

TalkingData: AdTracking Fraud Detection Challenge

Baseline Implementation

Lixuan Mao (lm769), Ang Li (al2386)

Abstract

In this report, full and detailed descriptions about the challenge and relating datasets will be given. Besides, baseline was implemented based on basic classification algorithms including Logistic Regression and several Ensemble Methods. In the meanwhile, how the experiments were conducted and the relating results and analysis will also be described in detail.¹

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. Ad channels can drive up costs by simply clicking on the ad at a large scale. With over 1 billion smart mobile devices in active use every month, China is the largest mobile market in the world and therefore suffers from huge volumes of fraudulent traffic.(TalkingData, 2018)

In this challenge, we have the datasets from TalkingData, China's largest independent big data service platform which covers over 70% of active mobile devices nationwide. Their current approach to prevent click fraud for app developers is to measure the journey of a users click across their portfolio, and flag IP addresses who produce lots of clicks, but never end up installing apps. With this information, they've built an IP blacklist and device blacklist.

While this method was successful, one could be always one step ahead of fraudsters if leveraging the power of machine learning algorithms. In this challenge, we are going to build models that can predict whether a user will download an app after clicking a mobile app ad.

Basically speaking, the challenge is to build classification models that tell if a user is going to download an app given several features of the clicking. And in the meanwhile, the models should also give the probability of a user is to downloading an app after clicking.

2 Related Work

Consider this is a Kaggle competition, many smart people has been contributed to this project. Those people also build their baseline model and improve their performance by adding more feature-wise work. Logistical model is very suitable for binary classification, and we found one published kernel using **xgBoost** and Bayesian optimization to improve the performance and finally reached 97.8 % accuracy on leader board. The main idea of his approach is doing a script that trying out different combination of xgBoost Bayesian hyper parameters to get the optimized combination. Consider the highest rank on the Kaggle competition is 98.18% this is a very efficient to increase the performance. (NanoMathias, 2018)

3 Baseline Model Description

In the baseline implementation, three types of model are employed: one is Logistic Regression, one is Support Vector Machine(SVM) and the last one is Ensemble Methods(Liaw et al., 2002).

3.1 Logistic Regression

Logistic Regression is widely used in various kinds of regression and classification tasks and is known for its simplicity and usefulness, whose characteristics are quite suitable to be used as baseline.(Agresti, 2002)

In the Logistic Regression model, L2 penalty is used as loss function and lib-linear algorithm is used to solve the optimization problem. As an

¹Baseline implementation codes can be found [here](#).

optimization problem, binary class L2 penalized logistic regression minimizes the following cost function:

$$\min_{\omega, c} \frac{1}{2} \omega^T \omega + C \sum_{i=1}^n \log(\exp(-y_i(X_i^T \omega + c)) + 1). \quad (1)$$

Addition to that, in order to reduce the unbalance of class data in the dataset, class weights were assigned to each class by the following formula:

$$\frac{n_samples}{n_classes * np.bincount(y)} \quad (2)$$

3.2 Support Vector Machine(SVM)

Support vector machines (SVMs)(Joachims, 1998) are a set of supervised learning methods used for classification, regression and outliers detection.

The reasons why SVM is suitable for this task are:

1. Uses a subset of training points in the decision function (called support vectors), so it is also memory efficient.
2. Versatile: different Kernel functions can be specified for the decision function. Common kernels are provided, but it is also possible to specify custom kernels.
3. Effective in both high and low dimensional spaces.

The loss function is still L2 penalty. And the kernel function used in this implementation is linear.

3.3 Ensemble Methods

The goal of ensemble methods is to combine the predictions of several base estimators built with a given learning algorithm in order to improve generalizability / robustness over a single estimator.

Two families of ensemble methods are usually distinguished:

In averaging methods, the driving principle is to build several estimators independently and then to average their predictions. On average, the combined estimator is usually better than any of the single base estimator because its variance is reduced. Examples of this family are Bagging methods and Forest of randomized trees.

By contrast, in boosting methods, base estimators are built sequentially and one tries to reduce

the bias of the combined estimator. The motivation is to combine several weak models to produce a powerful ensemble. Examples are AdaBoost, Gradient Tree Boosting, etc..

In the baseline implementation of Ensemble Methods, methods from both families were adopted, which were Bagging methods, Random Forests, Extremely Randomized Trees, AdaBoost and Gradient Tree Boosting.

The detailed parameters of these models can be found in the codes provided.

4 Dataset Description

Following files are available:

File	Description	Size
train.csv	training set	92490470
train-sample.csv	sample training	100000
test.csv	test set	18790470
test-supplement.csv	more test	38790470

Table 1: Files

Each row of the training data contains a click record, with the following features:

- ip: ip address of click.
- app: app id for marketing.
- device: device type id of user mobile phone (e.g., iphone 6 plus, iphone 7, huawei mate 7, etc.)
- os: os version id of user mobile phone
- channel: channel id of mobile ad publisher
- click-time: timestamp of click (UTC)
- attributed-time: if user download the app for after clicking an ad, this is the time of the app download
- is-attributed: the target that is to be predicted, indicating the app was downloaded

The test data is similar, with the following differences:

- click-id: reference for making predictions
- is-attributed: not included

The datasets were heavily unbalanced which the count of **0 label** contribute more than 99% of the data and **1 label** remains 1%. Although this distribution matches the intuitive thinking since most of the users won't download app after clicking ADs, different class weight became necessary in this scenario.

On the dataset side, we have two version of the baselines. The first baseline is that using only train-sample.csv to complete the model, which contains limited information and it can reach 90% accuracy on test set. And we also have another version, which took 12 hours to finish training on the whole training set and reach 95.2% accuracy.

5 Experimental Setup

5.1 Preprocessing

Most of the data has been already encoded by data provider, however, many feature engineering could be performed in this stage to increase performance. As the baseline model, we only preprocessed the time-related features, which extracted original UNIX stand timestamps to five new categories: **month, day, hour, minute, second**.

```
pd.to_datetime(train_df.click_
//time).dt.second.astype('uint8')
```

More insight conclusions could be got from these data: people usually click ADs in midnight are less likely to actual download the APP maybe he/she is falling sleep at that time.

And we also believe that most of the performance increment would happen in this area with performing more interesting feature engineering.

6 Results and Analysis

In the baseline implementation, **train_sample.csv** was used to generate preliminary results. **train_sample.csv** was split into two parts:

train set(80%) and validation set(20%).

And several metrics were employed to measure the performance of each model. The results can be seen in Table 2.

From the results in Table 2, one can see that although SVM can achieve hight accuracy score, but it cannot give the probability of a user downloading an app after clicking. Besides, by comparison, Ensemble Methods, especially Random Forests, Ada Boost and Gradient Boosting have

generally better performance than others in term of Area Under Curve (AUC).

References

- Alan Agresti. 2002. *Logistic regression*. Wiley Online Library.
- Thorsten Joachims. 1998. Making large-scale svm learning practical. Technical report, Technical report, SFB 475: Komplexitätsreduktion in Multivariaten Datenstrukturen, Universität Dortmund.
- Andy Liaw, Matthew Wiener, et al. 2002. Classification and regression by randomforest. *R news* 2(3):18–22.
- NanoMathias. 2018. *Bayesian Optimization of xgBoost — LB: 0.9769*. <https://www.kaggle.com/nanomathias/bayesian-optimization-of-xgboost-lb-0-9769/comments>, US.
- TalkingData. 2018. *TalkingData AdTracking Fraud Detection Challenge — Kaggle*. www.kaggle.com/c/talkingdata-adtracking-fraud-detection, Beijing, China.

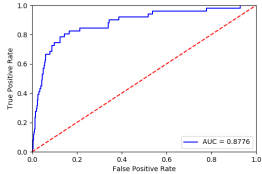
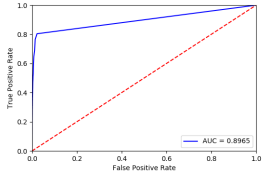
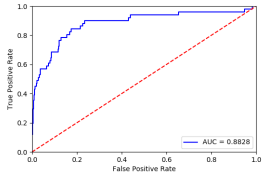
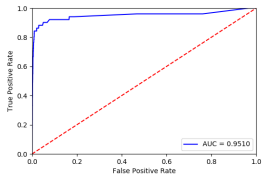
Model Name	Accuracy Score	Matthews Corr Coef (MCC)	AUC	ROC
Logistic Regression	0.8196	0.08405	0.8776	
	SVM	0.9974	N/A	
Bagging Methods	0.9975	0.3888	0.8964	
	Random Forests	0.9724	0.2548	
Extra Trees	0.8900	0.09264	0.8828	
	Ada Boost	0.9973	-0.0005056	
Gradient Boosting	0.9974	0.3200	0.9509	

Table 2: Performance of Different Models