Machine Learning Engineer Nanodegree

Capstone Proposal

Yoh-Hao Chang
question0802@gmail.com
March 3rd, 2018

Proposal

I choose this Kaggle competition,'TalkingData AdTracking Fraud Detection Challenge', as my final capstone project of Machine Learning Engineer Nanodegree. A quick description of this project is to create a model which can properly predicts whether a user will download an app after clicking a mobile app ad through applying the technique of machine learning.
Most of information can be directly found in the following link:
https://www.kaggle.com/c/talkingdata-adtracking-fraud-detection

Domain Background

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. In the paper, 'Detecting Click Fraud in Online Advertising: A Data Mining Approach' (The Journal of Machine Learning Research, v.15 n.1, p.99-140, January 2014), it summarize lots of observations and analyses of the fraud click detection. It also addressed some important issues in data mining and machine learning research, including highly imbalanced distribution of the output variable, heterogeneous data (mixture of numerical and categorical variables), and noisy patterns with missing/unknown values.

TalkingData, China's largest independent big data service platform, covers over 70% of active mobile devices nationwide. They handle 3 billion clicks per day, of which 90% are potentially fraudulent. Their current approach to prevent click fraud for app developers is to measure the journey of a user's 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.

Problem Statement

The problem is quite straightforward: how to know whether a user will really download an app after clicking a mobile app ad? That is, how to distinguish between meaningful clicks

and fraud clicks? Currently, TalkingData does have some methods to prevent click fraud. But there are still rooms for improvement. While successful, they can then always be one step ahead of those fraudsters.

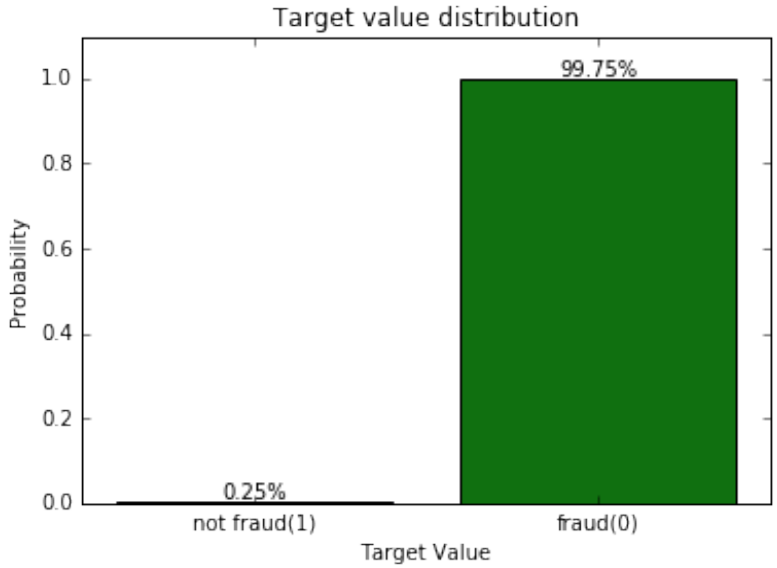
Datasets and Inputs

To support your modeling, they have provided a generous dataset covering approximately 200 million clicks over 4 days. All of the necessary data sets can be found and download from: https://www.kaggle.com/c/talkingdata-adtracking-fraud-detection/data

- **train.csv** - the training set
- **train_sample.csv** - 100,000 randomly-selected rows of training data, to inspect data before downloading full set

There are about hundreds million recorded data and each data entry is with 8 features. It contains some features like: ip address of click, app id for marketing, os version id of user mobile phone, device type id of user mobile phone, the target that is to be predicted, indicating the app was downloaded, etc. Some features of data are encoded already. Except of two features are recorded in UTC time format. The size of the data is really quite big. It maybe takes much time use the whole training data set and the whole testing data set for our model development and self-evaluation due to limited memory. We will further split the training data set up to three pieces: one for training, another for validation and the other for testing. Of course our final model should be further examined by the test data in the public Leaderboard of Kaggle.

In the reference paper, there is an critical issue that they do have a problem of the mobile advertising data are unique and complex, involving heterogeneous information, noisy patterns with missing values, and highly imbalanced class distribution. After a quick check of target value distribution as shown below, we do have a extremely imbalanced dataset. 99.75% of the dataset are labeled as fraud click and only 0.25% are not fraud click. Hence, we maybe need to include some data re-sampling strategies for handling this imbalanced label distribution. But My current plan is to establish a model with the original data. If the model shows bad performance in cross-validation, I will then take the re-sampling of data into account.



Solution Statement

Our goal is to develop an algorithm/model which can precisely predict whether a user will download an app after clicking a mobile app ad based on the recorded properties of that user. The performance will be evaluated on area under the Receiver operating characteristic (ROC) curve between the predicted probability and the observed target. The smaller differences between our predictions and truths, the better our solution model will be. Then such model can be used to distinguish between meaningful clicks and fraud clicks and reduced the amount of wasted money caused by fraudulence.

Benchmark Model

This project is actually taken from one of Kaggle competitions. They provide a benchmark model which is developed by a random forest method. The socre of this benchmark model is 0.911. And the score is evaluated on area under the Receiver operating characteristic (ROC) curve between the predicted probability and the observed target.

Evaluation Metrics

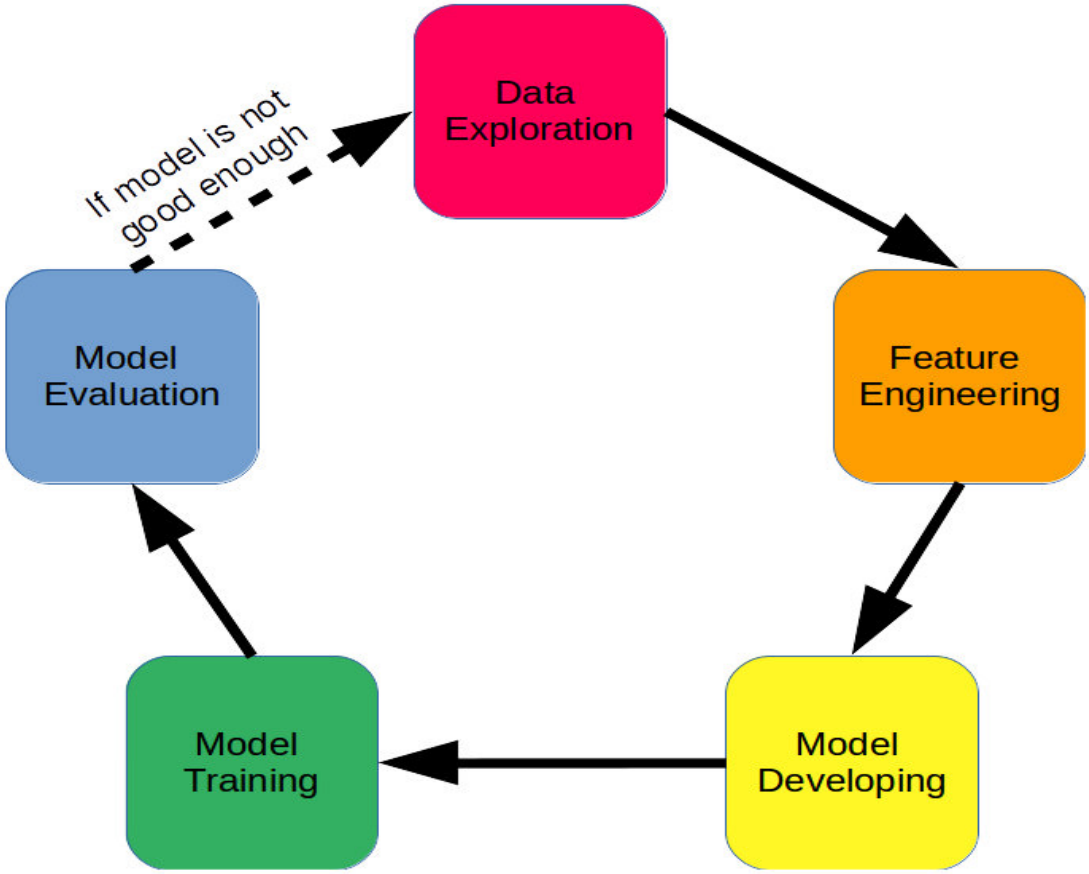
Evaluation metric will be the area under the Receiver operating characteristic (ROC) curve between the predicted probability and the observed target. Such metric is also called as 'AUC'. AUC as a further interpretation of ROC is a very straightforward and easy-understanding metric of a binary classifier system. Since now we are trying to establish a model to predict whether a user will download an app after clicking a mobile app or not. This is exactly a binary classification problem.

Given a threshold parameter T, the instance is classified as "positive" if $X>T$, and "negative" otherwise. X follows a probability density $f_1(x)$ if the instance actually belongs to class "positive", and $f_0(x)$ if otherwise. Therefore, the true positive rate is given by $TPR(T)=\int_T^\infty f_1(x)\,dx$ and the false positive rate is given by $FPR(T)=\int_T^\infty f_0(x)\,dx$. The ROC curve plots parametrically $TPR(T)$ versus $FPR(T)$ with T as the varying parameter. Then the AUC is simply the area under the ROC. Generally, we can judge our model through the value of AUC like follows:

AUC=0.5 (no discrimination)
$0.7 \leqq AUC \leqq 0.8$ (acceptable discrimination)
$0.8 \leqq AUC \leqq 0.9$ (excellent discrimination)
$0.9 \leqq AUC \leqq 1.0$ (outstanding discrimination)

Ref: https://en.wikipedia.org/wiki/Receiver_operating_characteristic

Project Design



by following a very traditional but useful work flow, we first approach this problem by investigating the data. Through this exploratory data analysis, we can establish some basic ideas about the interrelationship between different features or the natural properties of each feature itself. We can even create some new features based on the existing features.

Next, we have to check and clean the data. Maybe sometime our data will contain lots of different values or even missing values. So in feature engineering of data for our model developing, we will handle the problem of missing value and outliers, and/or normalize numeric features. If we have any categorical feature or text format feature, additional data preprocessing techniques will be included.

For developing a proper model for our project, we now will split the whole training data set into to three pieces: one for training, another for validation and the other for testing. This step is for cross-validation. We have already learned something from the projects of ‘boston_housing’ and ‘finding_donors’. Lots of useful algorithms can be our candidate solvers:

- **Deceision Tree**
- **Ensemble Methods** (Bagging, AdaBoost, Random Forest, Gradient Boosting)
- **Neural Network (Multiple Layer Perception)** Stochastic Gradient Descent

Basically, I will try to focus on building two kinds of classifiers: one is based on the neural network; the other is based on the random forest. For the neural network, I will construct 2-3 fully connected layers and try different activation functions for hidden layers. Then the output layer will be passed through a sigmoid function for converting the output values as probabilities. Other parameters like optimizer and loss function(s) are to be decided. For the random forest, it's an ensemble learning method which operate by constructing a multitude of decision trees as collecting the contributions from many weak learners. It keeps the advantages of decision tree and makes the final mode more general. That is, with proper parameter setting of each decision trees, random forest can effectively avoid overfitting the training data.

Grid search method will be optional for finding the best combination of model's parameters. Once we have finished training the models. They will be evaluated by using the evaluation metric, AUC. Models will be check thoroughly to see if there is anything insufficient. Kaggle's official evaluation will be also taken into account. Depending on the performance of these models, we maybe have to go to some previous step to see if there is anything missing or wrong. Once we have an acceptable model (the one with better performance on evaluation and testing), whose score is at least over 0.92, we can stop and publish that model.

Ref: https://en.wikipedia.org/wiki/Artificial_neural_network
Ref: https://en.wikipedia.org/wiki/Random_forest