EECS 349 Project Proposal – Group 21

Qiping Zhang (qze7487) Han Wang (hwm4792) Ji Lin (jle2795)

Fraudulent Click Detection

1. Motivation

The task that we are going to address in this project is Ad-Tracking and Fraud Detection. The ultimate goal we want to achieve is to build a model to predict whether a user will download an app after clicking a mobile app advertisement, based on a vast collection of users' clicking record on app advertisement as our original training data.

According to an investigation from TalkingData, China's largest independent big data service platform, the existence of potentially fraudulent clicks for app advertisements has led to an inevitable problem. A number of users simply click on the ad link of apps without downloading them, which results in wasted ad investment from app service providers. By building this model to identify such users and their corresponding devices to put them into blacklists, such problem could be solved or its latency could be minimized to utmost extent. This would definitely help those companies advertising online to save investment and make more profit. Therefore, such task is valuable to explore and address.

2. Data Acquisition:

We would acquire data from an ongoing featured competition on Kaggle, which is provided by TalkingData. The given dataset covers approximately 200 million clicks over 4 days, including several informative attributes such as app-id, users' device, os etc. Among such attributes, 'is_attributed' is chosen as output label in training process and as predicted result in testing process. Based on the given data, we would build and evaluate the performance of our model by comparing our output class with the given standard classification.

3. Feature Selection:

The features that we will use for our project are listed as follows:

1) Ip: Ip address of click.
2) App: App id for marketing.
3) Device: Device type id of user mobile phone.
4) Os: Os version id of user mobile phone.
5) Channel id: Channel id of mobile ad publisher.
6) Click_time: Time of user clicking on the ad.
7) Attributed_time: If a user downloaded the app after clicking the ad, this feature gives the downloading time; Otherwise, this field is left empty.
8) Is_attributed: A binary target to be predicted, indicating whether the app was downloaded.

4. Initial Approach:

The data processing in our project will be as follows: Since many of the features from the given dataset are not numerical values, it is currently hard to measure their distance and split them according to their difference and correspondence. So, our goal is to encode such feature values and convert them to measurable numerical values, which could properly reflect their variance and influence on the classification. For example, originally the device type given will be like "iphone 6 plus", but we could manually collect all types appeared in the dataset and assign them

an id respectively. Meanwhile, if two different device types are in the same family (e.g. iphone 6 plus and iphone 7), we assign them a closer id; Otherwise we give them id value far from each other. Besides, we could also convert “time” format to the number of seconds from a certain moment, so that the measure of distance between different times could be more convenient. In a word, we would change all features into numerical values, so that we could achieve an informative comparison on them to facilitate our classification.

As for the ML techniques we will try first, Nearest Neighbor would be a prior choice. This is because each attribute of our data set is categorical, and after the pre-processing of our data, the distance measure between two values of any attribute should be intuitive and reflective. Also, since in our problem each feature has a great number of various possible values (e.g. IP address), it is not quite realistic to apply decision tree to split on each feature. In this case, Nearest Neighbor seems to be a suitable choice for the first try.

To evaluate our success in first try (i.e. nearest neighbor), we will firstly use ZeroR as a baseline, to check whether NN could perform better than using most frequent case as prediction. Then we would split the training data set to apply 10-fold (this is flexible) cross validation to help our testing. We could check the average accuracy/precision/recall on 10% test set to see whether the classification works well and make changes to the model according to this feedback.