Udacity

Capstone Project Report

Starbucks Promotion Offer Recommendation

Suhas Karanth
5-23-2021

Contents

Project Overview:

Starbucks is a global organization that has over 32,000 coffeehouses, stores and roastery reserves. To improve the user experience to buy their products at their stores, Starbucks has mobile apps through which customers can enroll in to rewards program and place orders on various products to pick them up in the store. Starbucks has rolled out various campaigns to either reward the loyal customers or boost the sales or promote the new products. All customers may not respond in same manner and optimizing the process of offer rollout to maximize the response is important. Their current goal is to improve the process of offering these offers to their customers. So, for this project, Starbucks has provided a simulated data set from their mobile app to develop an algorithm that can recommend optimal offers to their customers that maximizes the response to the sent offer.

Problem statement:

Starbucks have various types of promotional offers for its customers, but do not know the appropriate type of offer to offer its customer. They want to avoid sending incorrect offer which might lead to negative user experience. They want to develop an algorithm that sends the promotional offer to the appropriate user and maximize the response rate.

Solution philosophy:

From an analytical standpoint, this problem can be tackled in multiple ways. It can be a classification problem where the appropriate offer must be classified among customers based on the propensity of the offer conversion. It can be a clustering problem where we segment the customers based on the propensity of offer conversion and recommend the offers that has high propensity of conversion. It can be a recommendation engine problem where we predict recommendations based on the customer-offer interactions.

If we were to treat this as a classification problem, we would be predicting the probability of conversion of an offer using characteristics/features of the customer.

Depending on the chosen solution, we must choose the evaluation metric to evaluate the model. For a classification exercise, we can choose accuracy measure as the primary metric and use precision, recall and F1 score to further validate the model. The accuracy, precision, recall and F1 score measures can be defined as-

$$Accuracy = \frac{Number\ of\ correct\ predictions}{Total\ number\ of\ predictions}$$

$$Precision = \frac{Number\ of\ correctly\ predicted\ offer}{Total\ number\ of\ predictions\ of\ the\ corresponding\ offer}$$

$$Recall = \frac{Number\ of\ correctly\ predicted\ offer}{Total\ number\ of\ actual\ corresponding\ offer}$$

$$F1\ score = \frac{2}{\frac{1}{Precision} + \frac{1}{Recall}}$$

For clustering, we can use silhouette coefficient to evaluate the clusters. It is defined for each of the clusters-

$$silhouette\ coefficient = \frac{\mathrm{D1} - \mathrm{D2}}{Max(D1, D2)}$$

D1: The mean distance between a sample and all other points in the same cluster

D2: The mean distance between a sample and all other points in the next nearest cluster

This value is bounded between -1 and 1 and high values indicate dense clustering that is well separated. Values around 0 indicate the overlapping of the clusters.

For recommendation engine, we'll use precision@k, recall@k and F1 score@k to evaluate the model. They are defined as-

$$recall@k = \frac{\#\ of\ true\ offers\ predicted}{\#\ of\ true\ offers}$$

$$precision@k = \frac{\#\ of\ true\ offers\ predicted}{\#\ of\ offers\ predicted}$$

$$F1@k = \frac{2}{\frac{1}{precision@k} + \frac{1}{recall@k}}$$

'@k' refers to number of recommendations made by the recommendation engine.

In this project, we'll explore the data a bit more to evaluate the best way to develop the model and recommendations of offers

Datasets and inputs:

There are three types of offers that can be sent: buy-one-get-one (BOGO), discount, and informational. In a BOGO offer, a user needs to spend a certain amount to get a reward equal to that threshold amount. In a discount, a user gains a reward equal to a fraction of the amount spent. In an informational offer, there is no reward, but neither is there a requisite amount that the user is expected to spend. Offers can be delivered via multiple channels.

There are 3 data sets available to develop the algorithm-
profile.json:- Data set containing the list of customers who are members of rewards program. This data set contains user specific details such as gender, age, income, and age of membership. The details-

- age (int) - age of the customer
- became_member_on (int) - date when customer created an app account

- gender (str) - gender of the customer (note some entries contain 'O' for other rather than M or F)
- id (str) - customer id
- income (float) - customer's income

portfolio.json:- Data set contains details about the offers. It contains information like the reward of the offer, channels through which the offers are presented, conditions to use the offer, duration of the offer, and type of the offer. The details-

- id (string) - offer id
- offer_type (string) - type of offer i.e. BOGO, discount, informational
- difficulty (int) - minimum required spend to complete an offer
- reward (int) - reward given for completing an offer
- duration (int) - time for offer to be open, in days
- channels (list of strings) - channels through which the offer can be sent to the customer

transcript.json:- This data set contains the activity details of the customers of Starbucks who are enrolled in to rewards program. The data set contains the details such as amount of the transaction, type of offer used, duration of the offer use before expiry, and reward obtained for the offer. The details-

- event (str) - record description (i.e. transaction, offer received, offer viewed, etc.)
- person (str) - customer id
- time (int) - time in hours since start of test. The data begins at time t=0
- value - (dictionary of strings) - either an offer id or transaction amount depending on the record

Data cleaning-

All the data sets require some form of cleaning before it can be analyzed.

Following steps are followed to clean the data-

Portfolio data:

- The duration column is in ‘days’. And the unit of time in transcript data is hours. So, we multiply the duration column by 24 to match the units in the transcript data
- Converted the ‘offer_type’ column in to dummy columns in case we need to use them as a feature in modeling exercise
- Created channels dummy variable as additional features

reward	difficulty	duration	id	web	email	mobile	social	bogo	discount	informational
10	10	168	ae264e3637204a6fb9bb56bc8210ddfd	0	1	1	1	1	0	0
10	10	120	4d5c57ea9a6940dd891ad53e9dbe8da0	1	1	1	1	1	0	0
0	0	96	3f207df678b143eea3cee63160fa8bed	1	1	1	0	0	0	1
5	5	168	9b98b8c7a33c4b65b9aebfe6a799e6d9	1	1	1	0	1	0	0
5	20	240	0b1e1539f2cc45b7b9fa7c272da2e1d7	1	1	0	0	0	1	0
3	7	168	2298d6c36e964ae4a3e7e9706d1fb8c2	1	1	1	1	0	1	0
2	10	240	fafdcd668e3743c1bb461111dcafc2a4	1	1	1	1	0	1	0
0	0	72	5a8bc65990b245e5a138643cd4eb9837	0	1	1	1	0	0	1
5	5	120	f19421c1d4aa40978ebb69ca19b0e20d	1	1	1	1	1	0	0
2	10	168	2906b810c7d4411798c6938adc9daaa5	1	1	1	0	0	1	0

Profile data:

- Identified missing data and checked for the possibility of data imputation.
- Realized that the missing data, missed information about age and income entirely. So, I eliminated those customers from the data exploration for feature engineering.

Non-missing data

	age	became_member_on	income
count	14825.000000	1.482500e+04	14825.000000
mean	54.393524	2.016689e+07	65404.991568
std	17.383705	1.188565e+04	21598.299410
min	18.000000	2.013073e+07	30000.000000
25%	42.000000	2.016052e+07	49000.000000
50%	55.000000	2.017080e+07	64000.000000
75%	66.000000	2.017123e+07	80000.000000
max	101.000000	2.018073e+07	120000.000000

Missing data

	age	became_member_on	income
count	2175.0	2.175000e+03	0.0
mean	118.0	2.016804e+07	NaN
std	0.0	1.009105e+04	NaN
min	118.0	2.013080e+07	NaN
25%	118.0	2.016070e+07	NaN
50%	118.0	2.017073e+07	NaN
75%	118.0	2.017123e+07	NaN
max	118.0	2.018073e+07	NaN

Transcript data:

- Transform the data wide form to observe the type of event for each customer-offer combination
- Iterate through all the customers to tie up the offer received, viewed, and completed along with the transaction amount associated with each offer.

id_offer	trans_amt	num_times_received	num_times_viewed	num_times_completed	total_amt_spent_towards_offer	avg_amt_spent_towards_offer
b245e5a138643cd4eb9837	NaN	1	1	0	22.16	22.16
b245e5a138643cd4eb9837	NaN	1	1	0	22.16	22.16
None	22.16	0	0	0	0.00	0.00
b143eea3cee63160fa8bed	NaN	1	1	0	8.57	8.57
b143eea3cee63160fa8bed	NaN	1	1	0	8.57	8.57
...	...	...	...	...	...	...
None	14.23	0	0	0	0.00	0.00
d4411798c6938adc9daaa5	NaN	3	3	3	97.34	32.45
d4411798c6938adc9daaa5	NaN	3	3	3	97.34	32.45
None	10.12	0	0	0	0.00	0.00
None	18.91	0	0	0	0.00	0.00

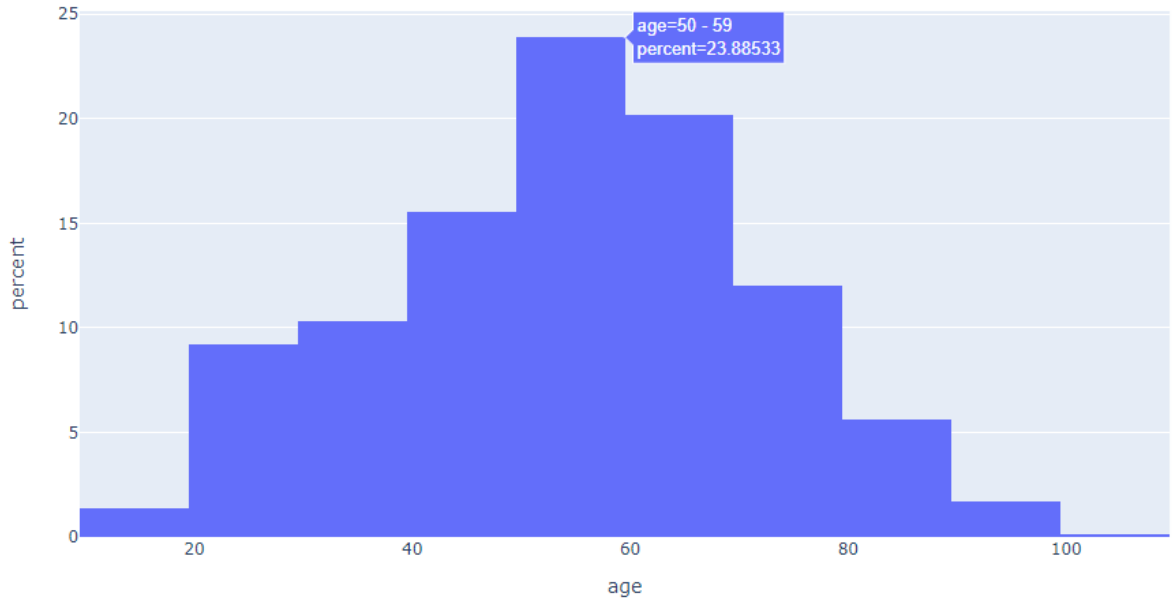
We see that a customer can receive same offer multiple times.

Once we have done the basic cleaning, we'll explore the data in detail to identify the feature relationships

Data exploration:

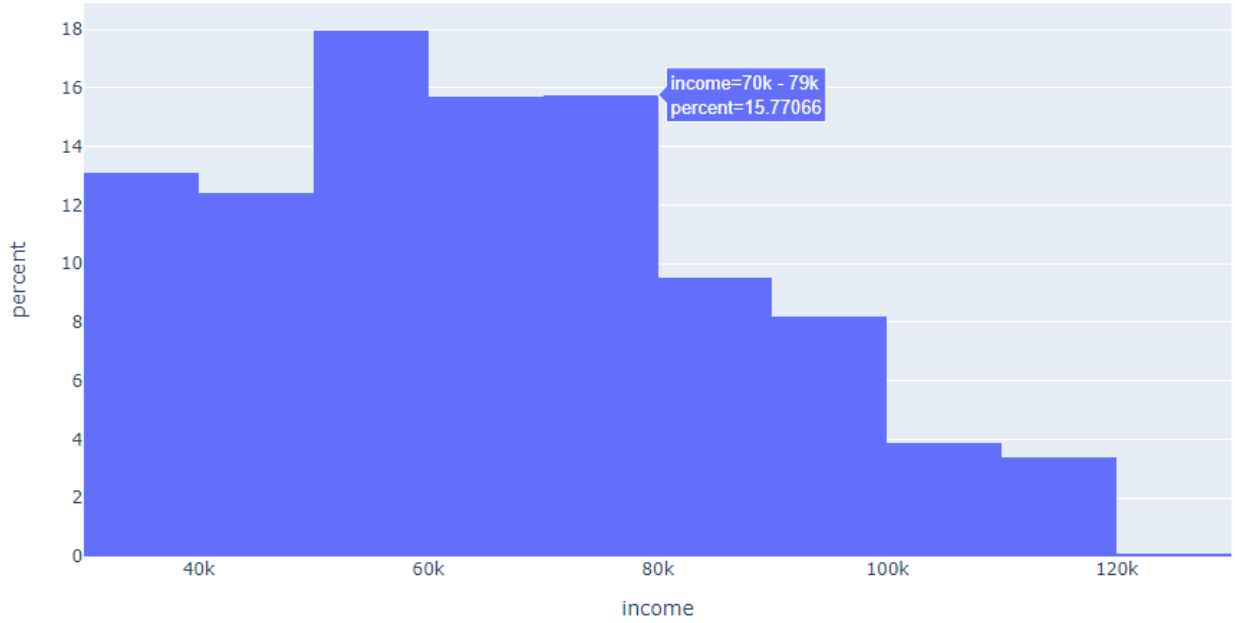
We'll start with exploring the basic data such as distribution of key features of customers such as age, sex, and income. Then, we'll check how are offers distributed across customers in terms of received, viewed, and completed.

Distribution of age of customers-



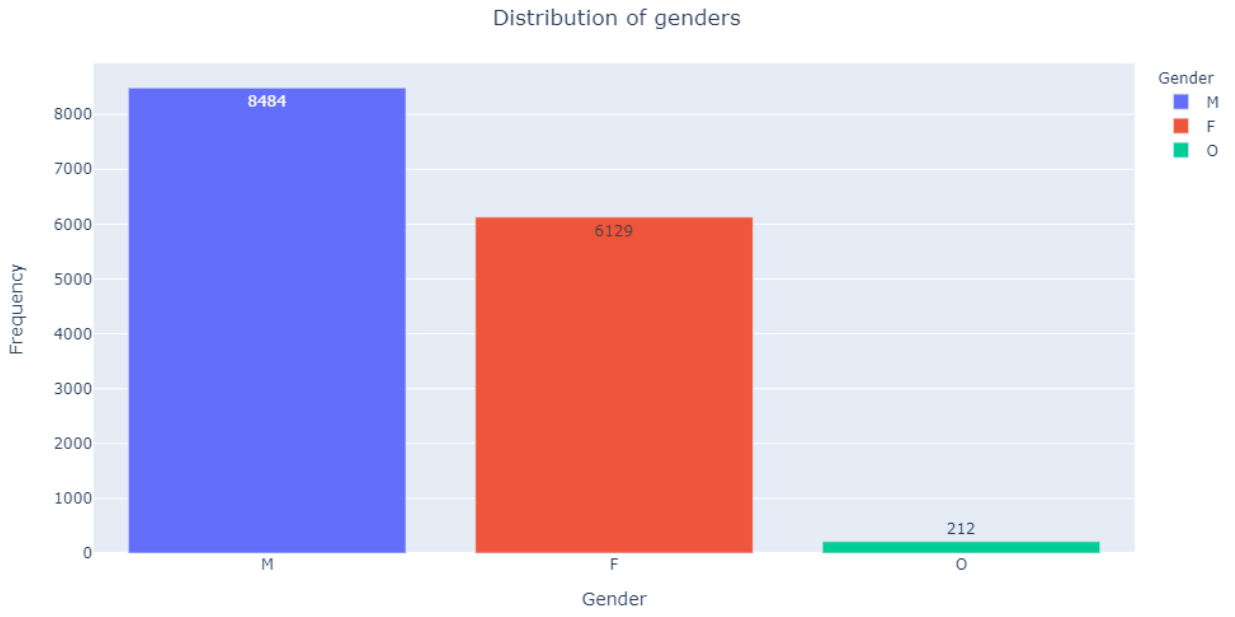
As we can see, age of customers is almost normally distributed, with average of about 55 years. Majority of the customers are in between 40 and 70 years.

Distribution of income of customers-



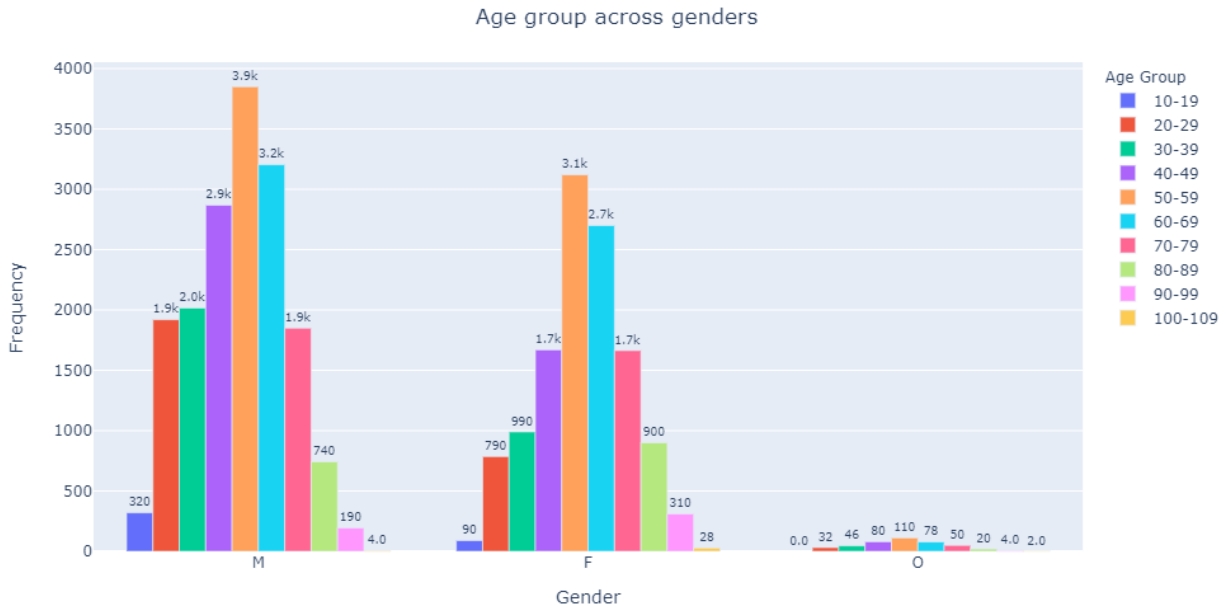
Majority of the customers' income is in between 30k and 80k and it is quite uniformly distributed.

Distribution of gender-



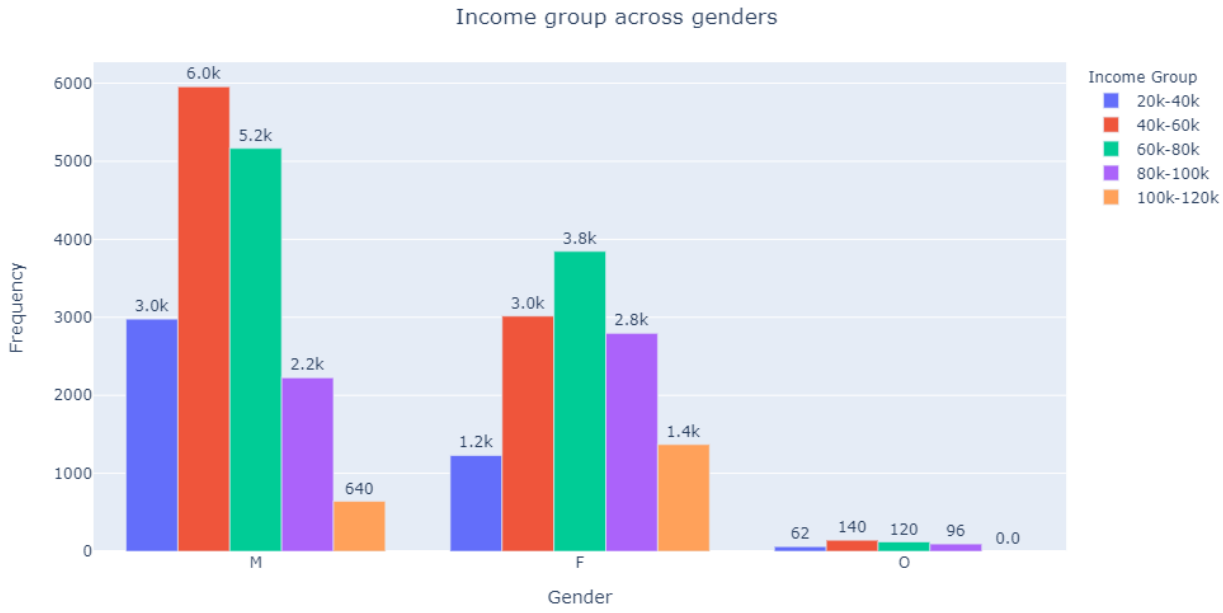
Males form most of the customers and ‘others’ are small portion in the entire data.

Let’s see how the age group varies across genders



Age group distribution is very similar for both male and females

Let's see how the income varies across genders

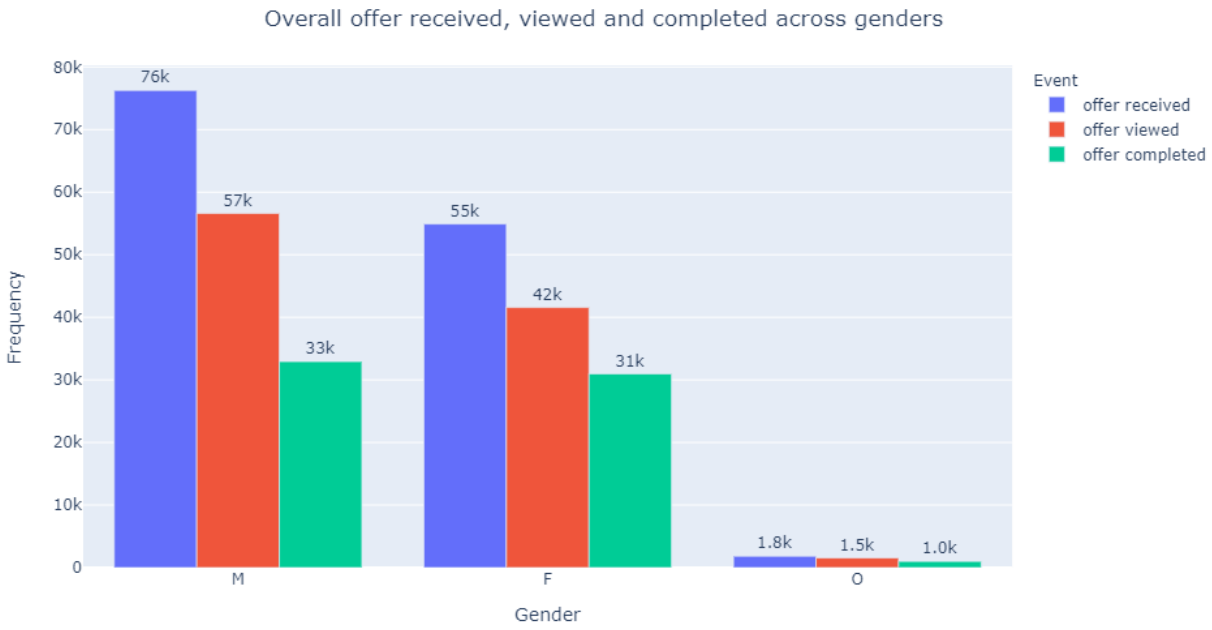


Males have slightly different distribution than females. Male customers' income is slightly lower than female customers' income. Female customers' income group distribution looks normally distributed whereas male customers' leans slightly towards left of the bell-shaped curve.

Now let's see how the offers are received, viewed, and completed across genders, income groups and age groups.

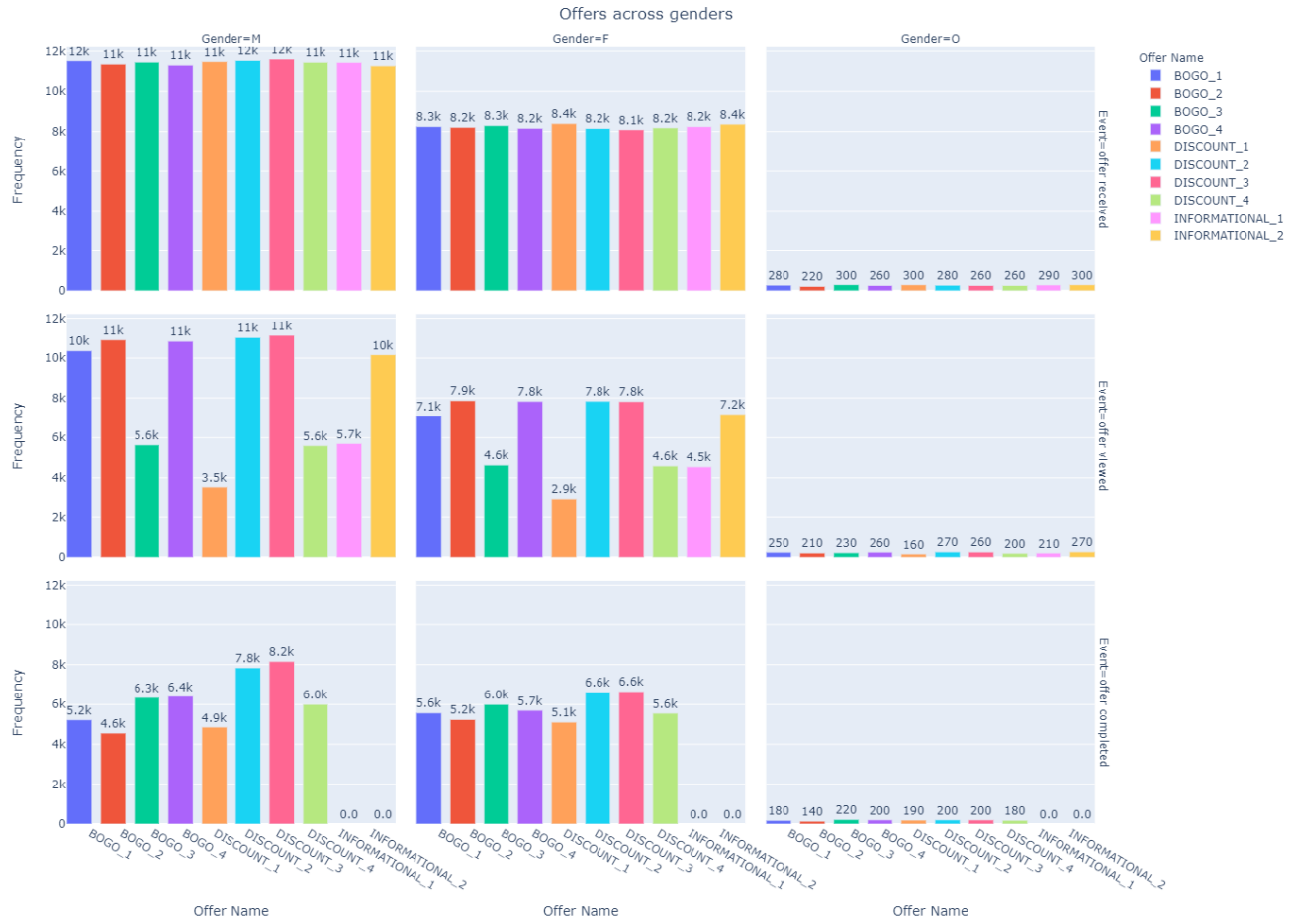
Offers received, viewed, and completed-

Across Gender:



gender	view_receive_ratio	completed_receive_ratio	completed_viewed_ratio
F	0.757066	0.563702	0.744588
M	0.742243	0.431850	0.581817
O	0.843886	0.546943	0.648124

The distribution of offer received, viewed, and completed looks similar across the genders. The male customers have slightly lower rate of completion compared to female customers. We must look across different offers if they show any significant variation.



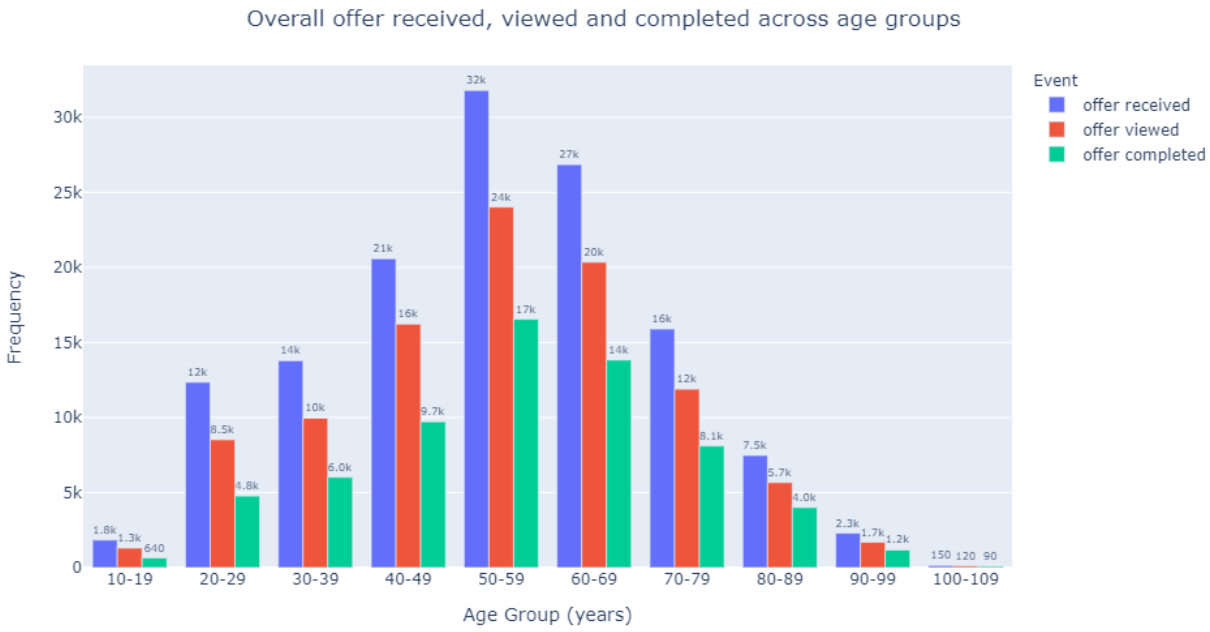
Offer received across offers is uniform across genders. And the distribution of viewed and completed across genders look similar.

Offer_Name	F	M	O
BOGO_1	5.0	8.0	8.0
BOGO_2	6.0	9.0	9.0
BOGO_3	8.0	6.0	4.0
BOGO_4	4.0	4.0	2.0
DISCOUNT_1	10.0	10.0	10.0
DISCOUNT_2	3.0	3.0	6.0
DISCOUNT_3	2.0	2.0	3.0
DISCOUNT_4	7.0	7.0	7.0
INFORMATIONAL_1	9.0	5.0	5.0
INFORMATIONAL_2	1.0	1.0	1.0

The above table ranks the offers across genders in terms of propensity of converting an offer in to completed. Across genders, discount_1 seems to be the least popular offer and discount 3 is

the most popular offer (non-informational). This implies that gender doesn't seem have an impact on offers viewed or completed.

Across age groups:



All offers received, viewed, and completed have similar patterns across all age groups. Let's check if there is a variation across offers.

Offer_Name	10-19	100-109	20-29	30-39	40-49	50-59	60-69	70-79	80-89	90-99
BOGO_1	6.0	5.0	6.0	7.0	8.0	5.0	5.0	5.0	5.0	5.0
BOGO_2	9.0	2.0	9.0	9.0	9.0	6.0	9.0	7.0	6.0	6.0
BOGO_3	7.0	4.0	5.0	6.0	5.0	8.0	7.0	9.0	7.0	8.0
BOGO_4	4.0	2.0	4.0	4.0	7.0	4.0	4.0	4.0	4.0	4.0
DISCOUNT_1	10.0	7.0	10.0	10.0	10.0	10.0	10.0	10.0	10.0	10.0
DISCOUNT_2	2.0	4.0	2.0	3.0	2.0	3.0	3.0	3.0	3.0	3.0
DISCOUNT_3	3.0	3.0	3.0	2.0	3.0	2.0	2.0	2.0	2.0	2.0
DISCOUNT_4	8.0	6.0	8.0	5.0	6.0	9.0	6.0	8.0	9.0	7.0
INFORMATIONAL_1	5.0	4.0	7.0	8.0	4.0	7.0	8.0	6.0	8.0	9.0
INFORMATIONAL_2	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0

The above table ranks the offers across age groups in terms of propensity of converting an offer in to completed. Across age groups, across offers received, viewed, and completed there doesn't seem to be any significant variation. discount_3 seems to be the most converted non-informational offer across most of the age groups and discount_1 seems to be the least popular offer across most of the age groups. This implies that age doesn't seem have an impact on offers viewed or completed.

Across Income groups:

The distribution of gap between offer received, viewed, and converted seems uniform across income groups. Let's check how they vary across various offers.

Offer_Name	100k-120k	20k-40k	40k-60k	60k-80k	80k-100k
BOGO_1	5.0	8.0	8.0	6.0	4.0
BOGO_2	4.0	9.0	9.0	9.0	7.0
BOGO_3	9.0	5.0	6.0	7.0	9.0
BOGO_4	3.0	4.0	4.0	4.0	5.0
DISCOUNT_1	10.0	10.0	10.0	10.0	10.0
DISCOUNT_2	2.0	2.0	3.0	3.0	2.0
DISCOUNT_3	1.0	3.0	2.0	2.0	1.0
DISCOUNT_4	8.0	7.0	5.0	8.0	6.0
INFORMATIONAL_1	7.0	6.0	7.0	5.0	8.0
INFORMATIONAL_2	6.0	1.0	1.0	1.0	3.0

The above table ranks the offers across age groups in terms of propensity of converting an offer in to completed. Majority of the offers are offered to the income group of 40k-60k and 60k-80k and their response to the offers is quite identical. Even the other income groups, the variation is not much and discount_1 seems to be the least popular offer across income groups. This implies that income as well doesn't have any significant impact on the offer being converted.

Almost all the features don’t seem to be showing variation with respect to the propensity of an offer being converted, so developing a classification model or clustering model would require intensive feature engineering that are mainly extracted from transaction data. Recommendation engine that utilizes user-offer combination seems to be a best alternative given the status of the data and I would be proceeding with that. In terms of algorithm, I would use FunkSVD algorithm and estimate appropriate latent features that would help me predict correct recommendations. As a benchmark model, I would use the most popular offers in terms of success of conversion (product of [minimum(number of offers viewed, number of offers completed)/(number of offers received)] and mean amount of transaction towards the respective offers)

Data preprocessing:

To use FunkSVD algorithm, data must be pre-processed. The algorithm takes the data in the form of a matrix where rows are customers and columns are the offers offered. The values within this matrix must be something that represents the affinity of customers towards the offer. So, we calculated a metric that provides that indication as the data doesn't offer customer feedback to the offer. The metric is a simple ratio between number of offers completed and number of offers offered and must be calculated per customer-offer combination. The rationale behind this is, if an offer wasn't completed, then the customer didn't like the offer. This metric in a way, can be treated as 'rating' of offer.

We know that the customer can complete the offer without viewing it. So, while calculating the metric, we take the minimum of number of offers viewed and completed to calculate the ratio. The final formula of the measure is,

$$Success\ ratio = \frac{\min(number\ of\ offers\ viewed, number\ of\ offers\ completed)}{number\ of\ offers\ received}$$

The above metric works well for non-informational offers that can be completed after viewing. But informational offers cannot be completed. So, we use offers viewed and offers received to calculate this ratio.

We calculate this ratio for every customer-offer combination.

id_customer	id_offer	success_ratio
0009655768c64bdeb2e877511632db8f	f19421c1d4aa40978ebb69ca19b0e20d	1.0
0009655768c64bdeb2e877511632db8f	fafdcd668e3743c1bb461111dcafc2a4	1.0
0009655768c64bdeb2e877511632db8f	2906b810c7d4411798c6938adc9daaa5	0.0
0011e0d4e6b944f998e987f904e8c1e5	2298d6c36e964ae4a3e7e9706d1fb8c2	1.0
0011e0d4e6b944f998e987f904e8c1e5	0b1e1539f2cc45b7b9fa7c272da2e1d7	1.0
...	...	...
ffeaa02452ef451082a0361c3ca62ef5	5a8bc65990b245e5a138643cd4eb9837	1.0
fff0f0aac6c547b9b263080f09a5586a	3f207df678b143eea3cee63160fa8bed	1.0
fff3ba4757bd42088c044ca26d73817a	5a8bc65990b245e5a138643cd4eb9837	0.5
fff8957ea8b240a6b5e634b6ee8eafcf	3f207df678b143eea3cee63160fa8bed	0.0
fffad4f4828548d1b5583907f2e9906b	5a8bc65990b245e5a138643cd4eb9837	1.0

Then divide the data set to training, validation, and test set for the modeling. After we divide them, we convert the data to matrix form so that a model can be developed.

Customer	Offer_1	Offer_2	Offer_3	Offer_4	Offer_5	Offer_6	Offer_7	Offer_8	Offer_9	Offer_10
0009655768c64bdeb2e877511632db8f	NaN	NaN	0.0	NaN	NaN	NaN	NaN	NaN	1.0	NaN
0011e0d4e6b944f998e987f904e8c1e5	NaN	1.0	NaN	1.0	NaN	1.0	NaN	NaN	NaN	NaN
0020c2b971eb4e9188eac86d93036a77	NaN	NaN	NaN	NaN	1.0	1.0	NaN	NaN	NaN	NaN
0020ccbbb6d84e358d3414a3ff76cffd	NaN	1.0	NaN	NaN	NaN	1.0	NaN	NaN	NaN	NaN
003d66b6608740288d6cc97a6903f4f0	NaN	NaN	NaN	1.0	NaN	NaN	NaN	NaN	NaN	1.0
...	...	...	...	...	...	...	...	...	...	...
fff3ba4757bd42088c044ca26d73817a	NaN	NaN	NaN	NaN	NaN	0.5	NaN	NaN	NaN	NaN
fff7576017104bcc8677a8d63322b5e1	NaN	NaN	NaN	NaN	NaN	NaN	0.0	NaN	NaN	NaN
fff8957ea8b240a6b5e634b6ee8eafcf	NaN	NaN	NaN	0.0	NaN	NaN	NaN	NaN	NaN	NaN
fffad4f4828548d1b5583907f2e9906b	NaN	NaN	NaN	NaN	NaN	1.0	NaN	NaN	NaN	NaN
ffff82501cea40309d5fdd7edcca4a07	NaN	NaN	1.0	NaN	NaN	NaN	NaN	NaN	NaN	1.0

The missing values need not be treated in the matrix as they are the offers not offered to the customers and the model that we have chosen works very well with sparse matrix such as this.

Modeling:

Now that we have created the necessary data sets for the modelling, we'll write the function that will estimate the success ratio based on the FunkSVD algorithm. A brief snippet about the algorithm below-

Matrix Factorization is the core methodology through which we will be predicting the user offer rating. Matrix factorization decomposes customer-offer matrix in to 2 matrices whose product results into original matrix. Funk SVD is a variant of this methodology and was developed by Simon Funk in 2006. Below is the example of matrix factorization done by Singular Vector Decomposition-

Matrix A				Matrix U				Matrix Σ				Matrix V^T			
59	86	49	=	3	1	3	X	4	0	0	X	1	4	1	
101	134	81		4	2	5		0	5	0		4	4	2	
96	144	76		5	2	4		0	0	3		3	2	3	
62	68	42		1	2	2									

We notice that the original matrix has values at all positions. Matrix factorization requires values to be present in all positions for the decomposition to work. But our data set have lots of values that are missing. That's where Funk SVD comes in to play.

Funk SVD estimates the decomposed matrices by using only the known values of the original matrix. This estimation is done iteratively by calculating the error of the estimates at each iteration and adjusting the estimates to further reduce the error. The error measure here is sum of squared errors and the method through which the adjustments are made is gradient descent. Below illustration shows how the method works-

Customer-Offer Matrix

	O1	O2	O3
C1		1	
C2	0.25		
C3			1
C4		0.45	
C5			0.75
C6	0.66		

≈

Customer-Latent (C)

	L1	L2	L3
C1	0.1	0.28	0.43
C2	0.12	0.47	0.23
C3	0.56	0.58	0.63
C4	0.76	0.09	0.67
C5	0.54	0.67	0.99
C6	0.73	0.1	0.12

X

Latent-Offer (O)

	O1	O2	O3
L1	0.98	0.78	0.22
L2	0.54	0.34	0.45
L3	0.12	0.90	0.21

Suppose the customer-offer matrix is available as illustrated above, and we want to factorize it with 2 matrices with 3 latent factors, we generate 2 matrices (Matrix C and Matrix O) with random values where the first matrix has all the customers on rows and latent factors on the column and the second matrix has latent factors on the rows and offers on columns. We then check the difference between the predicted and actual where the actual rating is present. For example, for C1 and O2, difference between actual and predicted are-

Predicted: 0.1*0.78 + 0.28*0.34 + 0.43*0.9 = 0.5602

Difference: Predicted-Actual = 0.5602-1 = -0.4398

The update rule is:

$$C_{new} = C_{current} + \alpha * 2 * (predicted - actual) * O_{current}$$

$$O_{new} = O_{current} + \alpha * 2 * (predicted - actual) * C_{current}$$

Here, C and O are Customer-Latent and Latent-Offer matrix respectively. α is learning rate through which the update of values happens. Let's assign it as 0.01. Substituting the values,

$$C_{new} = \begin{matrix} 0.10+0.01*2*(-0.4398)*0.78 \\ 0.28+0.01*2*(-0.4398)*0.34 \\ 0.43+0.01*2*(-0.4398)*0.90 \end{matrix} = \begin{matrix} 0.093139 \\ 0.277009 \\ 0.422084 \end{matrix}$$

$$O_{new} = \begin{matrix} 0.78+0.01*2*(-0.4398)*0.10 \\ 0.34+0.01*2*(-0.4398)*0.28 \\ 0.90+0.01*2*(-0.4398)*0.43 \end{matrix} = \begin{matrix} 0.77912 \\ 0.337537 \\ 0.896218 \end{matrix}$$

So, using the above rule we replace the values in the first row of Customer-Latent Matrix by 0.093139, 0.277009, 0.422084 and second column of Latent-Offer matrix by 0.77912, 0.337537 and 0.896218. We run this for all the customer-offer combinations and pre-determined number of iterations to reduce the overall error and obtain the minimum.

So, we use this algorithm to estimate the decomposed matrices. Once the matrices are generated, their product would contain the predictions for all the combinations of customer-offer success ratio. Through that, we can predict the customer behavior for the offers he/she didn't receive.

Model Development & Validation:

The model will be developed using training data and validation data. The function takes in these 2 data sets along with learning rate and number of latent features.

We observed that the validation error reached minimum around 40-70 iterations for default number of iterations (100) and learning rate (0.005) when executed for latent features of 7 or 8. This implied that the model trained quickly. So, we had to estimate the apt number of latent features and iterations that balances training and validation error.

Below is the result of the model hyperparameter tuning of number of latent features-

Iterations	MSE Train	MSE Validation
47.0	0.117288	0.194296
60.0	0.124518	0.187516
65.0	0.134947	0.184376
65.0	0.143176	0.183297
59.0	0.154401	0.182817
56.0	0.162697	0.183825
46.0	0.174324	0.186869
42.0	0.179996	0.186722
38.0	0.184969	0.183812
31.0	0.192178	0.190010

Each row represents the number of latent features starting from 1. And the values (training data MSE and iterations) correspond to the lowest validation MSE among 100 iterations for the selected number of latent features.

As we can see, for 4 latent features and 65 iterations, both training error and validation error balances out. After 4 latent features, the training error for the corresponding lowest validation error increases and the lowest validation error also increases. So, we fix the model with 65 iterations and 4 latent features.

We then develop the final model using full training data set using the features estimated in previous step.

Results:

The final model is then evaluated on the new data that is test data. This evaluation is done by precision@k, recall@k and F1 score@k as discussed earlier. ‘k’ is the number of recommendations done by the model. So, we tested the model for different values of ‘k’ to test the robustness of the model. Precision, recall and F1 score requires True label and a false label to calculate the metric. To assign ‘True’ and ‘False’, we categorized success ratio of an offer that is above 0.5, as relevant offer to the customer, else it is not.

Precision@k would be the ratio between number of true recommendations predicted and number of all the predictions for k recommendations done by the model

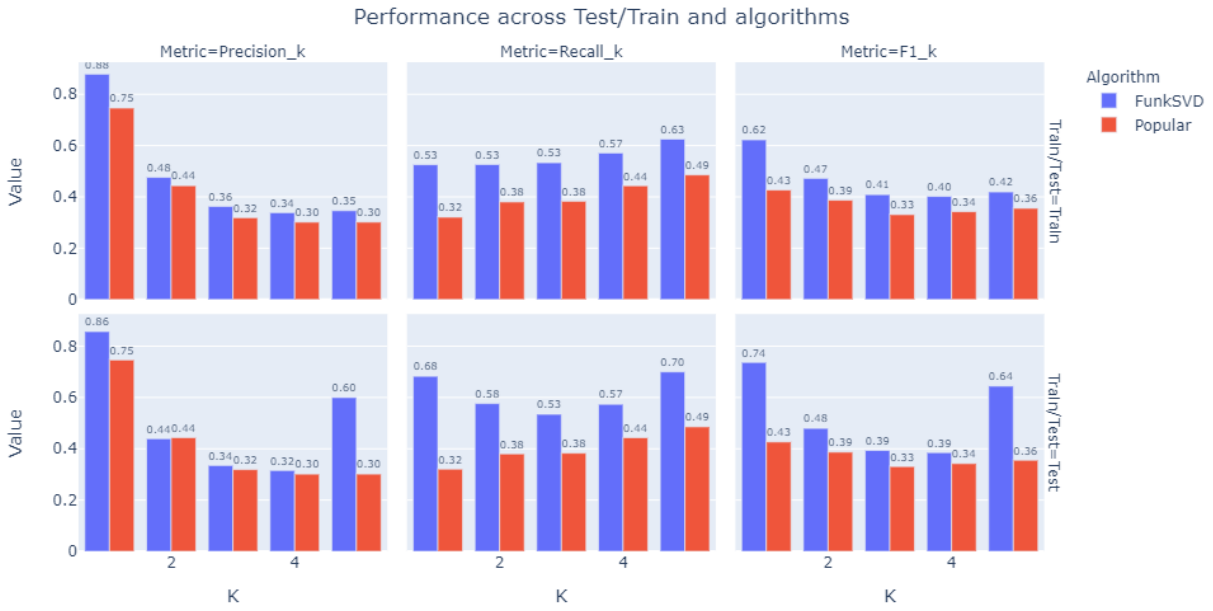
Recall@k would be the ratio between number of true recommendations predicted and total number of true recommendations for k recommendations done by the model

F1 score@k would be the harmonic mean between Precision@k and Recall@k

By the definition, we can see that these metrics must be calculated per customer and then average them to get the sense of the overall performance.

As the benchmark model, we selected the most popular offers offered to be recommended to all the customers. We calculated the above metrics for those recommendations too to compare it with the Funk SVD model.

Below are the results-



As we can see, the model performs well in the test data implying the developed model generalizes well. We also see that for all k recommendations ranging from 1-5, Funk SVD model performs better than the benchmark model.

Conclusion:

Funk SVD model is a powerful recommendation engine model that is based on collaborative filtering methodology. This means, that it is a powerful and a great tool to recommend offers to an existing customer base. If the customer already exists in the system, and has had a decent history, then this methodology works very well. The model fails when it comes to predict for a new customer. This is a limitation of the model developed and must be used keeping that in mind. Otherwise, the model throws light on each customer's probable behavior towards the

offers that were not offered in the past. This helps the marketing department to make an informed decision about targeting right set of customers for their campaigns.

Further improvements:

- The developed model doesn't predict for a new customer. This can be mitigated by developing a content-based model that considers user features such as age, sex, income, other demographic details
- Try different techniques to optimize the gradient instead of gradient descent to obtain better accuracy
- Try implementing deep neural networks to achieve matrix factorization to achieve higher accuracy than the current model

References:

- Matrix Factorization- https://rpubs.com/Argaadya/recommender-svdf
- Precision, Recall and F score- https://medium.com/@m_n_malaeb/recall-and-precision-at-k-for-recommender-systems-618483226c54#:~:text=In%20the%20computation%20of%20precision,precision%20at%20k%20to%201.
- Funk SVD- https://sifter.org/~simon/journal/20061211.html