Tay Cong Run
24 June 2021

CAPSTONE PROPOSAL

MACHINE LEARNING ENGINEER NANODEGREE

Contents

Tay Cong Run
24 June 2021

Domain Background

Starbucks is a prominent multinational beverage service provider, specializing in handmade artisan coffee. It introduced the Starbucks® mobile app to appeal to its loyal, tech-savvy customers. To date, there are approximately 16 million active Starbucks® Rewards members (Sokolowsky, 2019) and possibly more. This opens up a whole realm of data inflow. Starbucks can analyse static historical data to better Know Your Customer (KYC). It can also push the boundaries by continually pushing tailored offers to customers to boost sales and gather passive A/B testing data at the same time, in a closed feedback loop.

Starbucks' customer base spans across multiple nations, regions and continents. Traditional qualitative sales strategies meetings in a boardroom can only get a firm so far in knowing its customers and deriving sales techniques from known customer behaviours. With a rapidly changing and growing customer base, it naturally becomes difficult for analysts to spot useful trends and for business teams to implement sales strategies built upon that, quickly enough to materialize these prospects across multiple regions. The use of a machine learning pipeline tied to this mobile application is apt to tackle this implementation lag. It could help to centralize and/or automate business information and decisions, which would allow Starbucks to provide recommendations that are more tailored to each customer, in a timelier manner.

Tay Cong Run
24 June 2021

Problem Statement

The Starbucks® mobile app frequently has offers in a bid to boost sales. There is room for optimization in the way these offers are pushed to its members. Depending on each user's taste, preference and consumption patterns, certain offers may appeal more to the user and/or the user may complete certain offers more easily. The problem statement is to **determine which demographic groups respond best to which offer type**, which would subsequently allow a **customised ranking of offers for each user**.

To better define 'responsiveness to offers', we define it as the **increase in average spending per day, after subtracting the points rewarded**. This ensures that our solution accounts for the sales boost attributed by an offer, while penalizing an offer pushed to a user that would have completed that it anyway.

$Y_{i,j,k} = \frac{transacted\ amount_{i,j,k} - offer\ reward_i}{duration_{i,j,k}}$ for each offer i, user j, and influence period k

The proposed solution is to use a regression machine learning algorithm to predict, for each user, the **daily average of transaction less offer reward** for each offer as well as no offer. Each offer can then be ranked by the **average daily transaction less offer reward** for that offer subtracted by that when no offer is made.

Evaluation Metric

Models

We have outlined a regression problem where the predicted value is continuous. The metric to be used is the **R-squared coefficient**. It is a scorer that takes a value between 0 and 1, representing the percentage of variability in the predicted variable that is explained by the predictor variables through the model, or simply put the goodness of fit. It is very commonly used to score regression models.

Heuristics

At each juncture an offer is pushed to a user, we compare 3 scenarios:

1) Offer A, which is randomly drawn from the 10 offers
2) Offer B, the offer that was actually sent
3) Offer C, the offer that would lead to the best response according to our machine learning model

Note: For Offers A & C, there is an added option of **not sending an offer**.

Tay Cong Run
24 June 2021

How well a set of offer recommendation heuristics that **recommends offer *i*** perform is calculated based on the **increase in $\hat{Y}_{i,j}$, the daily average of transaction less offer reward** i.e. $\sum_{j}^{J}(\hat{Y}_{i,j} - \hat{Y}_{no\ offer,j})$ **across all offer periods *j***.

Tay Cong Run
24 June 2021

Data Exploration

The data collected from the backend of the Starbucks application are *users*, *offers* and *transactions* data.

Users Data

The users dataset contains exactly **17000** data points, each representing a unique user ID on the Starbucks® mobile app. In addition, we have their:

- Age
- Date of registration
- Gender* (*male, female, others*)
- Income*

* - contains nulls

	gender	age	person	became_member_on	income
0	None	118	68be06ca386d4c31939f3a4f0e3dd783	20170212	NaN
1	F	55	0610b486422d4921ae7d2bf64640c50b	20170715	112000.0
2	None	118	38fe809add3b4fcf9315a9694bb96ff5	20180712	NaN
3	F	75	78afa995795e4d85b5d9ceeca43f5fef	20170509	100000.0
4	None	118	a03223e636434f42ac4c3df47e8bac43	20170804	NaN

FIGURE 1: PREVIEW OF USERS DATA FROM PROFILE.JSON

Offers Data

The offers dataset contains the following information about **10** different offers:

- Type (*buy-one-get-one, discount, informational*)
- Channels of distribution (*web, email, mobile, social*)
- Duration (*in days*)
- Difficulty (*minimum required spend to complete an offer*)
- Points rewarded upon completion

	reward	channels	difficulty	duration	offer_type	offer_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 2: OFFER DATA FROM PORTFOLIO.JSON

Transactions Data

The Transactions Data contains **306534** data points, each representing an event. In addition, we know the:

- Type of event (*offer received, offer viewed, transaction, offer completed*)
- Pertained user ID
- Time elapsed since the start of test (in hours)
- Relevant information (*offer id, amount transacted, reward for offer completion*)

	person	event	value	time
0	78afa995795e4d85b5d9ceeca43f5fef	offer received	{'offer id': '9b98b8c7a33c4b65b9aebfe6a799e6d9'}	0
12650	389bc3fa690240e798340f5a15918d5c	offer viewed	{'offer id': 'f19421c1d4aa40978ebb69ca19b0e20d'}	0
12654	02c083884c7d45b39cc68e1314fec56c	transaction	{'amount': 0.8300000000000001}	0
12658	9fa9ae8f57894cc9a3b8a9bbe0fc1b2f	offer completed	{'offer_id': '2906b810c7d4411798c6938adc9daaa5...	0

FIGURE 3: PREVIEW OF TRANSACTIONS DATA FROM TRANSCRIPT.JSON

Preprocessing of Transactions Data are as follows:

1) Extract *offer_id* or *offer id* from **offer *** *event*.
2) Extract *amount* from **transaction** *event*.
3) Extract *reward* from **offer completed** *event*.

	person	event	time	offer_id	amount	reward
0	78afa995795e4d85b5d9ceeca43f5fef	offer received	0	9b98b8c7a33c4b65b9aebfe6a799e6d9	NaN	NaN
12650	389bc3fa690240e798340f5a15918d5c	offer viewed	0	f19421c1d4aa40978ebb69ca19b0e20d	NaN	NaN
12654	02c083884c7d45b39cc68e1314fec56c	transaction	0	NaN	0.83	NaN
12658	9fa9ae8f57894cc9a3b8a9bbe0fc1b2f	offer completed	0	2906b810c7d4411798c6938adc9daaa5	NaN	2.0

FIGURE 4: PREVIEW OF TRANSACTIONS DATA AFTER PREPROCESSING

Data Transformation

To analyse $Y_{i,j,k}$ meaningfully, we breakdown the timeline for users into **offer periods** (from an *offer received* event until either an *offer completed* event or the offer expires from reaching its *duration*) and **non-offer periods** (any other time). **Offer periods** are further broken down, based on whether the offer was viewed, into **offer influence periods** (from an *offer viewed* event until either an *offer completed* event or the offer expires from reaching its *duration*) and **non-influence periods** (any other time).

The following steps are taken to transform Transactions Data from the granularity of **per event**, to a granularity of **per user per offer period**:

1) Find **offer periods** that were applied a user during the experiment. This includes finding the *start_time* from when an offer is received, *end_time* if the offer expires after its *duration*, *view_time* if the user viewed that offer, and *complete_time* if the user eventually completes that offer.

	person	event	time	offer_id	amount	reward
5101	6263d948f09b4cfc87b0b59955f269f1	offer received	0	2906b810c7d4411798c6938adc9daaa5	NaN	NaN
13855	6263d948f09b4cfc87b0b59955f269f1	transaction	0	NaN	8.58	NaN
36098	6263d948f09b4cfc87b0b59955f269f1	transaction	72	NaN	11.88	NaN
36099	6263d948f09b4cfc87b0b59955f269f1	offer completed	72	2906b810c7d4411798c6938adc9daaa5	NaN	2.0
38745	6263d948f09b4cfc87b0b59955f269f1	transaction	84	NaN	13.58	NaN
47001	6263d948f09b4cfc87b0b59955f269f1	transaction	126	NaN	15.78	NaN
58334	6263d948f09b4cfc87b0b59955f269f1	offer received	168	f19421c1d4aa40978ebb69ca19b0e20d	NaN	NaN
67327	6263d948f09b4cfc87b0b59955f269f1	offer viewed	168	f19421c1d4aa40978ebb69ca19b0e20d	NaN	NaN
73269	6263d948f09b4cfc87b0b59955f269f1	transaction	180	NaN	19.94	NaN
73270	6263d948f09b4cfc87b0b59955f269f1	offer completed	180	f19421c1d4aa40978ebb69ca19b0e20d	NaN	5.0
82188	6263d948f09b4cfc87b0b59955f269f1	transaction	204	NaN	13.16	NaN
86054	6263d948f09b4cfc87b0b59955f269f1	transaction	216	NaN	9.92	NaN
98218	6263d948f09b4cfc87b0b59955f269f1	transaction	264	NaN	15.58	NaN
115999	6263d948f09b4cfc87b0b59955f269f1	offer received	336	2906b810c7d4411798c6938adc9daaa5	NaN	NaN
138305	6263d948f09b4cfc87b0b59955f269f1	transaction	366	NaN	9.28	NaN
155774	6263d948f09b4cfc87b0b59955f269f1	offer received	408	9b98b8c7a33c4b65b9aebfe6a799e6d9	NaN	NaN
190079	6263d948f09b4cfc87b0b59955f269f1	offer viewed	462	9b98b8c7a33c4b65b9aebfe6a799e6d9	NaN	NaN
190080	6263d948f09b4cfc87b0b59955f269f1	transaction	462	NaN	5.82	NaN
190081	6263d948f09b4cfc87b0b59955f269f1	offer completed	462	2906b810c7d4411798c6938adc9daaa5	NaN	2.0
190082	6263d948f09b4cfc87b0b59955f269f1	offer completed	462	9b98b8c7a33c4b65b9aebfe6a799e6d9	NaN	5.0
197491	6263d948f09b4cfc87b0b59955f269f1	transaction	486	NaN	18.59	NaN
199176	6263d948f09b4cfc87b0b59955f269f1	transaction	492	NaN	11.56	NaN
200686	6263d948f09b4cfc87b0b59955f269f1	transaction	498	NaN	10.77	NaN
206681	6263d948f09b4cfc87b0b59955f269f1	offer received	504	9b98b8c7a33c4b65b9aebfe6a799e6d9	NaN	NaN
219860	6263d948f09b4cfc87b0b59955f269f1	transaction	510	NaN	15.98	NaN
219861	6263d948f09b4cfc87b0b59955f269f1	offer completed	510	9b98b8c7a33c4b65b9aebfe6a799e6d9	NaN	5.0
299846	6263d948f09b4cfc87b0b59955f269f1	transaction	684	NaN	8.46	NaN

FIGURE 5: TRANSACTIONS DATA FOR A USER

	start_time	offer_id	end_time	view_time	complete_time
0	0	2906b810c7d4411798c6938adc9daaa5	168	NaN	72
1	168	f19421c1d4aa40978ebb69ca19b0e20d	288	168.0	180
2	336	2906b810c7d4411798c6938adc9daaa5	504	NaN	72
3	408	9b98b8c7a33c4b65b9aebfe6a799e6d9	576	462.0	462

FIGURE 6: OFFER PERIODS FOR A USER

2) Using the above offer periods, we generate contiguous blocks of **offer influence and non-influence periods** *start_time* and *end_time* for the user, going from the experiment *start_time* (T=0) till *end_time* (T=714).

	start_time	end_time	offer_id	person
0	0.0	168.0	no_offer	6263d948f09b4cfc87b0b59955f269f1
1	168.0	180.0	f19421c1d4aa40978ebb69ca19b0e20d	6263d948f09b4cfc87b0b59955f269f1
1	180.0	462.0	no_offer	6263d948f09b4cfc87b0b59955f269f1
3	462.0	462.0	9b98b8c7a33c4b65b9aebfe6a799e6d9	6263d948f09b4cfc87b0b59955f269f1
2	462.0	714.0	no_offer	6263d948f09b4cfc87b0b59955f269f1

FIGURE 7: OFFER INFLUENCE & NON-INFLUENCE PERIODS FOR A USER

3) Left-join with Users Data on *person* to get *gender, age, income,* and date which he *became_member_on*.
4) Left-join with Offers Data on *offer_id* to get offer *reward*
5) Sum the amount transacted by each *person* within *start_time* and *end_time* of each offer influence or non-influence periods.

start_time	end_time	offer_id	person	amount	reward	gender	age	became_member_on	income
0.0	168.0	no_offer	6263d948f09b4cfc87b0b59955f269f1	49.82	NaN	M	41	20161002	48000.0
168.0	180.0	f19421c1d4aa40978ebb69ca19b0e20d	6263d948f09b4cfc87b0b59955f269f1	19.94	5.0	M	41	20161002	48000.0
180.0	462.0	no_offer	6263d948f09b4cfc87b0b59955f269f1	73.70	NaN	M	41	20161002	48000.0
462.0	462.0	9b98b8c7a33c4b65b9aebfe6a799e6d9	6263d948f09b4cfc87b0b59955f269f1	5.82	5.0	M	41	20161002	48000.0
462.0	714.0	no_offer	6263d948f09b4cfc87b0b59955f269f1	71.18	NaN	M	41	20161002	48000.0

FIGURE 8: PREVIEW OF DATA POST TRANSFORMATION

These steps are wrapped in functions defined in **df_transform.py**, and implemented in **Data Preview and Transform.ipynb**. The resulting transformed data has **109828** rows, and is persisted in **inputs_pre.csv**.

Tay Cong Run
24 June 2021

Feature Engineering

Several features that are believed to affect the predicted variable, *avg_transaction_less_reward*, and the predicted variable itself, are engineered from existing data.

- Membership Length *(membership_length)*: The number of days since user *became_member_on* relative to that of the oldest member, is more meaningful than a date input.
- Normalized Membership Length *(norm_membership_length)*: How long a user has been a member **relative to their age** could reflect how much Starbucks is a part of the user's lifestyle, which can contribute to their spending patterns.
- Daily Average of Transaction Less Reward (*avg_transaction_less_reward*): Calculation for the predicted variable was described in our Problem Statement.
- Time Since Previous Transaction (*time_since_prev_transact*): We expect some cooldown time between users' transactions. The length of time since the previous transaction made by the user till the current influence period *start_time* would account for it. It is traced from the Transactions Data.
- User View Rate & Completion Rate *(view_rate, complete_rate)*: The percentage of offers received that are viewed or completed is indicative of a user's behavioural pattern when met with offers.
- User Transaction Aggregates *(transact_mean, transact_std, transact_range)*: From each user's Transactions Data, the different aggregates (mean, standard deviation, range) of transaction amounts give information about the user's spending pattern.

Feature engineering are performed in **Data Exploration.ipynb**, along with detailed code to engineer each feature defined above.

Input Data Analysis & Exploratory Visualizations

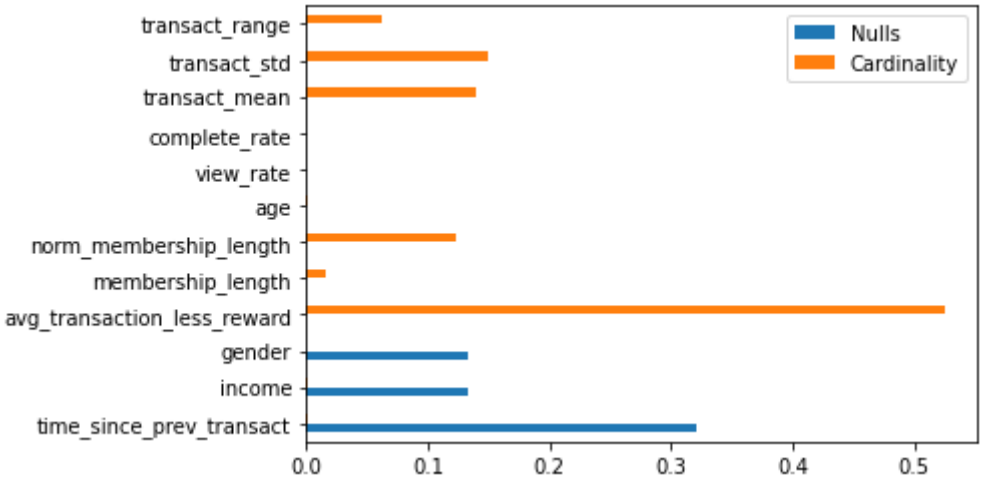


FIGURE 9: NULLS AND CARDINALITY

The input space for prediction consists of 11 independent variables, as shown in Figure 9 les the predicted variable.

There are many inherent duplicates in our demographic variables (age, gender, membership length, income) and *view_rate* and *complete_rate*, as we merged 109828 influence periods on a Profile Data of **17000** members.

We identify 14860 influence periods corresponding to 2085 users with null *gender* and *income* values, and nonsensical *age* of 118. There are little insights we can extract from these rows. Attempting to include these rows would pose a problem given our limited demographic features. Hence, we will drop these rows.

More than 30% of data points have null *time_since_prev_transact* values.

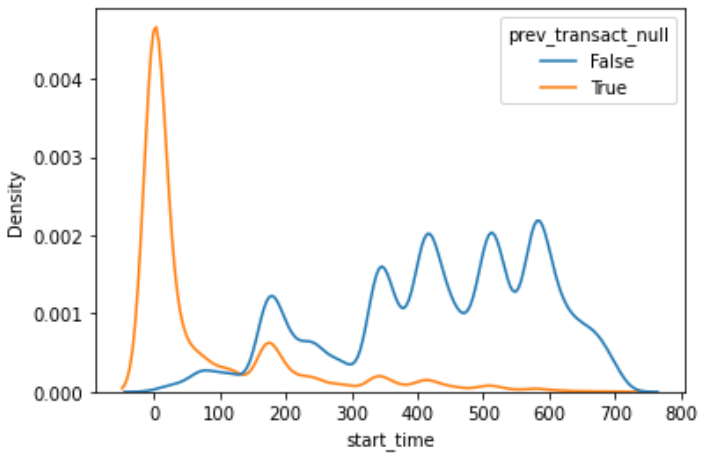


FIGURE 10: START TIMES OF INFLUENCE PERIODS WITH & WITHOUT PREVIOUS TRANSACTIONS

From Figure 10, most null *time_since_prev_transact* occurs at influence periods towards the start of the experiment (a case of cold start). The large number of missing values makes it imperative for us to impute these nulls with an appropriate value, keeping in mind that due to experiment design, *time_since_prev_transact* is inherently capped at 714.

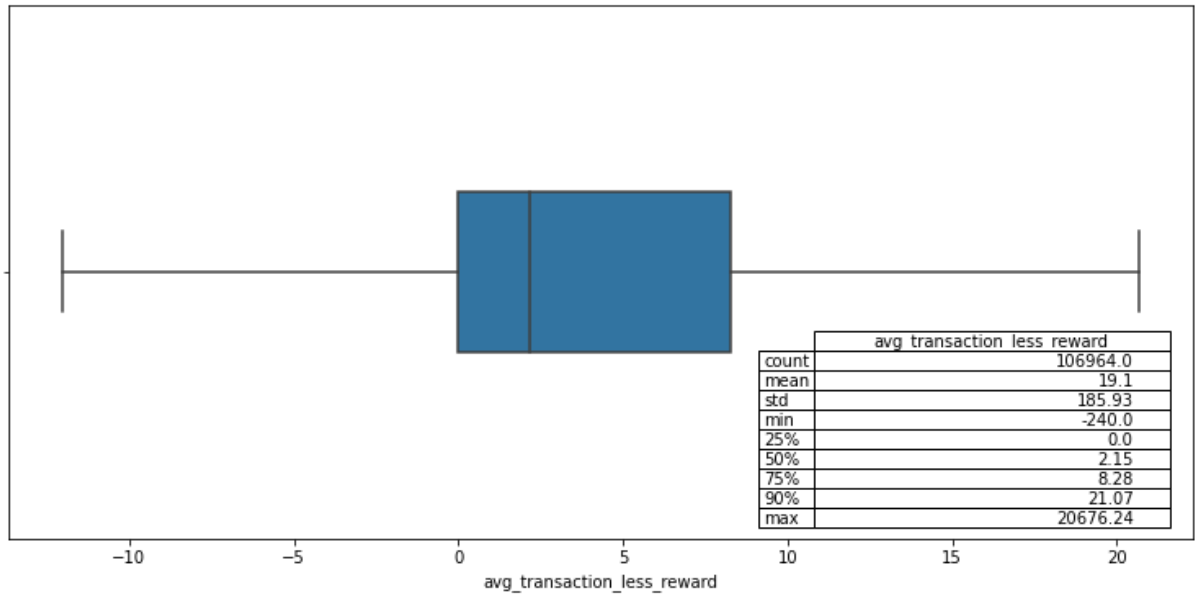


FIGURE 11: BOXPLOT OF PREDICTED VARIABLE

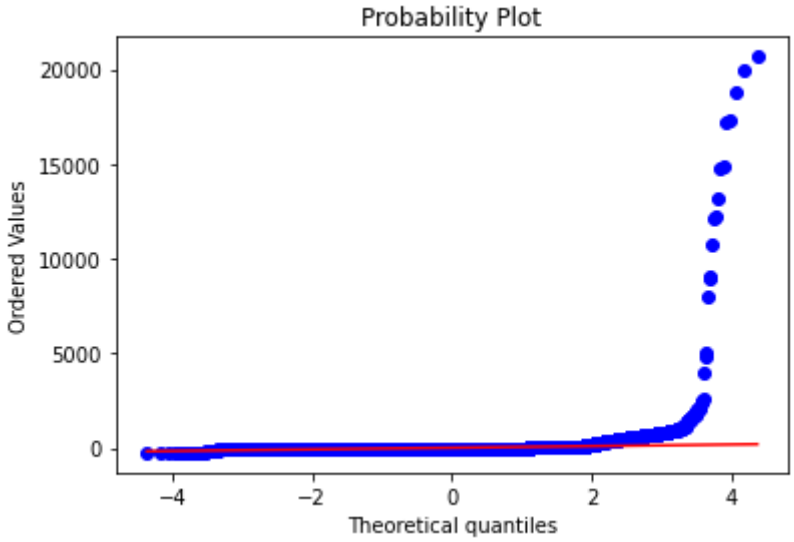


FIGURE 12: PROBABILITY PLOT OF PREDICTED VARIABLE AGAINST NORMAL DISTRIBUTION

Figure 11 reveals that there are outliers bearing absurdly high positive values (a maximum of $20676.24 per day when the 90th percentile is a mere $21.07 per day). The same right skewedness is affirmed by Figure 12. In fact, these outliers consist of influence periods spanning less than a day, with aggregated transactions ranging from just $17 to $1015. Even though they represent good offer receptions, the predicted variable is heavily in need of outlier handling, normalisation and scaling.

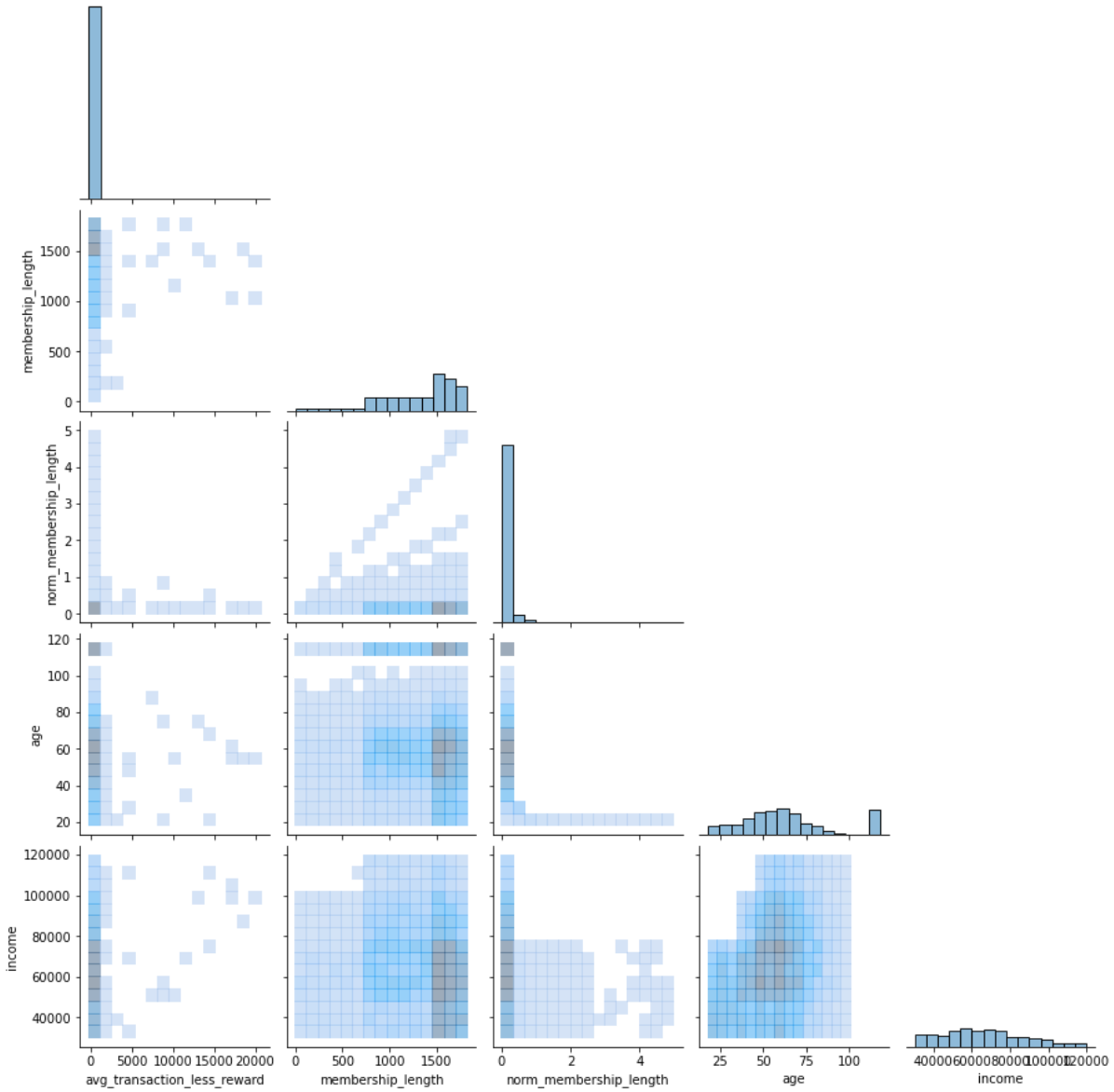


FIGURE 13: DENSITY PLOT BETWEEN PREDICTED AND DEMOGRAPHIC VARIABLES

We can see from Figure 13 that:

- There are outliers in *avg_transaction_less_reward*.
- There are oddities in *age* where multiple data points exceed 100 years old.
- A large portion of long-running members are mid-income and between 40 to 60 years old.
- We vaguely observe that larger *avg_transaction_less_reward* tends to come from longer-running members, higher income, and older age.

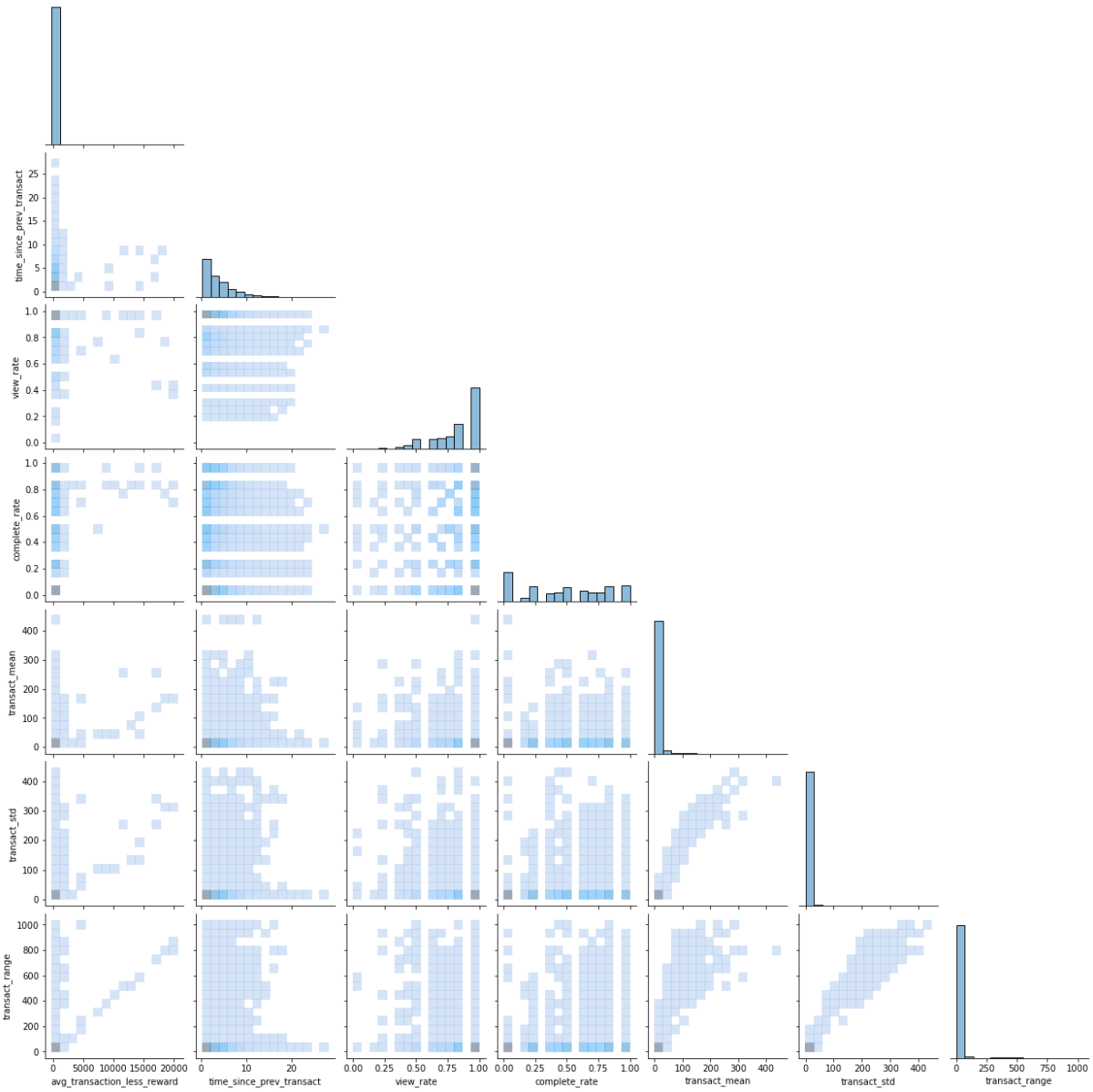


FIGURE 14: DENSITY PLOT BETWEEN PREDICTED AND TRANSACTION VARIABLES

Figure 14 shows *transact_mean, transact_std* and *transact_range* have skewed distributions similar to *avg_transaction_less_reward*. For high values of *avg_transaction_less_reward*, the three transaction aggregates have strong positive correlation with it.

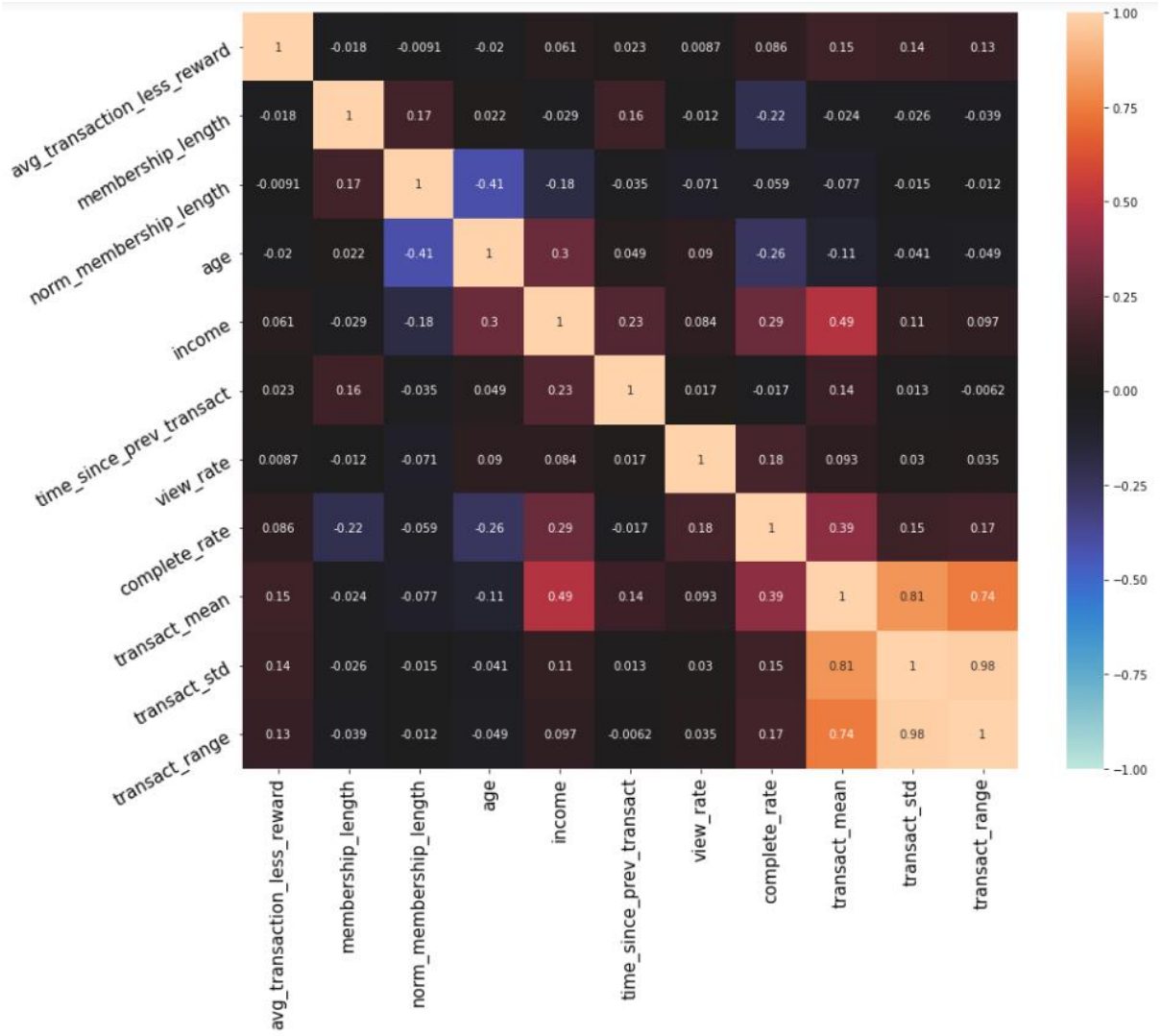


FIGURE 15: CORRELATION HEATMAP IN THE INPUT SPACE

There are few variables which correlates strongly with the predicted variable. This leads us to think that simple linear regression may not fare well in this case, and a more sophisticated machine learning algorithm may fare better in capturing the nuances between the predicted variable and our limited features. In addition, the transaction aggregates display strong correlation among each other. Hence, we may need to perform feature selection to prevent collinearity from impeding our models.

To better grasp how offers are being pushed out and how receptive the users are, for each offer we can count the **number of occurrences** and the **number of user-distinct occurrences** of the three offer events: *offer received, offer viewed,* and *offer completed*.

offer_type	offer_id	offer_received	u_offer_received	offer_viewed	u_offer_viewed	offer_completed	u_offer_completed
bogo	ae264e3637204a6fb9bb56bc8210ddfd	7658	6374	6716	5696	3688.000000	3177.000000
bogo	4d5c57ea9a6940dd891ad53e9dbe8da0	7593	6330	7298	6132	3331.000000	2885.000000
informational	3f207df678b143eea3cee63160fa8bed	7617	6331	4144	3635	nan	nan
bogo	9b98b8c7a33c4b65b9aebfe6a799e6d9	7677	6355	4171	3658	4354.000000	3784.000000
discount	0b1e1539f2cc45b7b9fa7c272da2e1d7	7668	6374	2663	2400	3420.000000	2978.000000
discount	2298d6c36e964ae4a3e7e9706d1fb8c2	7646	6325	7337	6122	5156.000000	4421.000000
discount	fafdcd668e3743c1bb461111dcafc2a4	7597	6332	7327	6150	5317.000000	4530.000000
informational	5a8bc65990b245e5a138643cd4eb9837	7618	6320	6687	5674	nan	nan
bogo	f19421c1d4aa40978ebb69ca19b0e20d	7571	6262	7264	6060	4296.000000	3741.000000
discount	2906b810c7d4411798c6938adc9daaa5	7632	6285	4118	3608	4017.000000	3480.000000

FIGURE 13: OFFER EVENTS STATISTICS

From *offer_received*, we can see that the 10 offers are being pushed out uniformly, each in the 7500 - 7700 range. Compare that to *u_offer_received*, its unique counterpart, there are more than 1000 offers that were re-sent to the same users.

To normalize and compare the reception between different offers, we further calculate the **percentage of received offers viewed** and the **percentage of viewed offers completed**, for each offer. Higher percentages indicate that users are more receptive to the offers. However, as users may complete an offer without viewing them, an overly high percentage of viewed offers completed reveals that the offer might have been redundant, i.e. users would have completed the offers anyway.

	reward	channels	difficulty	duration	offer_type	offer_id	view_rate	complete_rate
0	10	['email', 'mobile', 'social']	10	7	bogo	ae264e3637204a6fb9bb56bc8210ddfd	0.876991	0.549136
1	10	['web', 'email', 'mobile', 'social']	10	5	bogo	4d5c57ea9a6940dd891ad53e9dbe8da0	0.961148	0.456426
2	0	['web', 'email', 'mobile']	0	4	informational	3f207df678b143eea3cee63160fa8bed	0.544046	nan
3	5	['web', 'email', 'mobile']	5	7	bogo	9b98b8c7a33c4b65b9aebfe6a799e6d9	0.543311	1.043874
4	5	['web', 'email']	20	10	discount	0b1e1539f2cc45b7b9fa7c272da2e1d7	0.347287	1.284266
5	3	['web', 'email', 'mobile', 'social']	7	7	discount	2298d6c36e964ae4a3e7e9706d1fb8c2	0.959587	0.702740
6	2	['web', 'email', 'mobile', 'social']	10	10	discount	fafdcd668e3743c1bb461111dcafc2a4	0.964460	0.725672
7	0	['email', 'mobile', 'social']	0	3	informational	5a8bc65990b245e5a138643cd4eb9837	0.877789	nan
8	5	['web', 'email', 'mobile', 'social']	5	5	bogo	f19421c1d4aa40978ebb69ca19b0e20d	0.959451	0.591410
9	2	['web', 'email', 'mobile']	10	7	discount	2906b810c7d4411798c6938adc9daaa5	0.539570	0.975474

FIGURE 14: OFFERS VIEW RATE & COMPLETION RATE

Offer indices 2, 3, and 9 are viewed only around half of the time. These happen to be offers that does not include social media as a channel, which speaks volume about the marketing value on social medias.

Offer index 4 fared even worse, being viewed less than half the time. It further omitted mobile platform as a channel. It could also be due to it having a high difficulty but low reward.

Aside from informational offers, offer indices 3, 4 and 9 have dangerously high completion rate, indicating they might have been poor offer choices to push to these users. We can already see that there is much room for improvements when it comes to user receptiveness and offer recommendation rules.

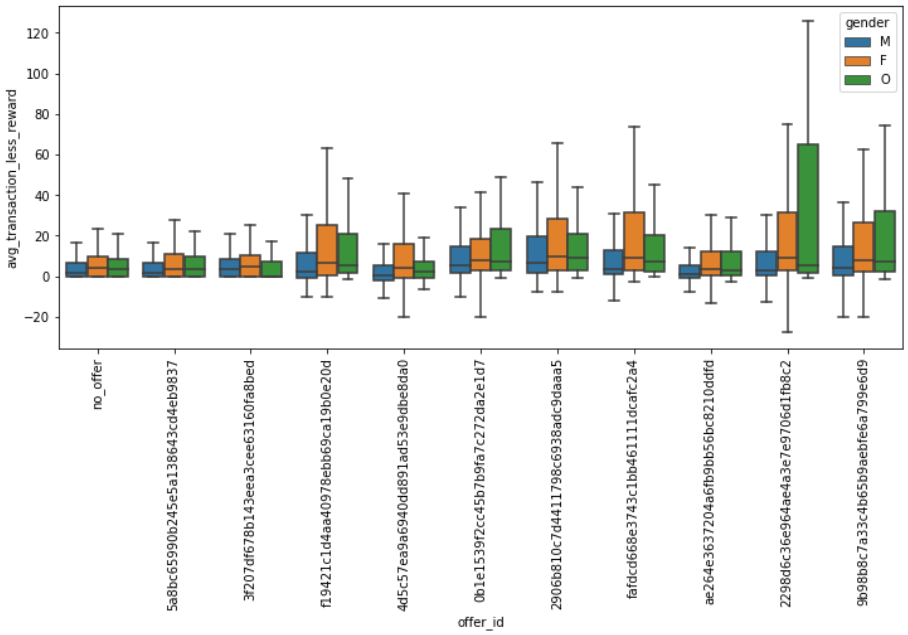


FIGURE 15: GROUPED BOXPLOTS OF PREDICTED VARIABLE BY OFFER AND GENDER

Figure 13 suggests there is no offer that vastly increases the mean *avg_transaction_less_reward*. Instead, offers increases the spread of the predicted variable. While most offers increase *avg_transaction_less_reward* for some people, some offers brought it into the negatives. These are our identified areas of improvement on the choice of offers pushed out to users.

Input data analysis and exploratory visualizations were performed in **Data Exploration.ipynb**.

Data Pre-processing

In addition to Data Transformation, further pre-processing is required to deal with the earlier highlighted issues of missing values and outliers in predicted variable, as well as to massage the data that can into a form that can be well-received by our model during model training. The steps are as follows:

1) Drop irrelevant columns that are not outlined in our Input Space.
2) Drop **14860** rows with null *gender* and *income* and an arbitrary *age* of 118.
3) Split data into train and test sets with a **9:1** ratio, and each into **X** (predictors) and **y** (predicted) sets. Train and test sets have **83488** rows and **9277** rows respectively.
4) Impute null *time_since_prev_transact* with randomly sampled values. This should be sufficient as its distribution already resembles an exponential distribution, a common distribution for time between random event occurrences. Even if cold start is a problem now, this same imputation step can be reused with future input data that spans a longer period and will automatically be more accurate.
5) Discretise *view_rate* and *complete_rate* into 10 equal width bins, owing to their low cardinality.
6) Encode *view_rate* and *complete_rate* with the mean value of *avg_transaction_less_reward* for each discrete bin, a good starting point for the model for fitting within each bin.
7) One-hot encode *offer_id* and *gender*. This captures the variability in our predicted variable for every categorical permutation. For our benchmark linear regression model, it allows easy interpretation of its coefficients.
8) Perform Yeo-Johnson transformation on the following features which have distributions very unlike normal, as seen in Figures 13 and 14:
    - *membership_length*
    - *norm_membership_length*
    - *transact_mean*
    - *transact_std*

- *transact_range*

9) Scale all numerical variables to a range **between 0 and 1** by subtracting its minimum then dividing by its range, if they are not already in that range. Standardisation of numerical variables prevent variables with larger values from overshadowing other variables in model training.
10) Generate polynomial features, **up to degree 3**, using all numerical variables. This captures any interaction effect between these limited features, by generating **284** additional features.
11) Cap outliers of *avg_transaction_less_reward* **below the 5th** and **above the 90th quantile** to the quantile values, respectively. Uneven quantile capping is employed as the predicted variable suffers from right skewed and we can avoid re-choosing arbitrary values when training on a larger input dataset in the future.
12) Perform Yeo-Johnson transformation to curb the skewedness and increase the normality of the predicted variable, same as Step 8.
13) Scale the predicted variable into to a range **between 0 and 1**, same as Step 9.

Data Pre-processing was performed in **Data Cleaning and Model Training.ipynb**. Refer to Annex A for numerical distribution plots after pre-processing.

Feature Selection

Due to the large number of polynomial features generated, we would like to trim the number of features down to **around 50** to speed up model training. To do this, we employ **model-based feature selection**. The basic intended model is first fitted using all features. Then from the fitted model, feature importance is inferred, and we pick the **top 50 most important features**.

We manually ensure that the encoded counterparts of inherent categorical features (i.e. *gender, offer_id*) are **always included** as these are important variabilities.

Benchmark Model

For the benchmark linear regression model, feature importance can be inferred from the **magnitude of its coefficients** in the fitted model.

Machine Learning Model

For the Gradient Boosting regression model, feature importance is derived from **Gini importance**. It is defined as the sum over the number of node splits by a feature across all trees, weighted by the number of samples through the nodes.

Feature selection was performed in **Data Cleaning and Model Training.ipynb**. Refer to Annex B for the list of selected features for both types of model.

Algorithms & Techniques

Our choice of machine learning algorithm is the Gradient Boosting. It uses an **ensemble of weak Decision Tree predictors** to make averaged estimations, which are presumed to be more consistent and reliable in a sense that **errors between weak predictors work to cancel each other out**. This is suitable in the case where individual prediction models fits poorly on the entire dataset but fits nicely on partial datasets.

A basic Gradient Boosting regressor is first fitted using default scikit-learn hyperparameters, then again with tuned hyperparameters.

Benchmark

Models

Before training a sophisticated machine learning model, we will fit a simple **Ordinary Least Square** multi-linear regression model as benchmark for making predictions for our dependant variable. It is a traditional closed-form solution derived from minimising the squared deviations between predicted and actual values of the independent variable.

Heuristics

For this small pool of 10 offers, two benchmarks model to compare against a model-optimised offer at each juncture an offer is to be made to a user would be:

1) Picking a random offers.
2) Picking an unoptimized offer, i.e. the original offer pushed.

We should expect the model-optimised offer to perform better than both benchmarks.

Tay Cong Run
24 June 2021

Model Training

The table below summarizes the important factors from model training.

Model	**Benchmark Model (MLR)**	**Gradient Boosting Regressor (Untuned)**	**Gradient Boosting Regressor (Tuned)**
Feature Selection Time	1.18s	7.41m	7.41m
Hyperparameters	scikit-learn defaults	scikit-learn defaults	Learning rate 0.1 Max depth 5 N estimators 125
Training Time	341ms	59.91s	25m 37s

Model Training was performed in **Data Cleaning and Model Training.ipynb**. The benchmark model, untuned and tuned Gradient Boosting Regressor models are persisted in **model/**.

Refinement

Several iterative steps were taken to improve the model prediction score, which was **~0.2** initially. They include:

1) Engineering additional features (*norm_membership_length, time_since_prev_transact, view_rate, complete_rate, transact_mean, transact_std, transact_range*) that were not immediately available from the input data but have better correlation with the predicted variable.
2) Normalization of less-normal-like numerical variables via Yeo-Johnson Transformation.
3) Inclusion of polynomial features.
4) Hyperparameter tuning to find the best combination of the following hyperparameters within a given parameter space:
    - **n_estimators**: The number of underlying weak predictors. It is robust to over-fitting so a larger number often results in better performance.
    - **max_depth**: The maximum depth of each individual decision tree regressor. It also serves as a limiter for the number of leaf nodes and split nodes for each tree. A greater depth captures higher orders of variable interactions.
    - **learning_rate**: The scale for each step length in the gradient descent procedure when fitting each tree. Smaller values tend to result in better test error, but could be over-sensitive to neighbourhood minimums. Larger values, on the other hand, converges faster but may not result in the lowest possible test error.

5) Careful selection of polynomial features to trim training time and allow for larger parameter space during hyperparameter tuning.

Hyperparameter Tuning was performed in **Data Cleaning and Model Training.ipynb**. Refer to Annex C for hyperparameter search results.

Model Evaluation & Validation

The table below summarizes the various metrics for each model and each train and test set:

Model	**Benchmark Model (MLR)**	**Gradient Boosting Regressor (Untuned)**	**Gradient Boosting Regressor (Tuned)**
R-squared Coefficient (Train / Test)	0.41 / 0.40	0.44 / 0.43 (+0.03 / +0.03)	0.49 / 0.45 (+0.08 / +0.05)
Mean Squared Error (Train / Test)	0.05 / 0.05	0.05 / 0.05 (-0 / -0)	0.04 / 0.05 (-0.01 / -0)
Mean Underestimation (Train / Test)	6.28 / 6.06	6.40 / 6.45 (+0.22 / +0.39)	6.12 / 6.28 (-0.16 / +0.22)
Mean Overestimation (Train / Test)	2.40 / 2.74	2.22 / 2.25 (+0.18 / -0.49)	2.16 / 2.27 (-0.24 / -0.47)
Underestimates outside Tolerance (Train / Test)	23% / 21%	23% / 23% (+0% / -2%)	22% / 23% (+0% / +2%)
Overestimates outside Tolerance (Train / Test)	7% / 9%	6% / 6% (-1% / -3%)	6% / 6% (-1% / -3%)

Based solely on R-squared and Mean Squared Error, there are absolute improvements going from the benchmark model to the untuned Gradient Boosting model to the tuned Gradient Boosting model, without much issue of over-fitting to the training set. However, **45%** goodness of fit is slightly lacking for use in a business scenario. Refer to Annex D for goodness-of-fit plots.

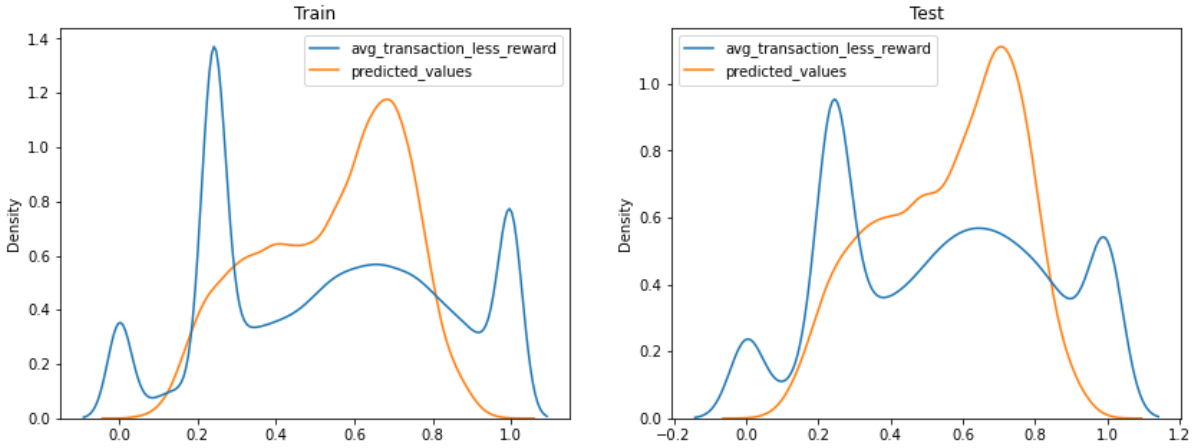


FIGURE 16: DISTRIBUTION PLOT OF PREDICTED AND ACTUAL VALUES (LINEAR REGRESSION)

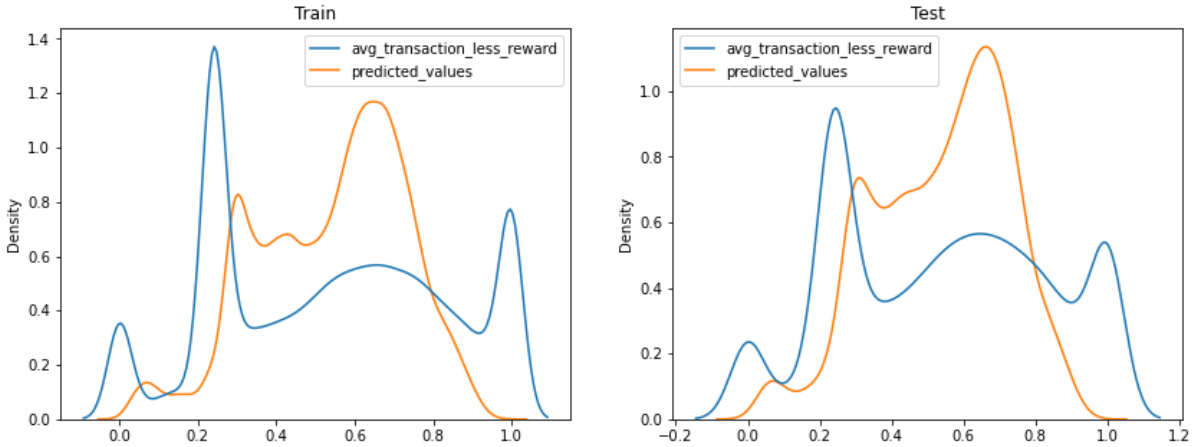


FIGURE 17: DISTRIBUTION PLOT OF PREDICTED AND ACTUAL VALUES (UNTUNED GRADIENT BOOSTING REGRESSOR)

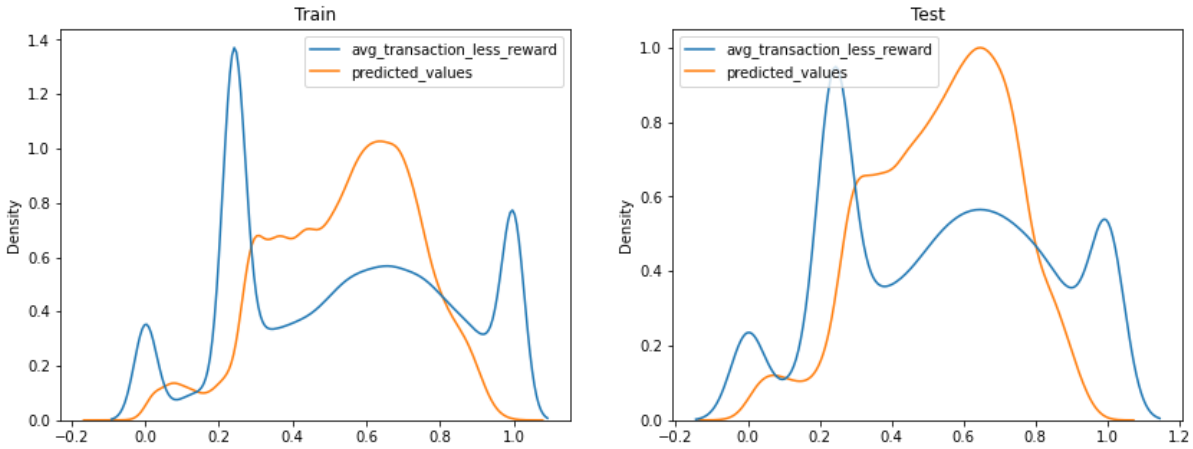


FIGURE 18: DISTRIBUTION PLOT OF PREDICTED AND ACTUAL VALUES (TUNED GRADIENT BOOSTING REGRESSOR)

Comparing Figures 16 through 18, we can observe that prediction performance suffers at end-tails (a typical regression issue) and at values near 0.2. Employing the Gradient Boosting algorithm captures the left-tail and values near 0.2 slightly better. Poor performance near the right-tail could be attributed to the strict capping of positive outliers.

To continue using our tuned model to make predictions and rank offers to push to users, we do so while staying informed about the **odds of making over- or under-estimates** and the **extent of them**. We also set up a tolerance band of +/- $5 around the actual *avg_transaction_less_reward* to get a feel of how probable it is for our model to make estimates within this margin of error. For our tuned model, our **estimates are within $5 of the actual value 71% of the time**, **overestimates exceed by $5 6% of the time, and underestimates exceed by -$5 23% of the time**.

Model Application

For every offer influence period j (total of $J = 42577$), use the model on the inputs to predict $\hat{Y}_{best,j}$ (the offer that results in the largest increase in **daily average of transaction less offer reward**), $\hat{Y}_{random,j}$ (a random offer) and $\hat{Y}_{offer\_id,j}$ (the original offer made). Then, subtract $\hat{Y}_{no\_offer,j}$ to get the predicted change in **daily average of transaction less offer reward**.

	change	random_change	best_change
count	42577.000000	42577.000000	42577.000000
mean	1.795946	1.491949	4.286205
std	2.886072	2.648243	3.634763
min	-7.204703	-6.930093	0.000000
25%	-0.346252	-0.220490	0.881534
50%	0.844794	0.411536	3.643736
75%	3.267408	2.649827	7.228889
max	19.323456	23.211087	23.211087

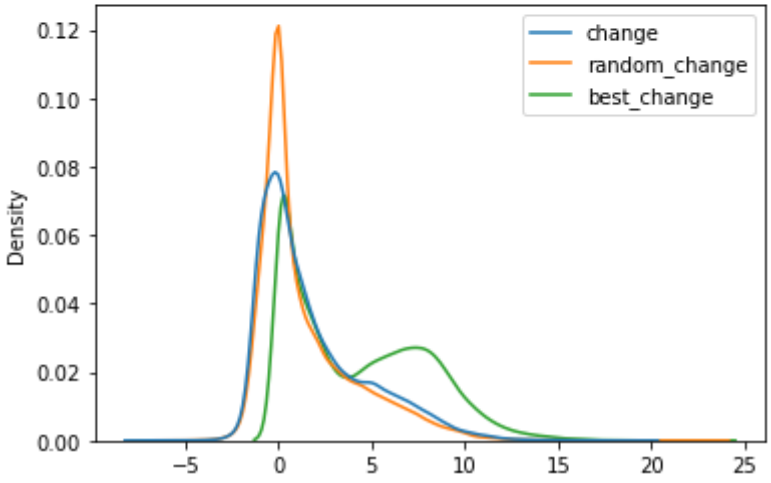


FIGURE 19: DISTRIBUTION & STATISTICS OF PREDICTED CHANGES IN DAILY AVERAGE OF TRANSACTION LESS OFFER REWARD

Figure 19 shows that according to our model, pushing random offers leads to the worst changes in **daily average of transaction less offer reward**, followed by the actual offers pushed, then the model-suggested offer. On average, the model predicts that each optimal offer recommendation can further boost **daily average of transaction less offer reward** by around $2.50 per offer influence period.

```
3f207df678b143eea3cee63160fa8bed    211
5a8bc65990b245e5a138643cd4eb9837    193
4d5c57ea9a6940dd891ad53e9dbe8da0    179
f19421c1d4aa40978ebb69ca19b0e20d    168
2298d6c36e964ae4a3e7e9706d1fb8c2    158
fafdcd668e3743c1bb461111dcafc2a4    142
9b98b8c7a33c4b65b9aebfe6a799e6d9    122
0b1e1539f2cc45b7b9fa7c272da2e1d7    111
ae264e3637204a6fb9bb56bc8210ddfd    107
2906b810c7d4411798c6938adc9daaa5    106
Name: offer_id, dtype: int64
```

	change	random_change	best_change
count	1497.000000	1497.000000	1497.0
mean	-0.912700	-0.855509	0.0
std	0.666230	0.663936	0.0
min	-6.852243	-6.046967	0.0
25%	-1.297009	-1.239731	0.0
50%	-0.797252	-0.766075	0.0
75%	-0.410346	-0.379580	0.0
max	-0.002209	0.000000	0.0

FIGURE 20: COUNTS OF AND LOSSES FROM REDUNDANT OFFERS MADE

Our model can also point out which offers made were **redundant**, as in Figure 20. On average, each redundant offer made would have cost Starbucks $0.91 in costs (in the form of rewards).

Project Summary

We were able to train a predictive model that can serve as a heuristic for recommending offers to push to users by predicting a meaningfully formulated variable, **daily average of transaction less offer reward**. Although model training requires available input data to be massaged into the right granularity, the input space of the model takes in easily available and calculable Users Data and outputs easily interpretable number that can rank offers based on business needs.

Using a machine learning algorithm with hyperparameters tuning improved the goodness-of-fit and shrunk the mean overestimates and underestimates, compared to the benchmark. Intuitive feature engineering, imputation, normalization, scaling and selection also served to improve the model performance. Nevertheless, the goodness-of-fit still leaves much to be desired. More features (e.g. location, race) and more data (beyond 1 month) could provide additional variability to predict the values our model performed poorly in and resolve cold start issues in data.

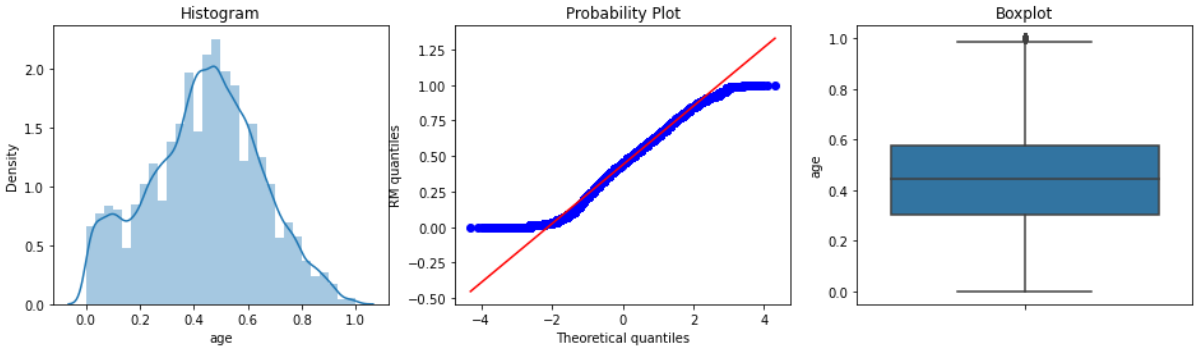
Although the model is not business-ready, it was integrated into what could be a framework to make optimal offer recommendations. At this stage, it should be used with care, bearing in mind the prediction tolerances and what underestimates and overestimates to expect.

Tay Cong Run
24 June 2021

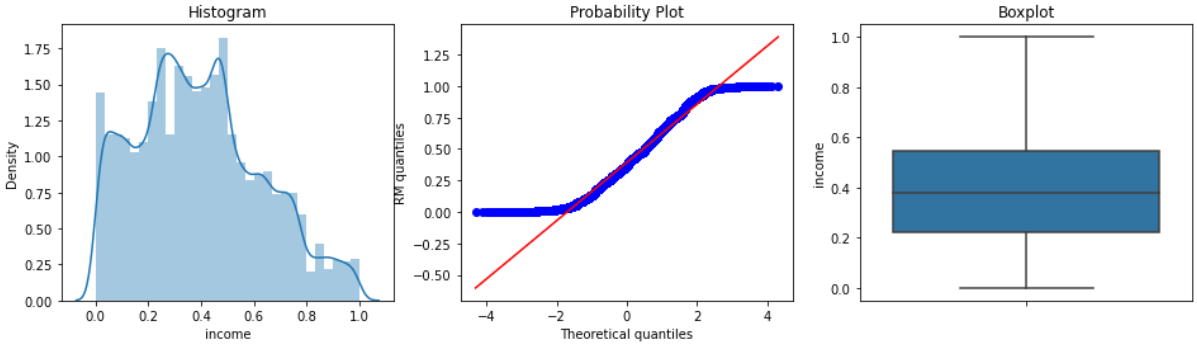
Annexes

Annex A: Diagnostic Distribution Plots of Continuous Numerical Variables after Pre-processing

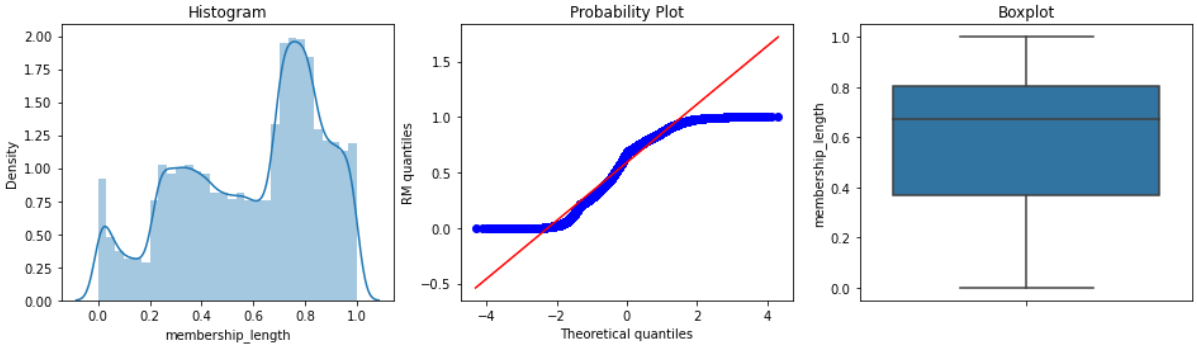
Age



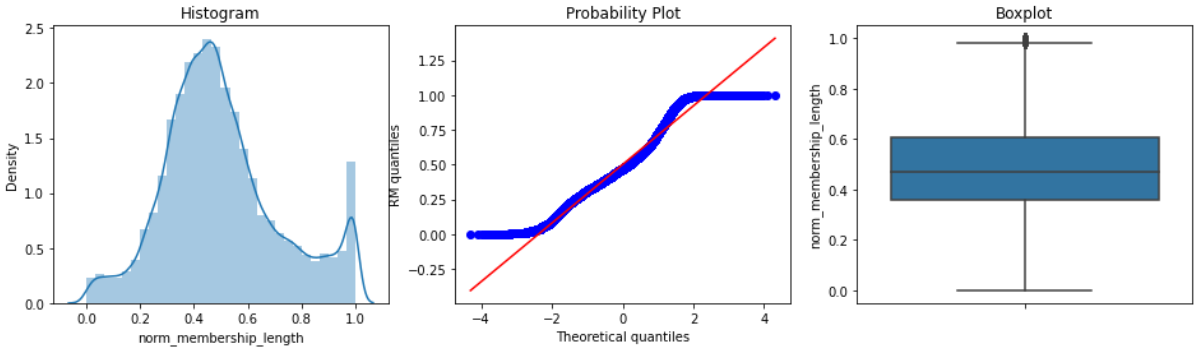
Income



Membership Length

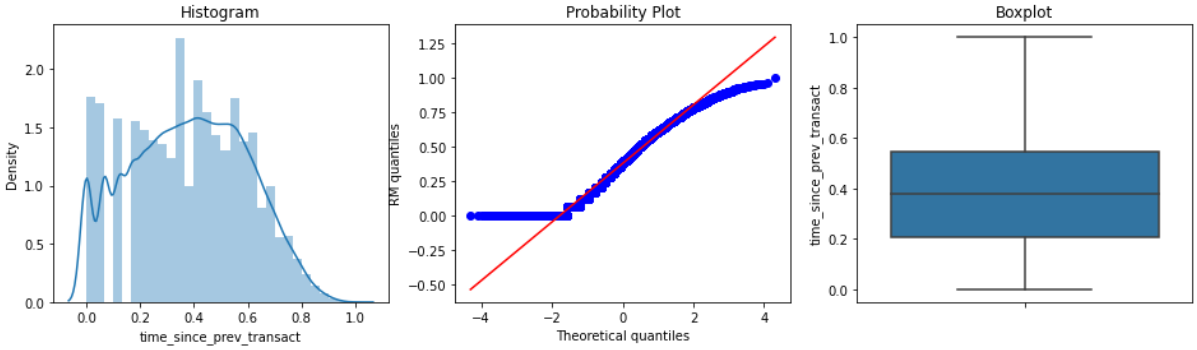


Normalised Membership Length

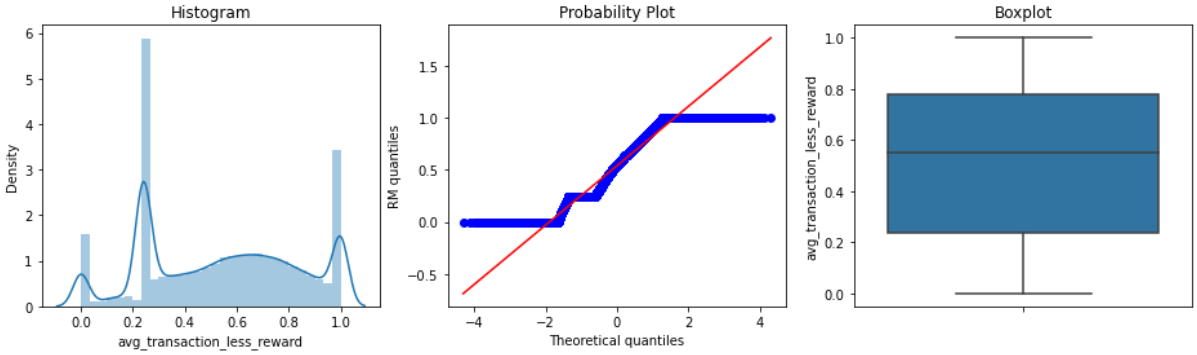


Tay Cong Run
24 June 2021

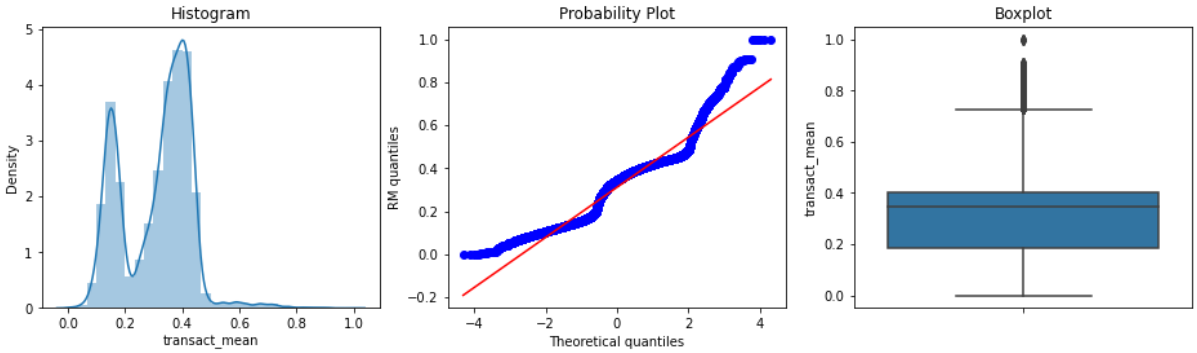
Time Since Previous Transaction



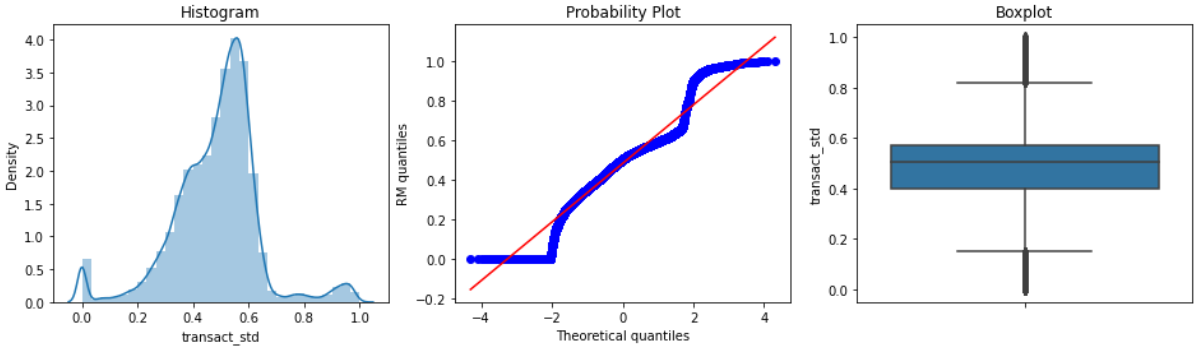
Daily Average of Transaction Amount Less Reward



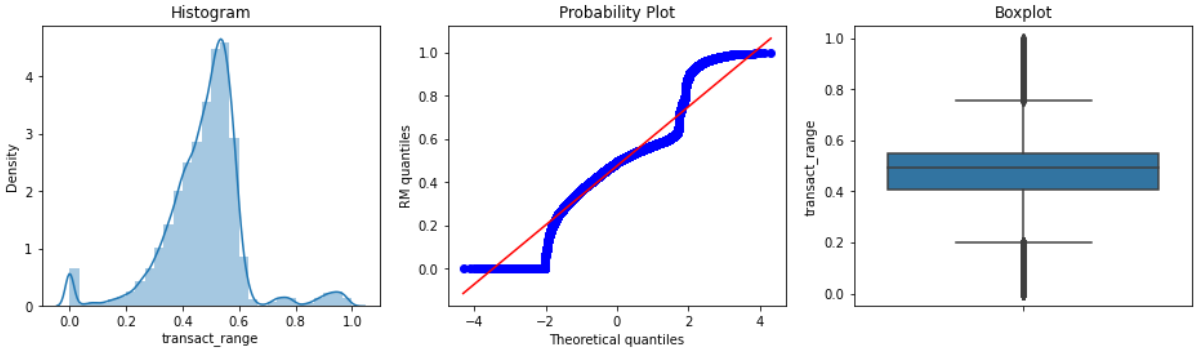
Transaction Mean



Transaction Standard Deviation



Transaction Range



Annex B: Columns Selected during Model-based Feature Selection

Linear Regression (62 features)

```
Selected columns:
[0, 1, 'norm_membership_length', 'complete_rate', 2, 'age', 3, 4, 5, 6,
 7, 8, 9, 260, 261, 'transact_mean', 263, 264, 272, 273, 'time_since_pr
ev_transact', 274, 276, 'income', 277, 278, 279, 282, 283, 284, 'offer_
id_3f207df678b143eea3cee63160fa8bed', 'transact_std', 'offer_id_ae264e3
637204a6fb9bb56bc8210ddfd', 'membership_length', 'offer_id_9b98b8c7a33c
4b65b9aebfe6a799e6d9', 262, 'offer_id_5a8bc65990b245e5a138643cd4eb9837
', 50, 63, 198, 199, 'view_rate', 'offer_id_2298d6c36e964ae4a3e7e9706d1
fb8c2', 'gender_M', 'offer_id_2906b810c7d4411798c6938adc9daaa5', 'offer
_id_f19421c1d4aa40978ebb69ca19b0e20d', 'offer_id_no_offer', 'offer_id_f
afdcd668e3743c1bb461111dcafc2a4', 224, 97, 225, 226, 227, 'offer_id_4d5
c57ea9a6940dd891ad53e9dbe8da0', 'gender_F', 115, 116, 117, 118, 250, 25
1, 'transact_range']
```

Gradient Boosting Regressor (62 features)

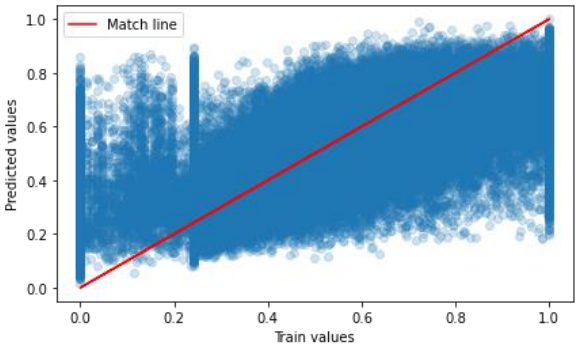
```
Selected columns: [0, 1, 'norm_membership_length', 'complete_rate', 2,
'age', 3, 4, 5, 6, 7, 8, 9, 260, 261, 'transact_mean', 263, 264, 272, 2
73, 'time_since_prev_transact', 274, 276, 'income', 277, 278, 279, 282,
 283, 284, 'offer_id_3f207df678b143eea3cee63160fa8bed', 'transact_std',
 'offer_id_ae264e3637204a6fb9bb56bc8210ddfd', 'membership_length', 'off
er_id_9b98b8c7a33c4b65b9aebfe6a799e6d9', 262, 'offer_id_5a8bc65990b245e
5a138643cd4eb9837', 50, 63, 198, 199, 'view_rate', 'offer_id_2298d6c36e
964ae4a3e7e9706d1fb8c2', 'gender_M', 'offer_id_2906b810c7d4411798c6938a
dc9daaa5', 'offer_id_f19421c1d4aa40978ebb69ca19b0e20d', 'offer_id_no_of
fer', 'offer_id_fafdcd668e3743c1bb461111dcafc2a4', 224, 97, 225, 226, 2
27, 'offer_id_4d5c57ea9a6940dd891ad53e9dbe8da0', 'gender_F', 115, 116,
117, 118, 250, 251, 'transact_range']
```

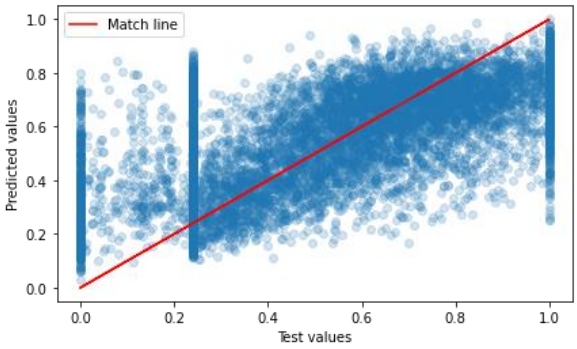
Annex C: Hyperparameter Search

```
r2 scores:
0.446 (+/-0.003) for {'learning_rate': 0.1, 'max_depth': 5, 'n_estimators': 125}
0.446 (+/-0.003) for {'learning_rate': 0.1, 'max_depth': 5, 'n_estimators': 100}
0.445 (+/-0.003) for {'learning_rate': 0.1, 'max_depth': 5, 'n_estimators': 75}
0.437 (+/-0.003) for {'learning_rate': 0.1, 'max_depth': 8, 'n_estimators': 75}
0.435 (+/-0.003) for {'learning_rate': 0.1, 'max_depth': 3, 'n_estimators': 125}
0.434 (+/-0.003) for {'learning_rate': 0.1, 'max_depth': 8, 'n_estimators': 100}
0.433 (+/-0.003) for {'learning_rate': 0.1, 'max_depth': 3, 'n_estimators': 100}
0.430 (+/-0.003) for {'learning_rate': 0.1, 'max_depth': 8, 'n_estimators': 125}
0.429 (+/-0.002) for {'learning_rate': 0.1, 'max_depth': 3, 'n_estimators': 75}
0.397 (+/-0.003) for {'learning_rate': 0.01, 'max_depth': 8, 'n_estimators': 125}
0.377 (+/-0.002) for {'learning_rate': 0.01, 'max_depth': 5, 'n_estimators': 125}
0.371 (+/-0.003) for {'learning_rate': 0.01, 'max_depth': 8, 'n_estimators': 100}
0.350 (+/-0.002) for {'learning_rate': 0.01, 'max_depth': 5, 'n_estimators': 100}
0.345 (+/-0.002) for {'learning_rate': 0.01, 'max_depth': 3, 'n_estimators': 125}
0.331 (+/-0.003) for {'learning_rate': 0.01, 'max_depth': 8, 'n_estimators': 75}
0.319 (+/-0.002) for {'learning_rate': 0.01, 'max_depth': 3, 'n_estimators': 100}
0.310 (+/-0.002) for {'learning_rate': 0.01, 'max_depth': 5, 'n_estimators': 75}
0.280 (+/-0.002) for {'learning_rate': 0.01, 'max_depth': 3, 'n_estimators': 75}
0.093 (+/-0.001) for {'learning_rate': 0.001, 'max_depth': 8, 'n_estimators': 125}
0.085 (+/-0.001) for {'learning_rate': 0.001, 'max_depth': 5, 'n_estimators': 125}
0.076 (+/-0.001) for {'learning_rate': 0.001, 'max_depth': 8, 'n_estimators': 100}
0.076 (+/-0.000) for {'learning_rate': 0.001, 'max_depth': 3, 'n_estimators': 125}
0.070 (+/-0.000) for {'learning_rate': 0.001, 'max_depth': 5, 'n_estimators': 100}
0.062 (+/-0.000) for {'learning_rate': 0.001, 'max_depth': 3, 'n_estimators': 100}
0.058 (+/-0.000) for {'learning_rate': 0.001, 'max_depth': 8, 'n_estimators': 75}
0.053 (+/-0.000) for {'learning_rate': 0.001, 'max_depth': 5, 'n_estimators': 75}
0.048 (+/-0.000) for {'learning_rate': 0.001, 'max_depth': 3, 'n_estimators': 75}
```

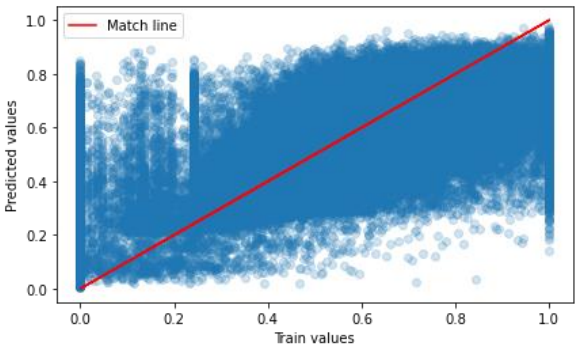
Annex D: Goodness-of-fit Plots

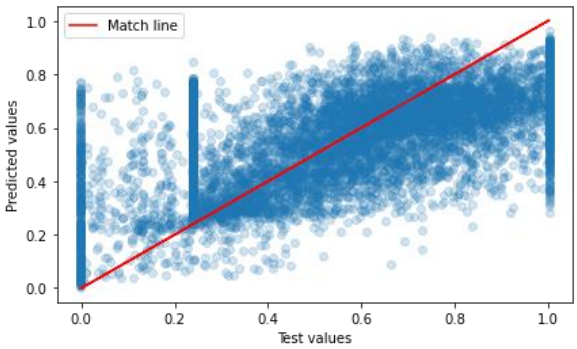
Linear Regression





Untuned Gradient Boosting Regressor





Tay Cong Run
24 June 2021

Tuned Gradient Boosting Regressor