TalkingData AdTracking Fraud Detection

Machine Learning Engineer Nanodegree Capstone Proposal

Pooya Rezaei
March 25, 2018

Introduction

Click fraud, which is clicking on advertisements with no real interests in the product or the offered services, is an important problem in online advertising that leads to wasted money for the service providers. TalkingData, China's largest independent big data service platform handle three billion clicks per day, of which about 90% are potentially fraudulent [1]. For this reason, TalkingData has turned to Kaggle community and defined a competition to detect fraudulent clicks. Their current approach is to define an IP and device blacklist. The goal is to improve this approach by detecting fraudsters using all available features of clicks.

Problem Statement

The challenge is to build an algorithm that predicts clicks that lead to downloading the apps. So this is a binary classification problem that uses all available features of the clicks. Many machine learning algorithms solve binary classification problems including Logistic regression, SVM, deep learning, bagging and boosting methods. Among these, ensemble methods especially boosting algorithms have been very promising in recent Kaggle competitions.

Datasets and Inputs

TalkingData has provided a dataset covering approximately 200 million clicks over four days as the training set. The goal is to predict fraudulent clicks in the testing set, which includes clicks on a fifth day. The data is available in the competition website on Kaggle. For each click in the training set, the following features are available:

- ip address of the click
- app id for marketing
- device type id that clicked (e.g., iphone 6, Huawei mate 7, etc)
- os version id of mobile phone
- channel id of mobile ad publisher
- click time
- download time (if the user downloaded the app)

- is_attributed indicating the app was downloaded, which is the target that is to be predicted

For privacy reasons, ip, app, device, os and channel are all encoded.

Benchmark Model Solution

One solution to the problem is to use Random Forest algorithm with the given features. There is potentially a lot of work that can be done in this project for feature engineering using the user ip and click time. However, for the base benchmark model we are going to use ip, app, device, os and channel as the features and train a random forest algorithm. These are all encoded categorical variables. The typical approach is to use one-hot encoding of the categorical features, but because of many unique values for each feature we are not going to do that here, and will use the features as they are. Figure 1 shows the number of unique values for each feature.

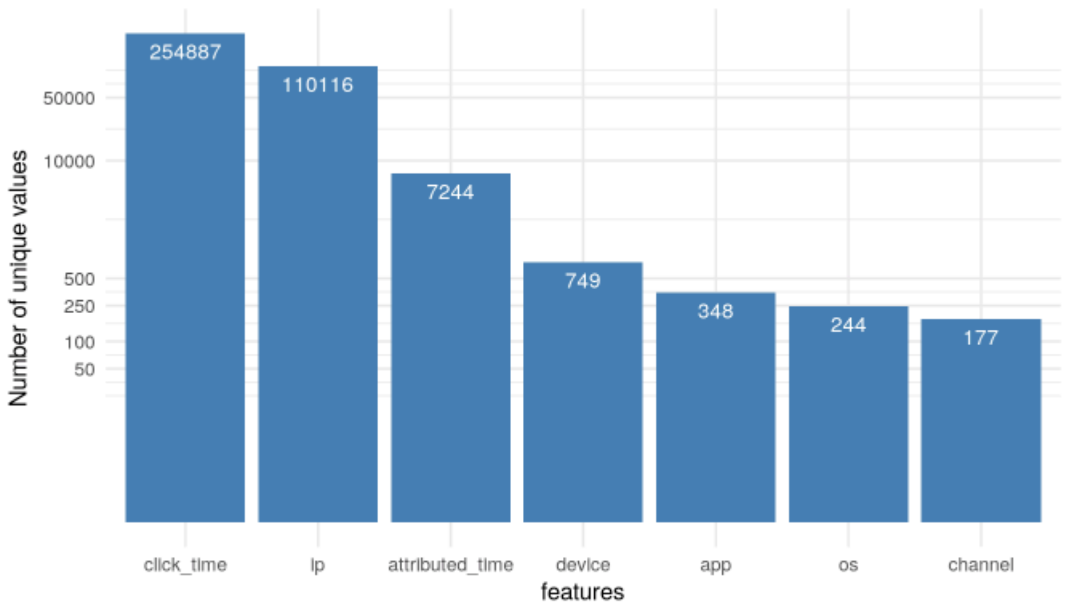

Figure 1: number of unique values for each feature [2]

There are about 200 million rows of data in the training set, and because of this high volume we have trained the benchmark model with only 4 million ending rows of the training data, and used 10% of the data for validation. We have trained a random forest algorithm with this data with max_depth of 10 and other default values in the sci-kit learn package.

Evaluation Metrics

There are many ways to evaluate the results of a binary classification problem like the one here. These include accuracy, precision, sensitivity/recall, F1 score and Area Under the ROC Curve (AUC). The first four metrics are all affected by class imbalance that exists in this problem. However, AUC is not affected by class imbalance, and so we are going to use that as the evaluation metric. This is also the metric that the competition is going to be evaluated on. The training AUC is 0.97565 and the evaluation AUC is 0.97562, which is very close to the training data AUC value. After we submit the test data result to the competition website we got the Leaderboard AUC score of 0.9288, which means that we are overfitting to the training data. Although the model generalizes very well to the evaluation data, it does not generalize to the test data. One reason for this difference is perhaps because the training set and the evaluation set are both data for one day, but the testing set has data for a different day.

Project Design

As mentioned before, the dataset for this project is quite large. There are about 200 million rows of data in the training set (~ 8 GB). Loading all the data is a challenge. I will test models with partial data on my personal computer, and run a final model using all the data on Amazon AWS. As mentioned before, boosting algorithms have been shown to be very effective for binary classification in previous Kaggle competitions, so the LightGBM package will be considered in this project that is based on Gradient boosting trees. The training data will be split to training and evaluation sets.

No particular data cleaning is required. LightGBM can take categorical features directly and there is no need to do one-hot encoding on the categorical features either. However, I need to do feature engineering based on the available features in this project. For example, the number of clicks in each hour of the day from the same device can be added as a feature. Other features engineered from a combination of ip, click time and device/app/os will be used in this project.

An idea to consider is to use evaluation data from a different day than the training set data. This way, we can evaluate how the algorithm trained on data for one day can generalize to the data for another day. This might be beneficial as the testing data is from a different day than the training data, and the benchmark model showed that using the training data might include some overfitting.

References

[1] TalkingData AdTracking Fraud Detection Challenge, Kaggle. URL : https://www.kaggle.com/c/talkingdata-adtracking-fraud-detection.

[2] TalkingData EDA and Class Imbalance, Kaggle kernel. URL:
https://www.kaggle.com/kailex/talkingdata-eda-and-class-imbalance