**Machine Learning Engineer Nanodegree - Capstone Proposal**

Ju-Ying Chen

[chenjyi@tsmc.com](mailto:chenjyi@tsmc.com)

**Proposal**

I choose the topic “TalkingData AdTracking Fraud Detection Challenge” from Kaggle competition as my final capstone project.

For companies that advertise online, click fraud can happen at an overwhelming volume, resulting in misleading click data and wasted money. An algorithm will be build through the technique of machine learning to predict whether a user will download an app after clicking a mobile app ad.

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, and 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 fradulent traffic. TalkingData, China’s largest independent big data service platform. They handle 3 billion clicks per day, of which 90% are potentially fraudulent.

We are challenged to build an algorithm that predicts whether a user will download an app after clicking a mobile app ad. There is a good research, “Detecting Click Fraud in Online Advertising: A Data Mining Approach”, it summarized lots of observations and analyzed 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, and noisy patterns with missing and unknown values.

**Problem Statement**

The problem is simple. How to recognize 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?

TalkingData does have some methods to prevent click fraud, but there are still rooms for improvement. We have to be one step ahead of those fraudsters.

**Data sets and Inputs**

Kaggle competition provides a generous data set 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

• test.csv - the test set

• sampleSubmission.csv - a sample submission file in the correct format

• UPDATE: test\_supplement.csv - This is a larger test set that was unintentionally released at the start of the competition. It is not necessary to use this data, but it is permitted to do so. The official test data is a subset of this data.

There are around hundreds million recorded data, and each data contains a click record, with 8 features. The 8 features are as following,

• 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

Size of the data is 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 split the training data to three pieces, one for training, one for validation and the other for testing. Our final model will be examined by the public Leaderboard of Kaggle.

There is another critical issue that 99.75% of the dataset are labeled as fraud click and only 0.25% are not fraud click. The mobile advertising data are highly imbalanced class distribution. We need to include some data re-sampling strategies for handling this imbalanced label distribution. In the begining, I will establish a model with the original data . If the model does not pass the cross-validation, I will 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 f1(x) if the instance actually belongs to class “positive”, and f0(x) if otherwise. Therefore, the true positive rate is given by TPR(T)=∫T

∞ f 1(x)dx and the false positive rate is given by FPR(T)=∫T

∞ 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 AUC 0.8 (a ≦ ≦ cceptable discrimination)

0.8≦AUC≦0.9 (excellent discrimination)

0.9≦AUC≦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