Click Fraud Detection

MLND Capstone Proposal
Shan Dou | June 30, 2018

---

1. Domain Background

Click fraud has been a "billion dollar" problem facing ***pay-per-click (PPC)*** advertisers. PPC is by far the most widely used compensation model for digital advertising (e.g., both Google AdWords and Facebook Ads are PPC platforms). As the name "pay-per-click" implies, PPC advertisers pay for every click on their ads. This compensation mechanism has clear benefits to advertisers because they don't need to pay for ads that don't generate clicks, but this same mechanism also is heavily abused by fraudsters through click fraud.

Click fraud is the ill-intentioned clicking of PPC ads by fraudsters to waste and/or to mislead advertisers' ad spending. According to a recent report by CNBC, click fraud cost advertisers $12.5 billion in 2016 and wasted nearly 20% of total ad spending. But the monetary loss is not limited to advertisers. Click fraud also hurts revenue streams for ad platforms because it degrades the overall appeal of digital advertising. As an example, the consumer giant Procter&Gamble slashed its digital ad spending by more than $200 million in 2017 . For the $200 billion market of digital advertising, the stakes for preventing click fraud are high, and this is where data mining and machine learning could come to help.

---

2. Problem Statement

The goal of the project is to build a click-fraud detector that serves ***ads platforms for mobile apps***. Quantitatively defining fraudulent clicks is a challenge in its own right. Existing studies have shown that fraud labels based on IP and device blacklists often are problematic as they are biased by the procedures used to generate those lists in the first place (e.g., Oentaryo et al., 2014). As a work-around, we employ the following simplication: **Clicks followed by app downloads are legitimate, whereas clicks that don't lead to downloads are fraudulent.** With this simplification in place, we can now frame the problem as a ***supervised learning*** problem, and more specifically, we are to construct a ***binary classifier*** for predicting whether or not clicks are followed by app downloads.

---

3. Datasets and Inputs

3.1 Data descriptions

The raw data consist of ***200 million*** clicks registered over 4 consecutive days (Nov. 6, 2017-Nov. 11, 2017). They are provided by China's independent big-data service platform TalkingData and are open to the public via a recent Kaggle competition. The data are structured and are in comma-separated values (CSV) format. Each row

of the data is comprised of the following fields:

1. `ip`: The IP address of the click
2. `app`: Advertising ID of the ad
3. `device`: Numeric ID assigned to various mobile devices (e.g., iPhone 6 Plus, iPhone 7, and Huawei Mate 7)
4. `os`: Numeric ID assigned to various OS versions used on the mobile devices
5. `channel`: Numeric ID assigned to various mobile ad channels
6. `click_time` (UTC): Timestamp of the click
7. `attributed_time`(UTC): When a click is followed by app download, this field records the app download time; it is left empty otherwise
8. `is_attributed`: `1` if the click is followed by app download; `0` otherwise. **This is the target value for the binary classifier**.

To keep the data anonymous, `ip`, `app`, `device`, `os`, and `channel` are all encoded and the encoding rules are not revealed to the public.

3.2 Data size considerations

This dataset is quite big (7GB and 823MB for training and testing data, respectively), which makes it cumbersome to use all the training records for exploratory data analysis (EDA) and model selection. To keep early stages of the project agile, a sample containing **1%** of the full-size records is generated via randomized sampling and subsequently used for analysis and experimentation.

3.3 Feature engineering considerations

The raw data contain very few directly usable features. First, 5 out of 7 of the raw features are categorical identifiers, and the amount of unique identifiers is too large for techniques such as one hot encoding to be applicable. For example, among the `1,849,038` randomly sampled records, the amount of unique values for the raw categorical features are: `n_ip = 96420, n_app = 323, n_device = 229, n_os = 569`. Second, timestamps, when used alone, do not appear to have sufficient discriminating power. **Figure 1** shows the distributions of the clicks as a function of click time. Although the magnitudes between `is_attributed = 0` and `is_attributed = 1` are markedly different, the shapes of the distributions are similar.

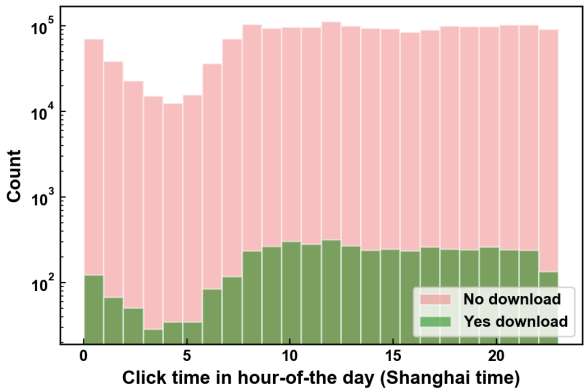

Figure 1: Distributions of clicks as function of click time.

These challenges call for extensive feature engineering. Here we treat `ip` as a primary identifier. We then combine the rest of the categorical features into groups of 2, 3, 4, and 5. As the last step, we tie these features and

feature combinations back to `ip` and count the amount of clicks belonging to each of the combinations. This feature engineering workflow gives us 31 additional features:

```
['0: count_ip-os', '1: count_ip-app', '2: count_ip-os-app', '3: count_ip-device', '4: count_ip-channel', '5: count_ip-os-device', '6: count_ip-os-channel', '7: count_ip-device-app', '8: count_ip-click_hour', '9: count_ip-channel-app', '10: count_ip-click_hour-os', '11: count_ip-os-device-app', '12: count_ip-click_hour-app', '13: count_ip-device-channel', '14: count_ip-os-channel-app', '15: count_ip-click_hour-device', '16: count_ip-click_hour-os-app', '17: count_ip-os-device-channel', '18: count_ip-click_hour-channel', '19: count_ip-device-channel-app', '20: count_ip-click_hour-os-device', '21: count_ip-click_hour-os-channel', '22: count_ip-click_hour-device-app', '23: count_ip-os-device-channel-app', '24: count_ip-click_hour-channel-app', '25: count_ip-click_hour-os-device-app', '26: count_ip-click_hour-device-channel', '27: count_ip-click_hour-os-channel-app', '28: count_ip-click_hour-os-device-channel', '29: count_ip-click_hour-device-channel-app', '30: count_ip-click_hour-os-device-channel-app']
```

---

4. Solution Statement

We will implement the fraud detector by solving a ***binary classification*** problem, and our binary targets are `1` for legitimate clicks and `0` for fraudulent clicks. To simplify the problem, these targets are further approximated by whether or not a click is followed by an app download. Because the amount of no-download clicks is much higher than its counterpart, we have ***highly unbalanced classes*** at hand that will require special care during model selections and evluations. In terms of machine learning algorithms, both linear (logistic regression) and nonlinear (e.g., random forest, extreme gradient boosting, and neural network) classifiers will be tested and evaluated, and the algorithm that best balances ***accuracy, performance, and interpretability*** will be selected.

---

5. Benchmark Model

Logistic regression is simple, fast, and easy to interprete. We will use it as the benchmark model both to evaluate the signal strengths of the input features and to compare its performance against those of more sophisticated models.

---

6. Evaluation Metrics

We will use the **A**rea **U**nder the reciever operating characteristic **C**urve (AUC) as the performance metric to evaluate machine learning algorithms. The closer the AUC score is to 1, the more accurate the predictions are.

---

7. Project Design

This project will likely require 15 to 20 hours of work distributed over a two-week period, and it will go through the following stages:

Stage	Expected outcome	Required time (%)	Progress(%)
	- Identify interesting patterns in raw		

1. Preliminary exploratory data analysis (EDA)	data - Identify directly usable features - Assess the need for feature engineering	5%	100%
2. Feature engineering + EDA on the resulting features	Generate enough usable features for subsequent machine learning steps	30%	90%
3. Model selection with cross validation	-Comparisons between logistic regression and other candidate models -Eliminate noisy and low-signal features, engineer additional features if needed -Select best model and establish end-to-end workflow	30%	0%
4. Incrementally include more training data and check model performance	- Obtain and inspect model learning curve (i.e., how model performance change as a function of data size) - Adjust model complexity if needed	5%	0%
5. Test model on test data, and assess fit for deployment	Decision on whether or not the final model is fit for deployment and evaluate the extent of overfitting	5%	0%
6. Summarize and report findings	Capstone project final report	25%	0%

References

1. Google AdWords: https://www.wordstream.com/articles/what-is-google-adwords
2. Facebook Ads: https://www.nytimes.com/2017/10/12/technology/how-facebook-ads-work.html
3. CNBC report: https://www.cnbc.com/2017/03/15/businesses-could-lose-164-billion-to-online-advert-fraud-in-2017.html
4. Wall Street Journal article about P&G slashing digital ad spending: https://www.wsj.com/articles/p-g-slashed-digital-ad-spending-by-another-100-million-1519915621
5. Digital advertising market size: http://www.strathcom.com/wp-content/uploads/2016/11/eMarketer_Worldwide_Ad_Spending-eMarketers_Updated_Estimates_and_Forecast_for_20152020.pdf
6. Oentaryo, R. et al. (2014) Detecting Click Fraud in Online Advertising: A Data Mining Approach. Journal of Machine Learning Research, 15, 99-140: http://www.jmlr.org/papers/volume15/oentaryo14a/oentaryo14a.pdf
7. Kaggle competition: TalkingData AdTracking Fraud Detection Challenge: https://www.kaggle.com/c/talkingdata-adtracking-fraud-detection