**TCDS Final Project – Tal Elisberg**

Finding Future Customers for installing Mobile Apps

In a Real Time Ad Bidding Stream of in-app commercials.

Project Introduction:

In the Ad-Tech industry, publishers of applications, are given the option to bid in a real-time-bidding (RTB) Internet protocoled bid stream.

Each advertiser’s main goal is to find users whose probability to consume the advertised content through ads is the highest.

The advertiser pays for each 1000 impression a price which is decided by a second-best bid price auction which happens every 100ms. The number of estimated bids in such a time frame is 200,000 bids (!).

Advertisers get rewarded when the user installs the published app, assuming no fraud was detected during the install (for example the usage of a bot for installing apps).

Project Goal:

Our goal, as data scientists in the project is to identify and predict the chance of install according to the main features to drive a user for install, given a certain bid-stream.

It should be assumed that most bid-streams has a certain bias towards the end target since each advertiser, chooses to have certain filter when bidding inside the RTB protocol in order not to waste efforts on ads that has a very low chance of conversion.

Previous knowledge:

* The known probability for a user to install out of 1000 impressions of the same ad is estimated in 2% installation rate. Meaning 1 installer can be found in a group of 500 people(!).
* The bid stream doesn’t contain users who already have installed the app. These can be found in a different table.
* Not all users are shown in a bid-stream, since the final bid made for an impression is sent for an auction, and some bids may not be high enough.
* Our data will revolve around impression and installation events in the bid stream.
* Since reasons for install can be various, we’ll assess the probability for install (y\_proba) by using best performing features, and review the effects on the actual install rate in live bidstream.
* - We already know, that users which ‘hang around’ in some app categories (action, casino, casual etc.) tend to have more than 1 app installed in this category.

- We also know that the chance of converting impression to an install reduced dramatically as the amount of times the same ad presented rises.

Finally, we know that, some users have developed a so-called ad-blindness and their probability to install is the lowest.

Project Design:

Sources of data:

1. Data Table name: Monopoly\_01-02\_2019

Description:

Bid-stream queried data from a campaign aimed for users who may install a game called ‘Monopoly Slots’

<https://play.google.com/store/apps/details?id=com.scientificgames.monopolyslots&hl=en>

This stream is characterized by data of users who have at least one “SLOTS” application installed in their mobile. The data is aimed at Android users only from the US. Time frame of the data is between 01-02/2019.

Source of data: Amazon Athena (private company account)

1. Enrichments:
   1. Data Table name: USER\_info

Description:

DMP of stored data on users. The query gives all users who have ‘Monopoly Slots’ installed on their mobile.

SELECT \* FROM USER\_INFO WHERE USER IN

(

SELECT DISTINCT USER\_ID FROM USER\_APP WHERE

USER\_APP = ‘com.scientificgames.monopolyslots’

)

Source of data: Snowflake Computation (Data Warehouse) , private company account.

* 1. Bundle(app) and category.

Dictionary-like Data frame to indicate which category / sub category belongs each app.

* 1. Data Table name: US\_TZ

Description:

Dictionary-like Data frame to indicate hour difference from UTC according to the US State, for analyzing user local hour.

Columns: state abbreviation (‘abbr’), UTC time difference

Source: Google

1. External Validation: First Week of 03-2019 from RTB
2. Subjects of installation group should be treated with inclusion criteria since the install group as very imbalanced (1:500).
3. Install event is the outcome of the model.
4. Data exploration strategy:
   1. Removing irrelevant columns from original RTB protocol that have:
      1. Useless information
      2. High percentage of (Nan Values / Total population).
   2. User based information:
      1. State
      2. YOB (aggregated by generation according to )
      3. Gender ( if possible)
      4. Type of device (maker, class (premium, standard, low cost)
      5. No . of apps in the category.
      6. no. of entries per week
      7. No. of apps the user has on its phone
   3. stream related information:
      1. local hour of the day in groups (4,6,8) checking which one is better.
      2. Day of the week
      3. Day of the month
      4. IS\_WEEKEND
      5. Publisher from which the add arrived (ssp id)
      6. No. of unique SSPs the user has been arriving through the bid stream (pre aggregated in the database by the company)
      7. The bundle frequency cap – no. of total times the ad was presented to a user. (pre aggregated in the database by the company)
5. Data enrichments:
   1. Due to a large amount of categorical data, some of the categorical features will be bagged according to density grouping methods.
   2. Other categorical data can be enriched by significance test according to p-value of a sub feature while exploring a feature.
   3. Final data before prediction will be added a k means cluster to identify a hidden sub group in order to use it as enrichment for the prediction.
6. Outliers and Nans.

Dealing with outliers:

Dropping outlying records or bagging them aside.

Dealing with Nans:

If data is missing more than 30% (Nan). Column should be dropped.

Otherwise, treated aside.

Models

Data Division:

\* First: Data is sorted by bidding time ascendingly.

01.01.2019 – 14.2.2019 – Training Data

15.02.2019 - 28.02.2019 – Dev Data

01.03.2019 – 07.03.2019 – Test Data.

* Data Balancing should be set as a hyper parameter to suggest data imbalance, due to pour performance of over/ under sampling.
* No stratification / subsampling methods should be taking place.

The models’ outcome should be a classification – of install vs non install.

Measures for training and evaluation of data:

1. Tree based models (Random Forest, Light GBM)
2. Logistic regression.

Model Deployment

* Final user of prediction:

Advertising companies who are interested in putting advertises in apps for

Getting more installs of published contents.

* **Prediction final metric will be evaluated as the price to pay for a single impression**

**As a result of the probability to install, having that the price for an install is 5$.**

**Paremeters:**

* **Total Cost of impressions.**
* **Return On Ad Spend – ROAS = Revenues / Total Spend**
* **Install Rate of each group.**
* **Total Revenues.**
* **Out of Project scope:**
  1. **Implementation:**
     + The prediction will be implemented in production via a Py4J implementation

when python code is meant for assessing real-time bids.

* 1. **Model Re-Training:**
     + Technically the model can be updated each day. Using PySpark to read last 30 days data from the bid stream.
     + Using Amazon’s EMR for setting up a virtual machine, training the model over last 30 days.
     + Deploying The Vectors in production.
     + Part of monitoring the model will be:
       - Sending alert for null vectors output each day.
       - Sending alerts if changes in model accuracy goes above/below 20% percent.
       - Alert for each time model is deployed in production.