Supervised Machine Learning for Click Fraud Detection

MLND Capstone Project
Shan Dou | Jul 21, 2018

I. Definition

1.1 Project Overview

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[1], 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[2]. For the $200 billion market of digital advertising[3], the stakes for preventing click fraud are high, and this is where data mining and machine learning could come to help.

1.2 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[4]). 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.

Data for this project were click-traffic records provided by TalkingData, which is China's largest independent big data service platform. The raw data are accessible through a Kaggle machine learning competition titled "TalkingData AdTracking Fraud Detection Challenge"[5]. The full dataset consists of 200 million click records

and takes 7GB of memory. Such a big data size is not ideal for exploratory data analysis or for evaluating performance of machine learning models. In this capstone project, we focus solely on the effectiveness of data processing and machine learning pipeline, and to keep the operations lightweight, only **0.1%** of the click records are randomly sampled and used throughout this report (downsampling was implemented in `preprocessing.csv_randomized`).

1.3 Metrics

We use the ROC AUC score as the performance metric of the binary classiflier. A classifer no better than random guesses yields a score of ~0.5, and a perfect classifier has a score of 1.

The ROC AUC score meausres the **a**rea **u**nder of the reciever operating characteristics (ROC) **c**urve. The ROC curve is a plot of true positive rate (recall) versus false positive rate (1 - specificity), where the true positive rate ($TPR = \frac{TP}{TP+FN}$) measures the fraction of the positive instances that are correctly detected by the classfier, and the false positive rate measures the fraction of the negative instances that are incorrectly classified as positive ($FPR = \frac{FP}{FP+TN}$). In **Figure 1** below, the dashed line is the ROC curve of random guess, and the ROC curve of a good classifier should stay as far away from the dashed line as possible. In ideal cases, it should almost be touching the upper left corner of the graph.

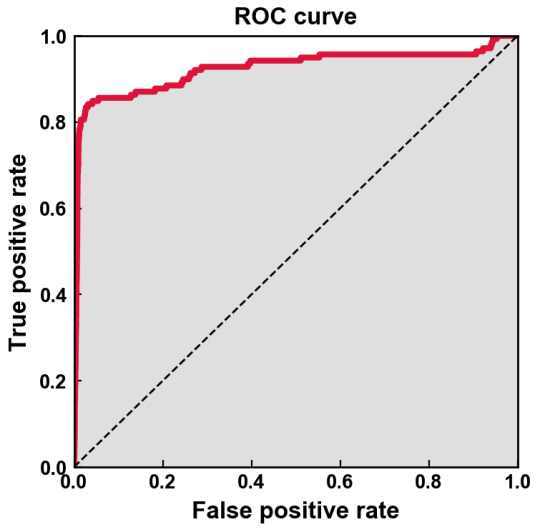

Figure 1: Example of receiver operating characteristic (ROC) curve. Shaded area denotes the area under the ROC curve (AUC).

Why use ROC AUC score as the metric? The ROC curve is a useful graphical tool that captures the tradeoff between recall and specificity: That is, the more `1`s we can capture, the more likely we also are to have missclassified `0`s as `1`s. An ideal classifier should help us capture as many `1`s as possible while misclassifying the least amount of `0`s as `1`s. From this perspective, when an ROC curve is almost touching the upper left corner of the graph, this very corner represents the desired situation of haivng high true positive rate and low false positive rate. The AUC score is built on top of this concept. It is the area under the ROC curve that quantitatively describes the curve's shape. The closer AUC is to 1, the closer the ROC curve is to the upper left corner of the graph, and relatedly, the more effective the classifier is.

II. Analysis

2.1 Data Exploration

The

2.2 Algorithms and Techniques

Logistic regression, random forest, stack ensemble, extreme gradient boosting, and light gradient boosting.

2.3 Benchmark

Logistic regression is used as benchmark.

III. Methodology

Data Preprocessing

Implementation

Refinement

IV. Results

Model Evaluation and Validation

Justification

V. Conclusion

Reflection

Improvement