Outbrain Click Prediction

Problem Description And Overview Of Approach

In this challenge, we are predicting which pieces of content users are likely to click on to satisfy more users' individual tastes. Each display id is assigned with several ad id while users get access to the display id using different platforms from all around the world and they would decide whether clicking the ad or not. Our goal is to display the ad in the descending order of their likelihood to be clicked.

As there are hundreds of millions of online users and millions of ads, few of the ads will be clicked by the users. Our dataset is high dimensional and sparse. Most of the data are categorical. The clicked probability is required in the results. Based on these characteristics, we will use logistic regression, naive bayes, random forest, extreme gradient boosting to train the model.

Feature Selection

First, we calculate the frequency of the features and delete the features appears with low frequency.

In this part, we tried to figure out the most suitable features to build the model. In order to make decisions, we have to do some initial analysis for all the data files. File events includes many features can be encoded as features, such as timestamp, platform and geolocation. However, the user ID isn't suitable to be a feature because not only 92.98% users appear less than 2 times, but it is also low effective if we record each user's click rate. We choose to ignore the user id column.

When it comes to files with respect to document ID, features in these files are high dimensional and sparse, which means it is difficult to include them in the model. What's more, after merging tables, we found there are too many Nan values in the test dataset and 85% of documents appear less than 10 times. To process these data will be of low efficiency and have high requirements for GPU, memory and storage. We can't find an easy approach to deal with them as the total dataset is 36 GB. In this situation, we decided to drop these files.

Then, we chose ad_id, campaign_id, advertiser_id, document_id, timestamp, platform and geo_location as our predictors, clicked as our prediction to fit the model. At last, we calculate the importance of each feature in the model and dropped 4 features not important.

Data Cleaning

Based on previous steps, we already had several predictors. After joining them into a single table, firstly we move all the Nan values found in the geo_location by replacing them with string which represents Not in_US. Then, we encoded this feature as 1 if the location is in the US, 0 otherwise. For timestamp, it was divided into three parts, we used 1 to represent 4 am to 12 pm, 2 to represent 12 pm to 8 pm, 3 to represent 8 pm to 4 am. The platform also needs modifying. We replaced '\N' value with corresponding platform

number and transformed string inputs into integer inputs. Other features such as doucument_id and campaign_id have so many unique values that we couldn't simply use one-hot encoding, because the dimension of training and testing datasets will be extreme lager. In this case, MinMax scaling was applied to them to normalize the range of these features.

Model Selection

We built several models by using the cleaned data and chose the best one to get the prediction of the test dataset. For logistic regression model , random forest model and extreme gradient boosting model, we applied grid search to find the best hyper parameters while Naive Bayes model doesn't have hyper parameters, so we didn't apply grid search to it. Using these models, we applied it to our training data that was split into training and test set. We obtain the accuracies for training, cross validation, and test for each of these models. After comparing the results, the best model we found is extreme gradient boosting.

We then use our best model and fit the whole training data to generate the feature importance scores of each feature. We decided to remove four of our features due to low importance scores when compared to the other features (Time_of_day_3, Time_of_day_2, platform_3, and document_id)

Prediction

Before applying the extreme gradient boosting model, we first applied the same procedure to clean the test dataset and removed the four unimportant features. Then, we applied the model to predict the probability of each ad being clicked by each user. After sorting the list of ads of each user with the highest probability of being clicked, we submit to Kaggle and obtained a score of about 60% in Mean Average Precision @ 12.

Evaluation

Submissions are evaluated according to the Mean Average Precision @12 (MAP@12):

$$MAP@12 = \frac{1}{|U|}\sum_{u=1}^{|U|}\sum_{k=1}^{min(12,n)} P(k)$$

where |U| is the number of display_ids, P(k) is the precision at cutoff k, n is the number of predicted ad_ids.