TalkingData AdTracking Fraud Detection Challenge

Mengrui Yin

July 2018

1 Introduction

Fraud risk is everywhere, but for companies that advertise online, click fraud can happen at an overwhelming volume, resulting in misleading click data and wasted money. China is the largest mobile market in the world and therefore suffers from huge volumes of fraudulent traffic. TalkingData is the largest independent big data service platform. 90% of 3 million clicks per day are potentially fraudulent. They turned to the Kaggle community for help in further developing their solution to recognize clicks which could be potential fraudulent. They have provided a large dataset with 200 million clicks over 4 days. Our task was to build an algorithm which can help predict whether a user will download an app after clicking an app ad. AUC or ROC were used to evaluate our models.

2 Data

TalkingData have provided 5 data files: train.csv, train_sample.csv, test.csv, sampleSubmission.csv, UPDATE: test_supplement.csv. The training data had 184,903,890 rows and the test data had 18,790,469 rows. Each row of the training data contained a click record, with the following features. Features ip, app, device, os and channel are encoded. The test data was similar, with the following differences: click_id was reference for making predictions and is_attributed and attributed_time were not included.

Only feature attributed_time in the training data had 99.75% missing values. The reason for the missing values was the users didn't download the app after these clicks. There was no missing value in the test data.

3 EDA

Features ip, app, device, os and channel were categorical variables but they were encoded as integers. So I converted them to categorical variables for analysis. Table2 was the number of unique values for features in the training data. I

Table 1: Data fields

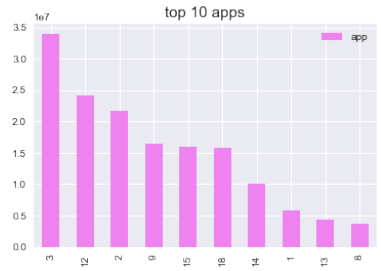
Feature name	explanation
ip	ip addresses of click
app	app id for marketing
device	device type if of user mobile phone(iphone 6 plus, iphone 7)
os	os version id of user mobile phone
channel	channel id of mobile and publisher
click_time	timestamp of click(UTC)
attributed_time	this is the time of the app download
is_attributed	the target that is to be predicted

Table 2: Number of unique values per feature

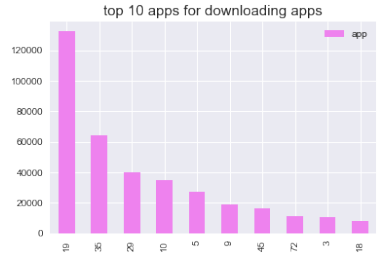
Feature name	Number of unique values
ip	277396
app	706
device	3475
os	800
channel	202
click_time	259620

found top10 for each feature with the highest number of clicks and also top10 for each feature with the highest number of downloads. For the specific value in the feature, it might have large number of clicks but have small number of downloads. Figure1 was app analysis. The left plot was top10 apps which had the highest number of clicks and the right plot was top10 apps which had the highest number of downloads. We could see the app which had large number of downloads were different from the app which had high number of clicks. Also for each app, I calculated the ratio of downloading and the ratio of not downloading and compared them. If the two ratios had large difference, it meant this feature might be helpful for predicting downloads. Figure2 was the ratios for 10 apps which were randomly selected from the training data. From the plot, we could think app was helpful. I did these analysis for ip, app, device, os and channel.

Then I did analysis for click_time. The start time for the training set was 2017-11-06 14pm and the end time was 2017-11-09 16pm. The training data covered four days. Figure3 was the distribution of click days. 6% clicks happened on 11-06 and around 30% clicks happened for each of the other three days. The reason the number of clicks in 6th was so low was that our training data for that was from 14pm to 12am, only 10 hours. Also, the data for 11-09 was from 0:00 am to 16pm, covering for 16 hours. So, the click numbers for 11-09 was



(a) top10 apps with highest number of clicks



(b) top10 apps with highest number of downloads 2

Figure 1: app analysis

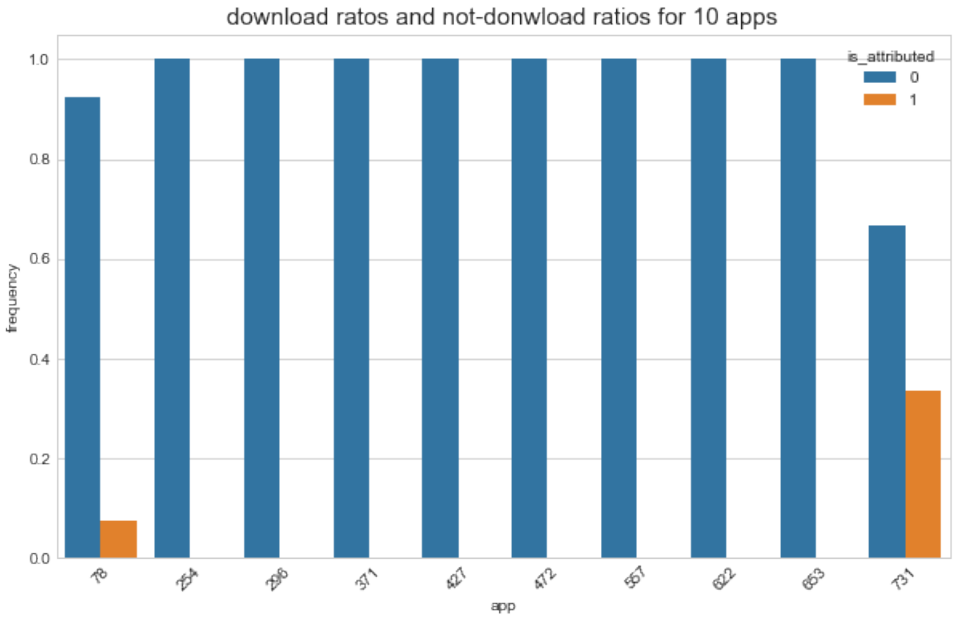


Figure 2: Download ratios vs not$_d ownloadratios for 10 apps$

little fewer than 11-07 and 11-08. The testing data collected information from 2017-11-10 4am to 2017-11-10 15pm.

For the target variable, is_attributed, it was the binary variable: 1 represented the user downloaded the app after clicking the app ad and 0 otherwise. Checking with the training data, 99.75% of data had value 0 and only 0.2471% data had value 1. So this was highly imbalanced data and we should deal with it carefully.

the variable 'attributed_time' was 100% correlated with the response variable 'is_attributed'. Also in the test data, we didn't have this variable. So I removed

Figure 3: Distribution of click days for the training data

it.

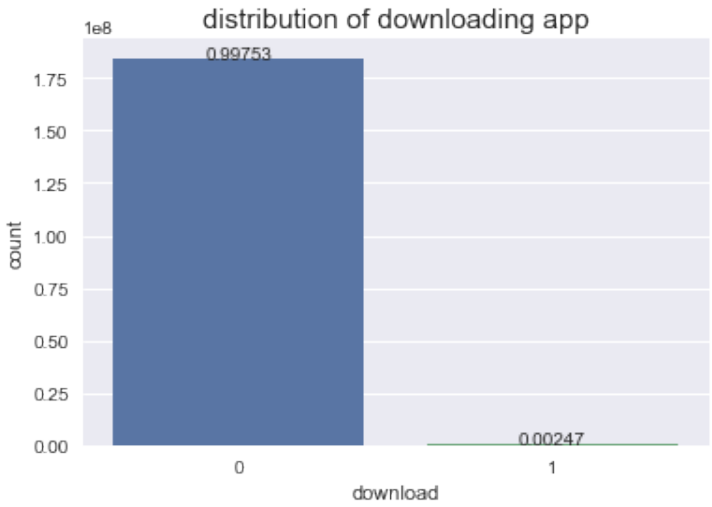
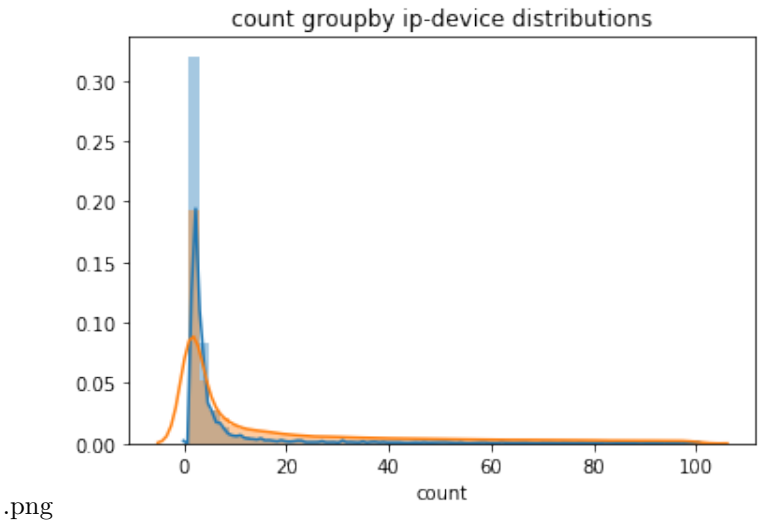


Figure 4: Distribution of is_ attributed for the training data

4 Feature Engineering

We only had 6 features which could help us for building model, so we should create more features based on the combination of the training data and the test data. We had more than 200 million data for the combination, which couldn't fit into 16G RAM. So I selected data which had the same click hour as the test data. The test data had click hours 14, 15, 4, 5, 6, 9, 10, 11, 12, 13. So we selected data from the training data which had these click hours and combined them with the test data together. Then our data reduced to 118,123,128 rows and I would create features based on them.

First I extracted day, weekday, hour, minutes and seconds from click_time. Then, I planned to create 3 types of features: count features, unique count features and previous click time. We could find many feature combinations from these 6 features, so it was hard to decide which new feature could be helpful in the model. What I did was listing some features I thought could be helpful based on some common-sense and compared its distribution in downloading data to its distribution in never downloading data. If these two distributions were different, then I would include this feature in my model. For example, I wanted to check the click numbers grouping by ip and device. First I selected the combinations of ip and device which were used for downloading app at least once and plotted the distribution of click numbers for groups. Then I selected the combinations of ip and device which were never used for downloading apps and also plotted the click numbers distribution for these groups. In the third step, I compared these two distributions. I found these two distributions had a little difference, so I kept this feature. By using this method, I created 28 features.



.png

Figure 5: count groupby ip-device distribution

5 Model

This was a binary task. I used gradient boosting model for two reasons. It was fast so it would be more efficient for such large data. In addition, it had parameters 'is_unbalance' and 'scale_pos_weight which could deal with the imbalanced problem. During around 100 million data, 65,128,309 rows were from the training data and the remaining 18,790,469 rows were the test data provided by the TalkingData for the final prediction. 65,128,309 rows were still a large number for 16G RAM to fit a model. So I selected data with day 8 and 9 from 65 million data. To train the model, I split them into the training set and the test set(to differentiate this testing set and the test set provided by the TalkingData, I called validation set here instead of the testing set.) The training set were from 65 million data whose click day were 11-08 and the validation set were data whose click day were 11-09. Then there were around 30 million data in the training set and 30 million data in the validation set. I treated ip, app, device, os , channel and hour as categorical variables. The 'scale_pos_weight' was set as 200 because our data was highly imbalanced. The auc score was 0.969577 for the validation set and 0.9643514 for prediction. I kept improving my model from two aspects: we had almost equal number of rows in the training set and the validation set. We could increase number of data in the training set which could help increase accuracy. Also, there were some features that weren't helpful for prediction since checking by feature importance, their importance values were 0. Then I built the second model with training 75% of 65 million data and using the remaining 25% data to test the model. The auc score for the validation set was improved to 0.975945 and for the prediction was improved to 0.9657776. Then the next step would be changing some useless features.

For my second feature engineering, I created additional two types of features: next click time and cum-count. I created 7 more features and used them to replace features that weren't helpful in the first model. In this model, we had 38 features. The auc score for the validation set was 0.980598 and for the prediction was 0.9723249! So the new features were really helpful! Also checking feature importance, only one feature 'dido_first' had value 0, so I considered to create other features to replace it for further improvement. Before continuing creating features, I just found that most of my features were created based on the feature ip, so I believed these new features had provided enough information about ip addresses and thus there was no need to keep feature ip in the model. In my fourth model, I removed the feature ip and it was truly helpful. The auc score for the validation set was improved to 0.981753 and for prediction was improved to 0.9734109, a little better than the model with feature ip.

For further improvement, I created 3 mean-value features and variance-value features and used them to replace the feature 'dido_first', which was useless in the model. For this model, I had 39 features. The auc score for the validation set was 0.981744 and for the prediction was 0.9735262 with these three new features. All 39 features were important in the model. Figure5 reported the importance value for each feature. From the plot, channel, os and app had really high importance values. Os should be important. Os version was supposed to

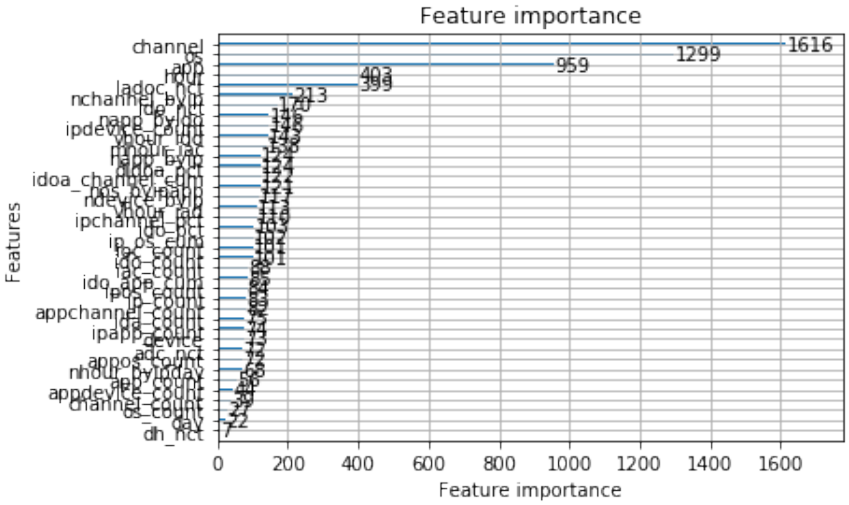


Figure 6: Feature Importance

be compatible with the app ads so that the user could view the ads they were interested in. App was also a key factor. There were surly some popular apps that had higher download numbers than others, such as fitness app, the app that was related to or helpful to our daily life. Since we didn't have user definition in the data, we were supposed to build user concept by ourselves. Then we could create features based on the user and learned some patterns for each user. I used the combination of ip, device and os to define a user. So some features based on user concept could provide information about the specific user. For example, we could know the number of clicks of the user which could help us to think whether the user had intention to download. More specifically, adding app and channel information on users would be more helpful. For example, for specific app, how many times the user had viewed. For each user, if there were one or more apps they were more interested in, then the click numbers for this app could be higher. Except the single user, I also came up with 'company concept', which was defined by ip address. Treating each ip address as a company, for each ip address they might have preferred apps or preferred channels. In addition, some features were created based on the combination of app and channel. There could be preferred channel for each app, which meant for the specific user, if they viewed the ad through that channel, the probability of downloading could be higher. This rule was validated by checking data. Previous click time and next click time were also useful. If a user wanted to find an app and download, the time gap between their clicks might be different from the clicks of users who had no intention to download the app. I treated this model as the final model. This model was built using 45,776,522 rows in the training set and 15,258,841 rows in the validation set. There were 39 features. Due to 16G RAM of my computer, I couldn't use more rows to train the model or add more features in

the model.

6 Conclusion

In this project, feature engineering was the key to improve your model accuracy. I did three times feature engineering and each time these new features could help me improve my auc score. I built 5 models and table3 was the summary of these five models. I didn't tune parameters in this project. Due to the size of the data, it was time-consuming to tune parameters. I believed tuning parameters could help me improve a little bit more.

Table 3: Data fields

model	auc for validation set	auc for prediction
model1	0.969577	0.9643514
model2	0.975945	0.9657776
model3	0.980598	0.9723249
model4	0.981743	0.9734109
model5	0.981744	0.9735262