Data Scientist Udacity Nanodegree

Capstone project report: Starbucks Challenge on Product Promotion

Contents

1. Project overview

This project deals with simulated data from Starbucks experimented with response from customers with promotion on products. Each offer and its features stored in portfolio.json. In total, the data contains 10 offers, four of which are discounts, four are BOGO (Buy One Get One free) and two are informational. The first two types of offer are similar but with one distinction: discount gives back credits to a customer once a certain dollar amount purchased met while BOGO gives a customer two drinks with price of one while that offer is valid. The informational offer is a notice or email to inform customers about a product or service without any credit or discount in place.

The goal of this project is to explore and investigate the influence of each offer on customers, not only as the whole population but as a certain group of customers. The influence is both ways, either customers really like one promotion or dislike a particular promotion. Understand customers' preference would serve customers the right or prefer offer while saving Starbucks marketing expenses to send and distribute promotion materials.

The dataset for this projects contains a file named profile.json which listed basic customers's demographics such as age, genders, income, membership date. The larger set of data is from transcript.json which logged transaction and records of users during experimenting periods.

2. Problem Statement

The overarching goal of this project is to identify a customer's preference to a promotion so that the promotion is more efficiency the deliver the right type offer to each customer. To achieve this goal, I will follow basic steps to gradually investigate the data from data exploration, cleaning and wrangling data, experiment with data modelling and building models for prediction. The final product of this project is a machine learning models or a simpler heuristic algorithm to recommend a list of ranked offers with the key inputs from users, users past transaction, and promotion data.

3. Metrics

In this dataset, I defined two metrics to rank if an offer and following transaction is successful or not

1. $\text{completion rate} = 1 - \frac{\text{number of completed}}{\text{number of viewed}} \times \frac{\text{number of completed}}{\text{number of received}}$

2. Average amount of dollar spent per offer

Successful rate of completion (#1) reflects the ratio of offer completed per received and a offered completed per view. For example:

- A successful offer presented with a completion rate of zero, in which number of an offer received was viewed then completed.
- A completion rate < 0 when number of viewed is smaller than the number of completions. This is important to marketing team since each offer costs money but does not viewed and thus does not have any influence on customers' behavior.
- A completion rate between 0 and 1 represented an offer that viewed but not completed. It could be an offer is difficulty to be completed, the customer is not frequent users of Starbucks products

Amount spent per each offer would indicate the absolute quantity of dollar spent. It is a general term can be used to compare if we know more underlying conditions such as if the user viewed an offer versus user received an offer but not viewed that offer.

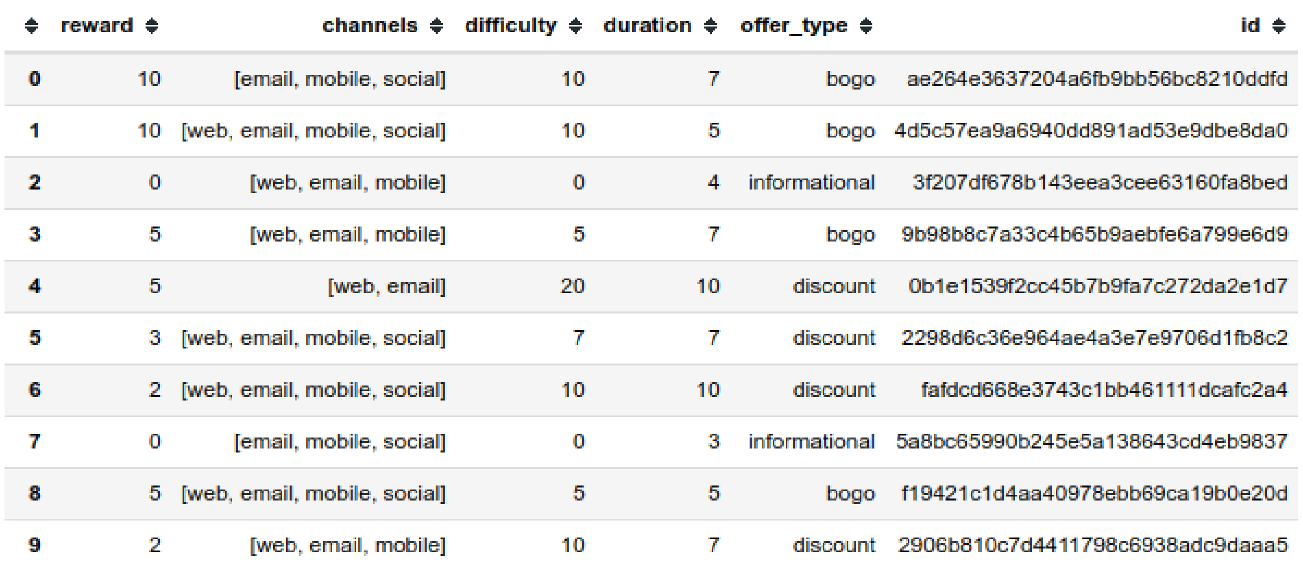
The metrics for learning machine is not applicable for my approach since there is no machine learning model built out from this project. Attempting to apply FunkSVD for collaborative filtering by decomposing a person-offer matrix failed since the predicting matrix is not reliable.

4. Data Exploration and Visualization

4.1. Portfolio:

Portfolio is a simple table with 10 rows and 6 columns featuring three types of promotion:

- **BOGO (Buy One Get One free):** if a customer received this offer, the customer can get two similar drinks and pay for one, essentially 50% except customer now have two drinks. BOGO works better for customers who has a friend or a colleague to join that drink
- **Discount**: after certain dollars purchased, for example $10 during valid period, a credit of $5 is added to customers' Starbucks account. This works well for individuals who want a discount but not to drink one product too much.
- **Informational**: a notice or email informing a new product or a new type of service. A customer is considered under *influence* during valid period if that customer viewed the offer. There is no record marks for completion of informational offers like BOGO or discount offers.



	reward	channels	difficulty	duration	offer_type	id
0	10	[email, mobile, social]	10	7	bogo	ae264e3637204a6fb9bb56bc8210ddfd
1	10	[web, email, mobile, social]	10	5	bogo	4d5c57ea9a6940dd891ad53e9dbe8da0
2	0	[web, email, mobile]	0	4	informational	3f207df678b143eea3cee63160fa8bed
3	5	[web, email, mobile]	5	7	bogo	9b98b8c7a33c4b65b9aebfe6a799e6d9
4	5	[web, email]	20	10	discount	0b1e1539f2cc45b7b9fa7c272da2e1d7
5	3	[web, email, mobile, social]	7	7	discount	2298d6c36e964ae4a3e7e9706d1fb8c2
6	2	[web, email, mobile, social]	10	10	discount	fafdcd668e3743c1bb461111dcafc2a4
7	0	[email, mobile, social]	0	3	informational	5a8bc65990b245e5a138643cd4eb9837
8	5	[web, email, mobile, social]	5	5	bogo	f19421c1d4aa40978ebb69ca19b0e20d
9	2	[web, email, mobile]	10	7	discount	2906b810c7d4411798c6938adc9daaa5

Figure 1: Portfolio with all features of each promotion

4.2. Profile:

Profile contains data of a table of 17000 rows by 6 columns listed user demographic data such age, income, year of membership, user_id and gender. Distribution and counting of each category are shown in Figure 2.

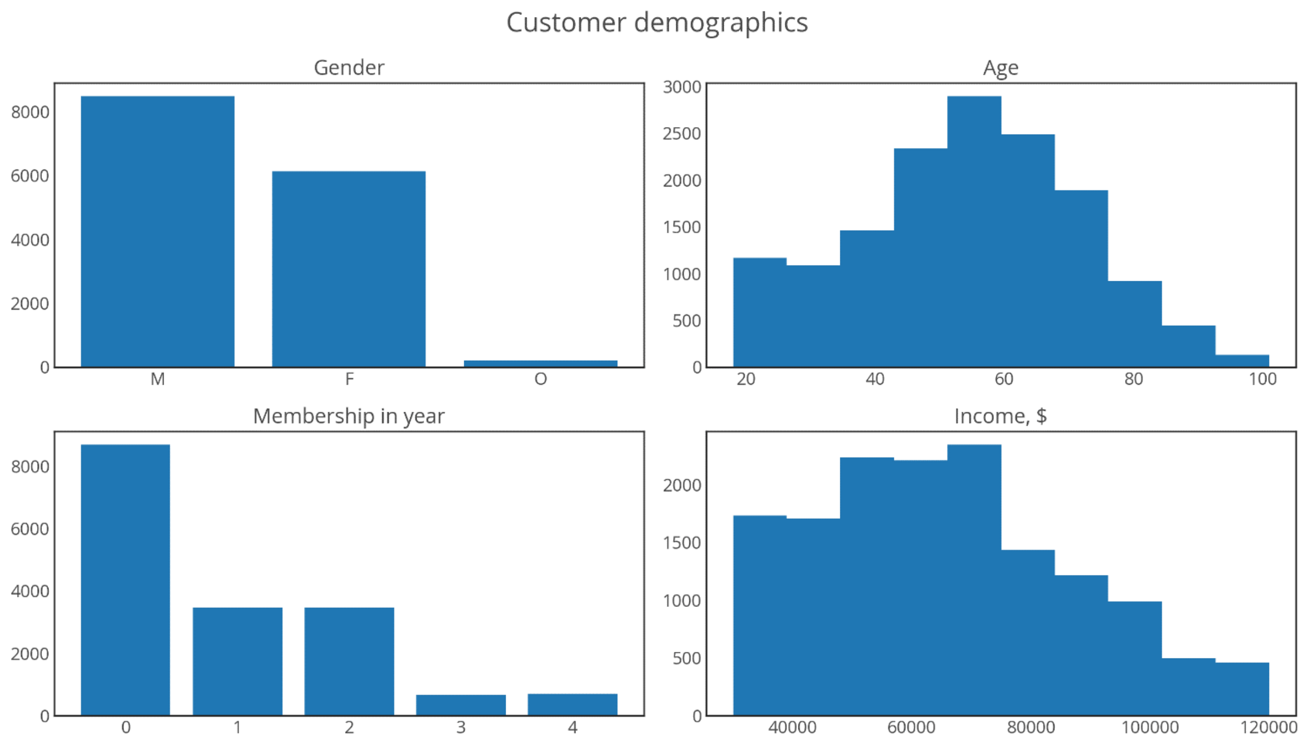


Figure 2: Customers' profile on the simulated Starbucks dataset

Age distribution of male and female customers showing a clear distinction between dominant male young customer. The ratio of male to female customer reduces over age and at senior ages, female customers are becoming more than male customers.

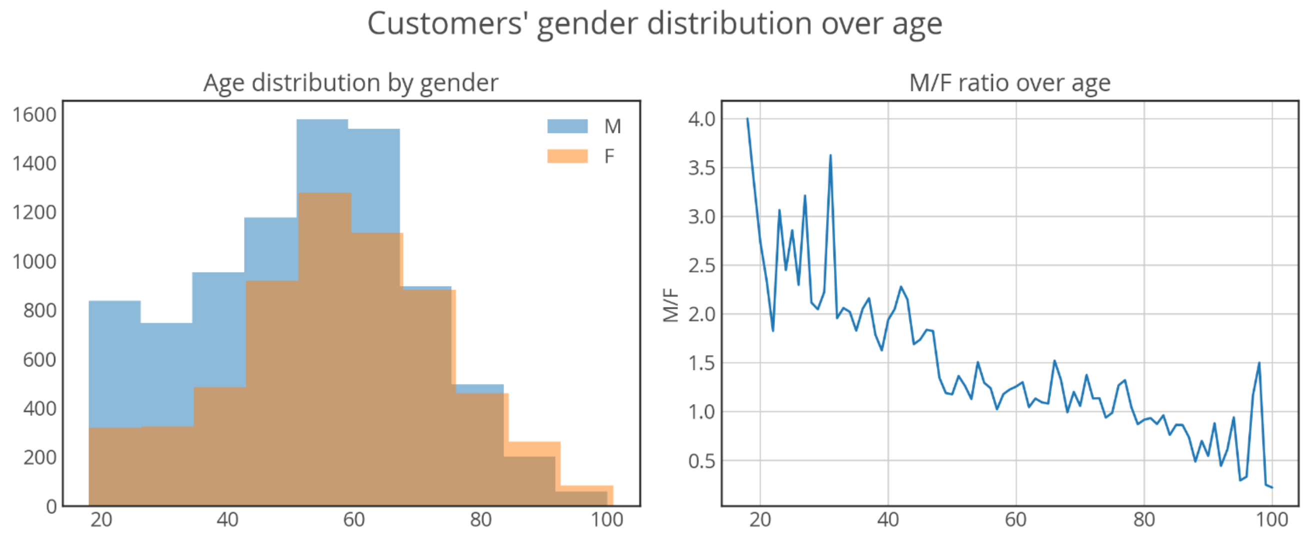


Figure 3: Distribution of customers' gender (left) and the ratio (right) over age

4.3. Transcript

Transcript file is a table of 306 534 rows by 5 columns listed all transaction during simulating periods. This is the main file in this dataset with a long-table format with records when promotion is received, viewed or completed with amounts of dollar in each transaction. Figure 4 below shows a visualization number of events in top-left graph with the bottom graph detailing for each offer. The top right graph shows a right-skewed distribution of amount per transaction. The average spending is $12.77 and the median value is $8.89.

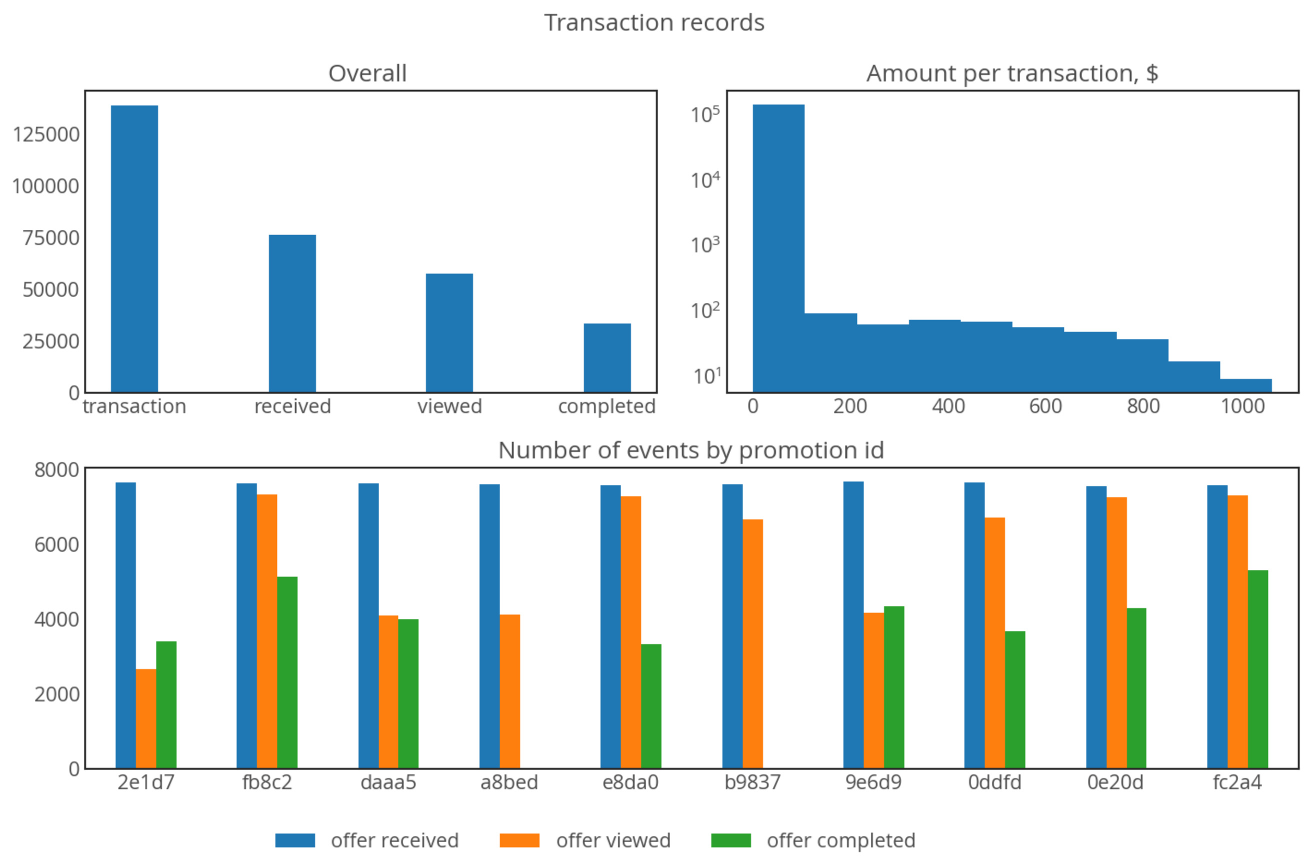


Figure 4: Transaction records of all offers, dollar spent, and categorizing to each offer

For all promotion received, about 75% were viewed and 45% of received were completed. The portion of offers that completed from those were viewed are 58%. We will continue to look closer on those metrics on Figure 5.

The bottom graph is important for the later step building a recommendation system with new users or users with no information. The overall rate of completion by each offer can use as the starting point to recommend for new users to Starbucks.

As we moving along with data analysis, it is important to recognize that not all offers were marketed equally. In total 10 offers, four of them were marketed via four channels (web, email, mobile, social), three offers were marketed by a mixed of three channels, and one were minimally marketed via web and email. The number of viewed is relevant to quantify the reach of each channel. The received counts were obtained from the source of promotion and the completed counts were automatic after certain dollar purchase are met for eight offers (discount or BOGO).

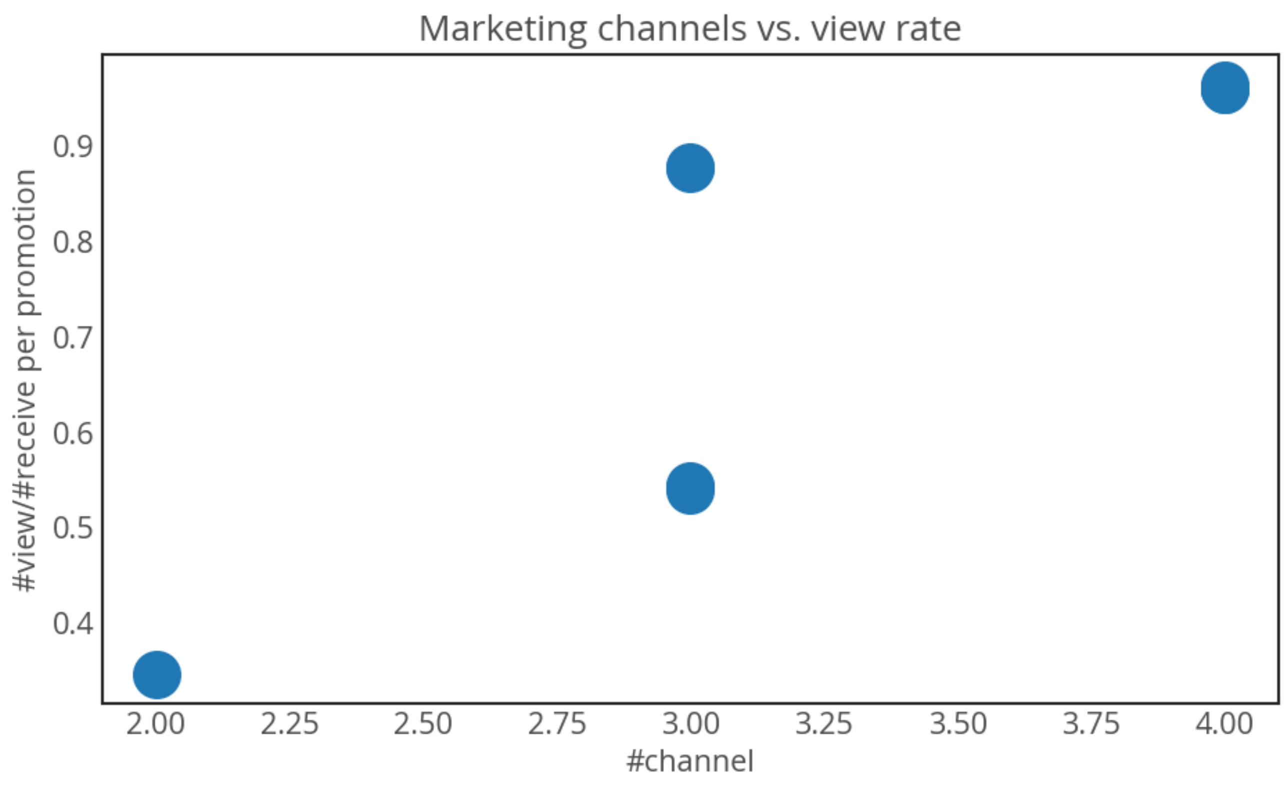


Figure 5: A closer look to number of views per number of channels informing an offer

For a mixed of three marketing channels, without social, the view rate is about 0.54 or 54% of customers received offers viewed the offers. If offers were marketed via social and emails, the view rate drops to 0.35. If only web channel was not deployed, the view rate is 0.87, which made the web channel the least influential one. Email was used in all ten channels which also means that we cannot draw any relation on view rate by email.

With a new arrangement of each promotion by each person, we can track the average dollar spent per offer by all customer received that offer (Figure 6). For discount or BOGO offers, the total amount is clearly higher when the offer was completed with a small increase when the offer viewed. Note that the amount of dollar ***was not really increased*** but the total offers that viewed or completed reduced and thus making the denominator becomes smaller with offers were received, viewed, and completed. A jump of the average amount with completed offers suggests a substantial amount of offer received was completed. A slightly increase of amount spent per offers received then viewed indicated a minimal effort to view an offer in comparison to complete that offer.

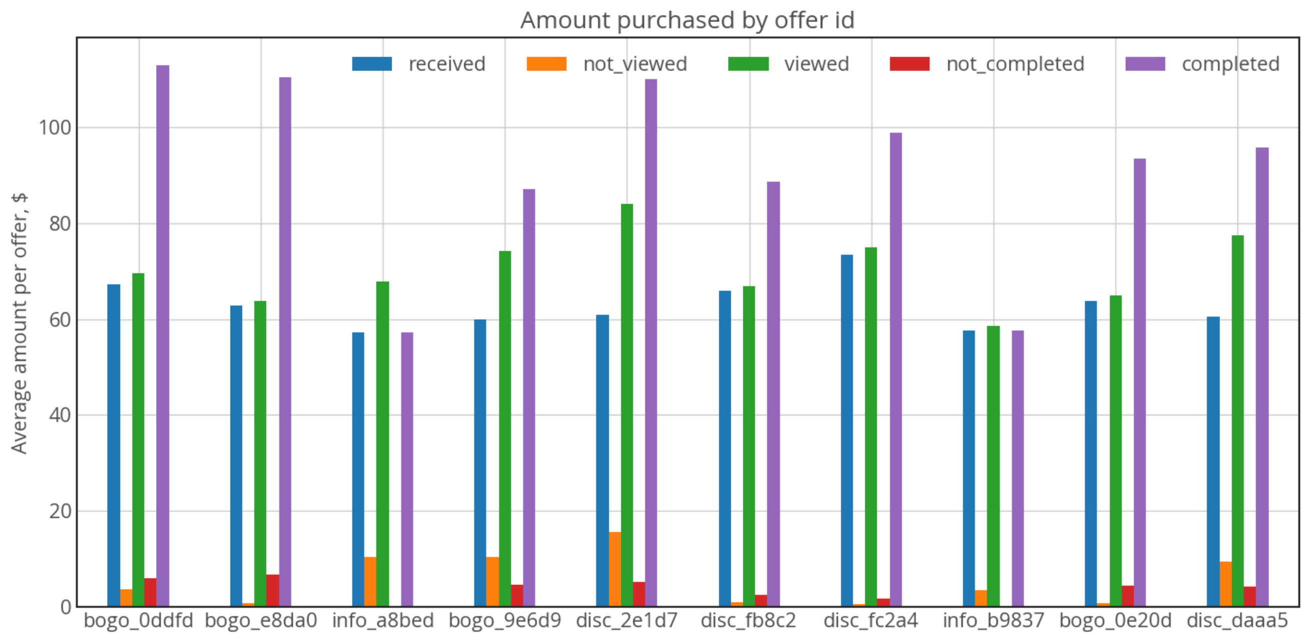


Figure 6: Average dollar spent for each offer

Figure 6 also clearly shows with offers that not viewed or not completed by customer during the offer is valid get ten times less dollar spent. This is inline with customers not interested with Starbucks offer or not frequent customers. Generally, this is not potential receiver for promotion.

Using rate of completion as defined in metrics #1, we can calculate the overall rate for all offers. Noted that with informational, there is no official record of “completed” in transcript file. Figure 7 only shows discount or BOGO offers with records of completed events. As defined above, the completion is the remainder of 1 to the product of number of completed squared and divides to the product of number of viewed and received. One caveat of the completion is it is highly subjective. In fact, it is my approach to define a metrics to rate if one offer is successful or not. Nevertheless, the first three offers that completed the most are BOGO offers. The rate of Figure 7 will eventually be used to make recommendation for new users based on the popular offers, in this case, the offers which were completed ordered from most to less.

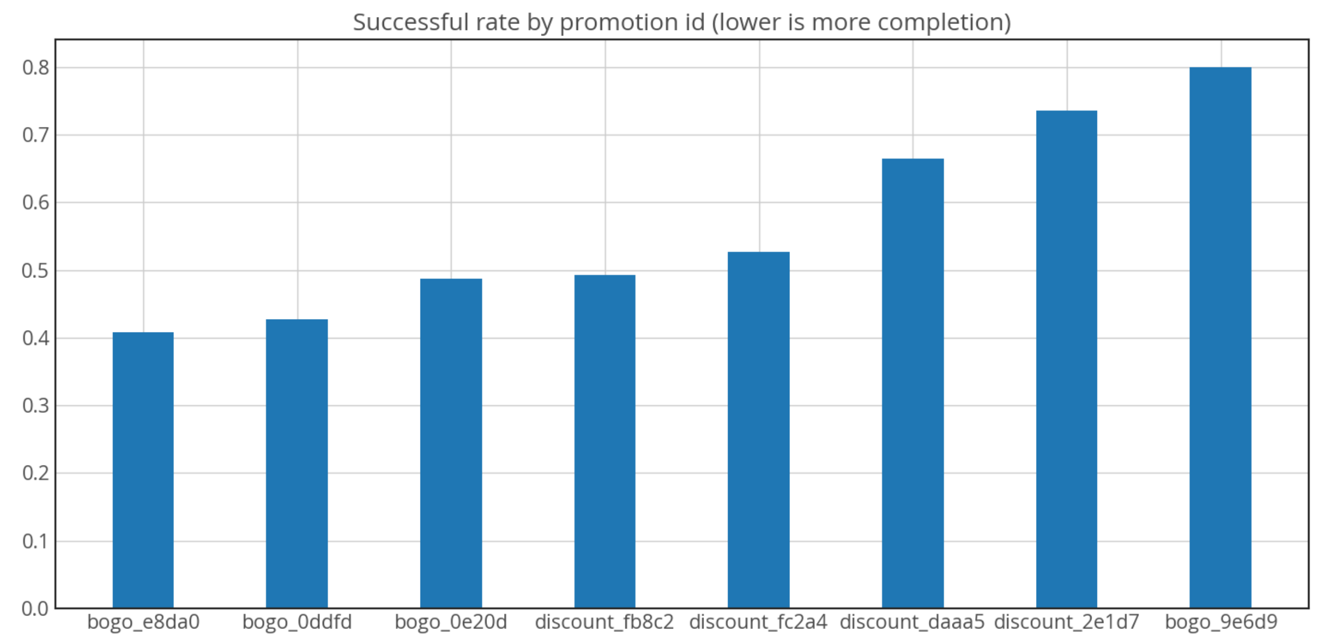


Figure 7: Rate of completion by each offer. A lower number shows higher rate of completion of offers

5. Data Processing:

The customers **profile** is pretty clean and require minimal effort to wrangle the data. After json file is loaded into a DataFrame using pandas library, we can check general information by:

```
profile = pd.read_json('data/profile.json', orient='records', lines=True)
profile.info(verbose=True)

class 'pandas.core.frame.DataFrame'>
RangeIndex: 17000 entries, 0 to 16999
Data columns (total 6 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   gender            14825 non-null  object
 1   age               14825 non-null  float64
 2   id                17000 non-null  object
 3   became_member_on  17000 non-null  datetime64[ns]
 4   income            14825 non-null  float64
 5   membership_age    17000 non-null  int64
dtypes: datetime64[ns](1), float64(2), int64(1), object(2)
memory usage: 797.0+ KB
```

For age column, above 2000 rows contains value of 118 which is not realistic for human age, so with this cell, the value is set to numpy.NaN. The most important output from the profile data is to create an encoded matrix with row for user and columns for features such as gender, age, range of income. The value in encoded profile is 0s or 1s. With encoded matrix ready, we can calculate np.dot product of two users and similarity of the two. A higher output is showed a higher similarity of two users. A function named find_similar_users() used the data to return top similar users from encoded profile matrix.

The transcript file is the main bulk of data. Similar to profile data, the data is loaded into Jupyter Notebook using `pandas` library.

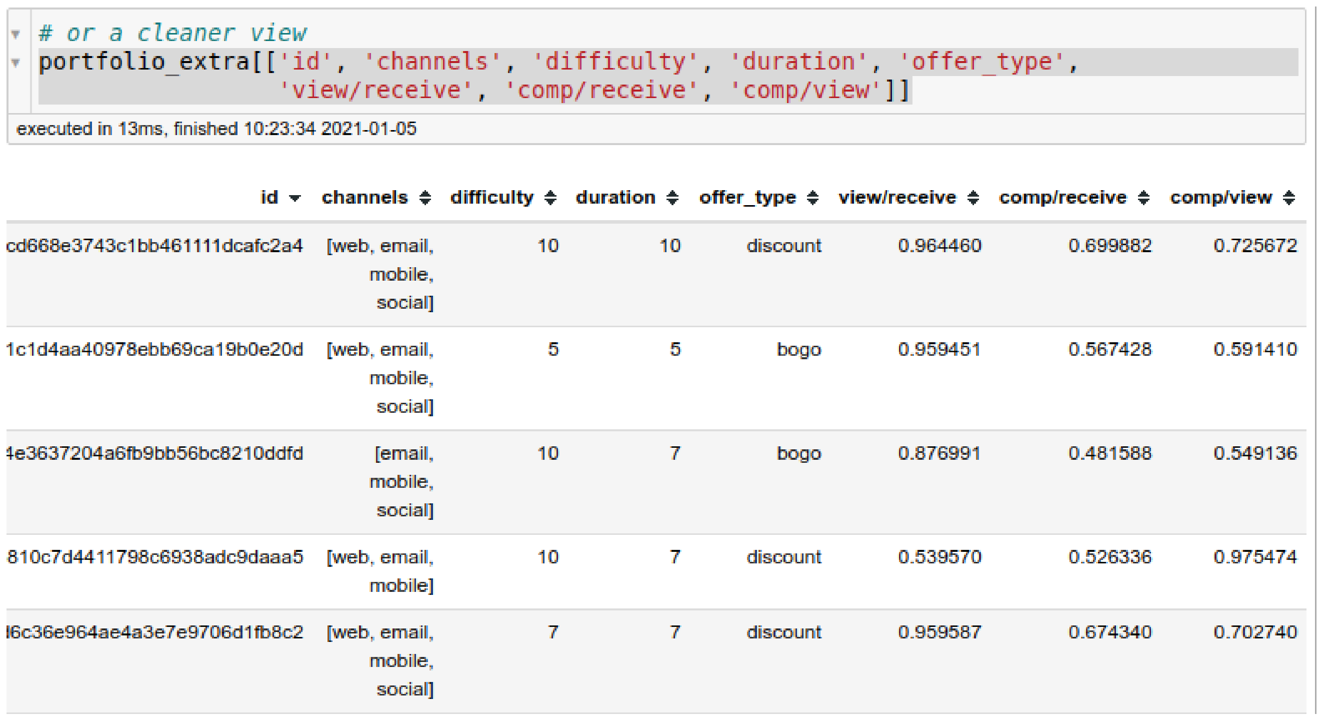
```
transcript = pd.read_json('data/transcript.json', orient='records', lines=True)
transcript.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 306534 entries, 0 to 306533
Data columns (total 4 columns):
 #   Column  Non-Null Count   Dtype 
---  ------  --------------   ----- 
 0   person  306534 non-null  object
 1   event   306534 non-null  object
 2   value   306534 non-null  object
 3   time    306534 non-null  int64 
dtypes: int64(1), object(3)
memory usage: 9.4+ MB
```

The **transcript** data is a long table with recorded transactions of each person and event occurred with each offer and with time marked as the hour from the beginning of simulation. The .info() output shows that transcript is filled. From this dataframe, we can make summary such as on Figure 4. The most simple and relevant steps is counting variable in categorical data by value_counts(), distribution of amount spending or with combination with groupby() attribute. Figure 7 shows a visualization completion rate by each offer by grouping on each offer and count events and then used to calculate the completion rate of each offer.

With a new arrangement of transcript data by counting transaction and event by each person on each offer, we can create a similar matrix like user-item matrix with movie recommendation. The key different is the value in the cell is not the rating, but a dictionary containing a key statistical of transactions by a person and an offer. This arrangement will enable us to analyze response by each customers or group of customers by `number received, viewed, completed, other_viewed_promo` and total amount spent. In the accompanying Jupyter Notebook, the new grouping data of transcript is a dataframe named `df_info`.

We can also take advantage of new grouping in `df_info` by merging data portfolio data on each offer id. The merged dataframe contains features of each offer and performance metrics. Here is an except of newly merged dataframe called `portfolio_extra`. With more granular data from `df_info`, we can calculate average dollar spent on each promotion when an offer was `viewed`, `completed` or `not_viewed` or `not_completed` such as in *Figure 6*.



```
# or a cleaner view
portfolio_extra[['id', 'channels', 'difficulty', 'duration', 'offer_type',
                 'view/receive', 'comp/receive', 'comp/view']]
```

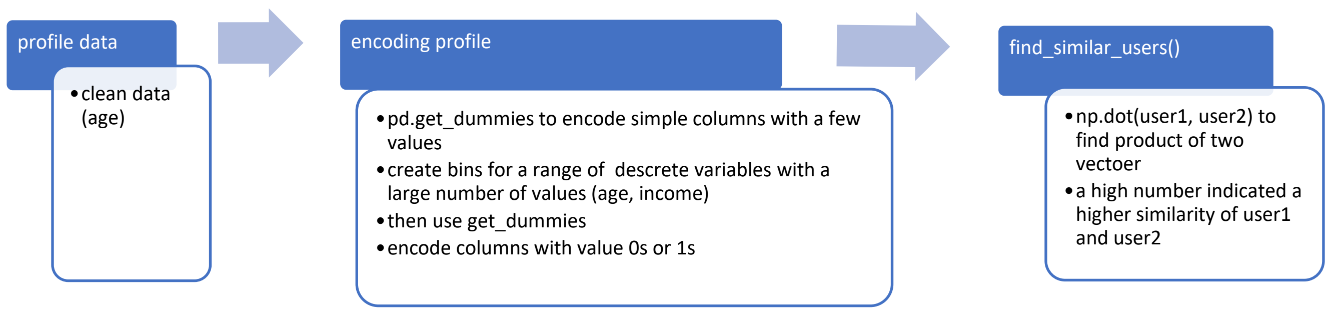
executed in 13ms, finished 10:23:34 2021-01-05

id	channels	difficulty	duration	offer_type	view/receive	comp/receive	comp/view
cd668e3743c1bb461111dcafc2a4	[web, email, mobile, social]	10	10	discount	0.964460	0.699882	0.725672
1c1d4aa40978ebb69ca19b0e20d	[web, email, mobile, social]	5	5	bogo	0.959451	0.567428	0.591410
4e3637204a6fb9bb56bc8210ddfd	[email, mobile, social]	10	7	bogo	0.876991	0.481588	0.549136
810c7d4411798c6938adc9daaa5	[web, email, mobile]	10	7	discount	0.539570	0.526336	0.975474
6c36e964ae4a3e7e9706d1fb8c2	[web, email, mobile, social]	7	7	discount	0.959587	0.674340	0.702740

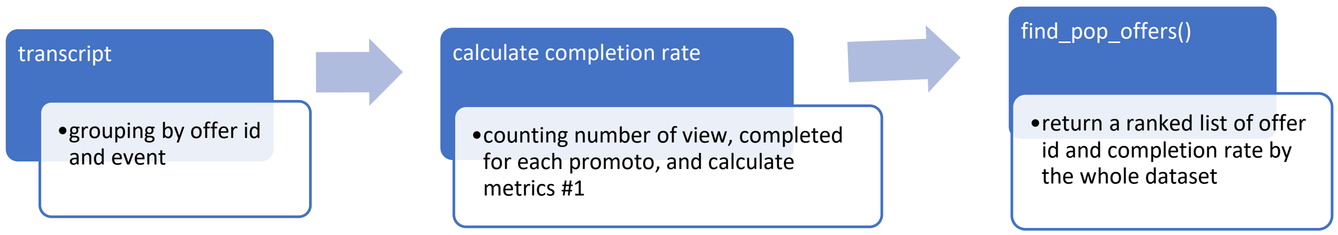
6. Implementation:

Below are diagrams depicting steps taken to build a recommendation system based simulated data from Starbucks.

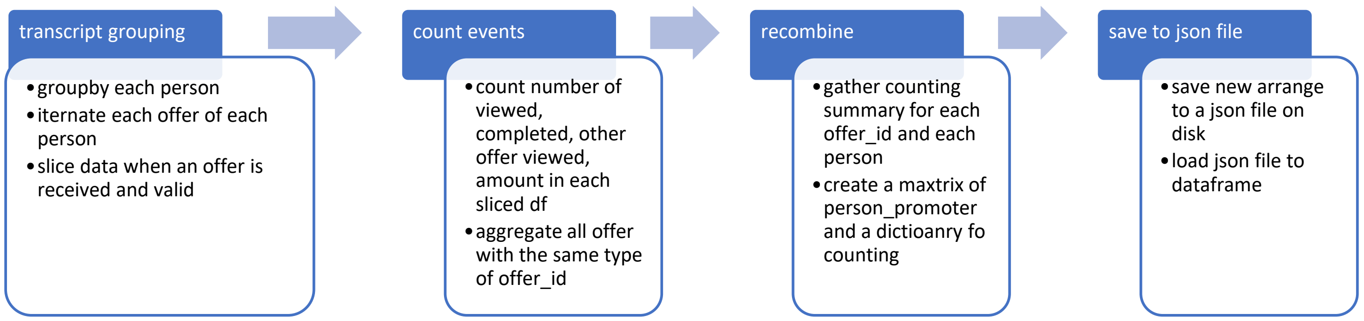
Profile data:


profile data
• clean data (age)
encoding profile
• pd.get_dummies to encode simple columns with a few values
• create bins for a range of descrete variables with a large number of values (age, income)
• then use get_dummies
• encode columns with value 0s or 1s
find_similar_users()
• np.dot(user1, user2) to find product of two vectoer
• a high number indicated a higher similarity of user1 and user2

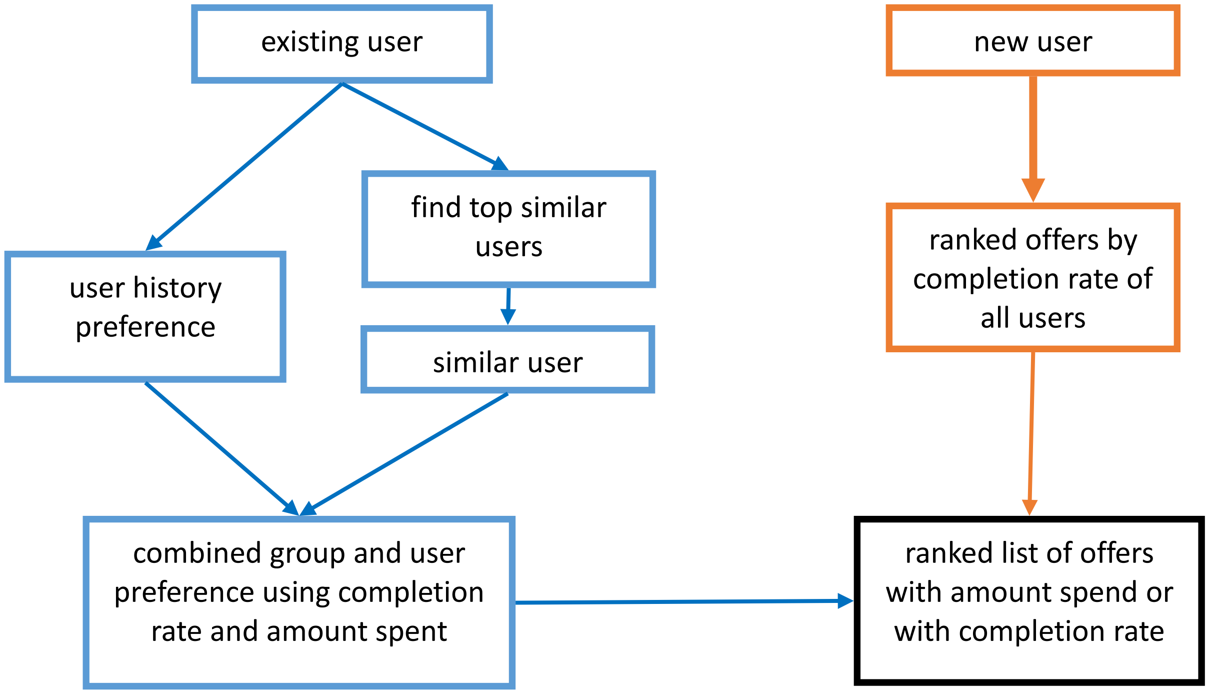
Transcript data:


transcript
•grouping by offer id and event
calculate completion rate
•counting number of view, completed for each promoto, and calculate metrics #1
find_pop_offers()
•return a ranked list of offer id and completion rate by the whole dataset

Rearrange transcript data


transcript grouping
•groupby each person
•iternate each offer of each person
•slice data when an offer is received and valid
count events
•count number of viewed, completed, other offer viewed, amount in each sliced df
•aggregate all offer with the same type of offer_id
recombine
•gather counting summary for each offer_id and each person
•create a maxtrix of person_promoter and a dictioanry fo counting
save to json file
•save new arrange to a json file on disk
•load json file to dataframe

User offer recommendation


existing user
find top similar users
user history preference
similar user
combined group and user preference using completion rate and amount spent
new user
ranked offers by completion rate of all users
ranked list of offers with amount spend or with completion rate

7. Refinement:

Rearrangement of transcript data is critical to get performance of each person on each offer id by the purchased amount. The original transcript data include transaction amount but we don't have a tag of promotion id so we cannot be sure which that amount belongs to which offer. The new arrangement tracks when a person received an offer and count all amount by that person during the promo is valid. Therefore, it is more reliable to track down amount spending rather by grouping and counting and averaging from the transcript file alone.

In addition, by iterating to each person and offer id received, we can add more counting besides number of viewed, completed and amount purchased. I added a "other offer viewed" to track if concurrent offers were occurred. Although that data is not used in this report, it is helpful to quantify effect of multiple offers sent to customers and their responses.

8. Model Evaluation and Validation

Not applicable.

9. Justification

As we have been familiar with movie recommendation system, using `FunkSVD` to decompose user-item matrix is more innate. However, with this dataset, the native `FunkSVD` produced a user-latent features matrix and pro-latent features that when combining by np.dot is not reliable. The value in cell is almost random although the error is small. Using third part Surprise did not improve the prediction of user preference on item, reporting Nan for error. Therefore, I used a heuristic approach with two defined metrics to rate successfulness of each offer in this dataset.

Let see how the recommendation works:

For an existing user:

```
user_id = 'ddce9ae854314d8cb347db0a7b5db9f8'

# group preference
group = evaluate_similar_users(user_id, profile_df=encoded_profile,
                               info_df=df_info, sort_amount=True)
group

{'f19421c1d4aa40978ebb69ca19b0e20d': 56.97,
 '0b1e1539f2cc45b7b9fa7c272da2e1d7': 45.9,
 '5a8bc65990b245e5a138643cd4eb9837': 44.15,
 '2906b810c7d4411798c6938adc9daaa5': 30.41,
 '2298d6c36e964ae4a3e7e9706d1fb8c2': 28.37,
 '4d5c57ea9a6940dd891ad53e9dbe8da0': 25.76,
 'ae264e3637204a6fb9bb56bc8210ddfd': 25.07,
 '9b98b8c7a33c4b65b9aebfe6a799e6d9': 23.66,
 'fafdcd668e3743c1bb461111dcafc2a4': 21.32,
 '3f207df678b143eea3cee63160fa8bed': 19.28}
```

# user preference

```
user = evaluate_user_history(user_id, df=df_info)
user

{'3f207df678b143eea3cee63160fa8bed': 0.0}
```

# based on both the user and similar users preference

```
recommend_offers(user_id=user_id, profile_df=encoded_profile,
                    info_df=df_info, sort_amount=False)

{'0b1e1539f2cc45b7b9fa7c272da2e1d7': 45.9,
 '5a8bc65990b245e5a138643cd4eb9837': 44.15,
 'ae264e3637204a6fb9bb56bc8210ddfd': 25.07,
 '9b98b8c7a33c4b65b9aebfe6a799e6d9': 23.66,
 '2906b810c7d4411798c6938adc9daaa5': 30.41,
 '2298d6c36e964ae4a3e7e9706d1fb8c2': 28.37,
 'f19421c1d4aa40978ebb69ca19b0e20d': 56.97,
 'fafdcd668e3743c1bb461111dcafc2a4': 21.32,
 '3f207df678b143eea3cee63160fa8bed': 19.28,
 '4d5c57ea9a6940dd891ad53e9dbe8da0': 25.76}
```

# change sort_amount from False to True

```
recommend_offers(user_id=user_id, profile_df=encoded_profile,
                    info_df=df_info, sort_amount=True)

{'f19421c1d4aa40978ebb69ca19b0e20d': 56.97,
 '0b1e1539f2cc45b7b9fa7c272da2e1d7': 45.9,
 '5a8bc65990b245e5a138643cd4eb9837': 44.15,
 '2906b810c7d4411798c6938adc9daaa5': 30.41,
 '2298d6c36e964ae4a3e7e9706d1fb8c2': 28.37,
 '4d5c57ea9a6940dd891ad53e9dbe8da0': 25.76,
 'ae264e3637204a6fb9bb56bc8210ddfd': 25.07,
 '9b98b8c7a33c4b65b9aebfe6a799e6d9': 23.66,
 'fafdcd668e3743c1bb461111dcafc2a4': 21.32,
 '3f207df678b143eea3cee63160fa8bed': 19.28}
```

# new user, this shows the rate of completion

```
recommend_offers(user_id='new_user')

{'9b98b8c7a33c4b65b9aebfe6a799e6d9': 0.41,
 '0b1e1539f2cc45b7b9fa7c272da2e1d7': 0.43,
 '2906b810c7d4411798c6938adc9daaa5': 0.49,
 'fafdcd668e3743c1bb461111dcafc2a4': 0.49,
 '2298d6c36e964ae4a3e7e9706d1fb8c2': 0.53,
 'f19421c1d4aa40978ebb69ca19b0e20d': 0.66,
 'ae264e3637204a6fb9bb56bc8210ddfd': 0.74,
 '4d5c57ea9a6940dd891ad53e9dbe8da0': 0.8,
 '3f207df678b143eea3cee63160fa8bed': nan,
 '5a8bc65990b245e5a138643cd4eb9837': nan}
```

We can experiment more with the `recommend_user()` function in the Jupyter Notebook, section 6.

10. Reflection

Through this project with simulated dataset from Starbucks, we successfully built a recommendation system of promotion for users, either users in the system or totally new users. The approach here is to maximize techniques and knowledge in Udacity's course material such as finding similar users by `numpy.dot` products. Getting recommendation function working requires creative approach with the arrangement of transcript data so that summary data could be personal as possible and in line with user similarity form profile data. The recommendation also includes a "cold-start" challenge when we don't know anything about user. In this case, the function returns the ranked list of completion offer to the new users.

Working with simulated data can be a bit doubtful at first because at the end, the goal is the solve real problem and real data. However, as noticed by lecturers, data scientists have to run simulations or theorical testing before the real experiments taking place.

Secondly, it is a challenge to grasp the granular aspects of a big dataset. It always takes time to seep into data features and figure out a pathway to get through data and draw any insight out of it. Getting stuck with a common recommendation technique with the dataset in hand while the problem is asking for similar recommendation can be stressful until we have to change course and try alternative approach

11. Improvement

The project can be improved to serve to the overarching goal more effective, that is the recommend offers to user that returns a higher engagement of the customer to products and return to company the amount of spending. To improve the current recommendation system, a few things can be investigated:

1. The `find_similar_users()` function is till simple such as how calculating users' similarity based on `numpy.dot` with five variables. Although, the profile data is given as it-is, it would be better to have more features such as location, professional, zip code, interaction with other Starbucks product, to be better locating similar users.
2. Metrics to rate the successfulness of a transaction. It is my choice to select rate of completion based on number of completed, viewed, and received. We can define other metrics to test and see if new rating metrics could improve (or not change) the current metrics.
3. Recommending offers by a heuristic rules and subjective defined metrics is not the best approach to benchmark performance of different approach. My approach is also manual and lack of generalization like those with machine learning model. To be able for machine learning model working, we also need to agree on the rating of each transaction or `offer_id`, and then fixing the problem that lead to a more reliable decomposed user-latent and offer-latent feature matrices.