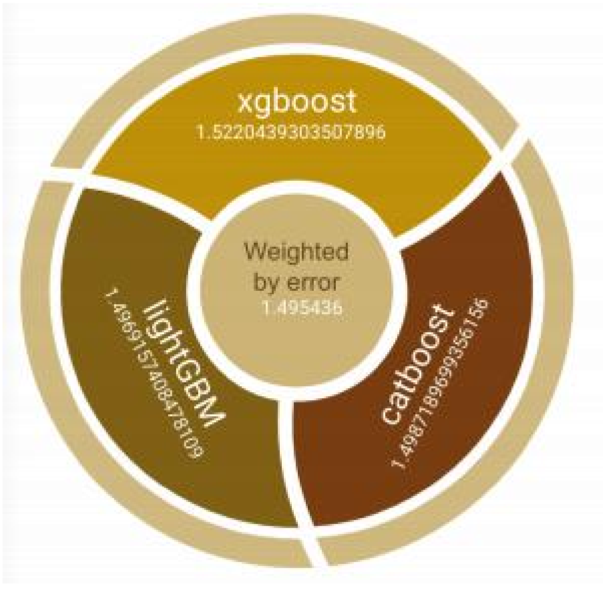

Figure 8: Results of Different Models

The same layer in stacking usually contains multiple models. Assume that there are Model2: lightgbm, Model3:xgboost. For these 3 models, we can repeat the above. Steps, after the end of the process, we can get the new A2, A3, B2, B3.

After that, we combine A1, A2, A3 side by side to obtain a matrix as training data. B1, B2, B3 are combined side by side to obtain testing data. Let the next layer of the model, based on their further training.