Capstone Project – Optimizing App Offers with Starbucks

Udacity Data Scientist Nanodegree

Yu Tao May 1, 2021

Definition

Project overview:

Understanding the customer behavior on the Starbucks rewards mobile app, specifically, the customer response to different offers, will help the company provide better customer experiences and increase the revenue. In this project, a machine learning classification model was developed to predict whether a customer will respond to certain offers, based on the customer's transactions and demographic information.

The data set contains simulated data that mimics customer behavior on the Starbucks rewards mobile app. Starbucks sends out offers to customers of the mobile app. An offer can be merely an advertisement for a drink (informational) or an actual offer such as a discount or BOGO (buy one get one free). Every offer has a validity period before the offer expires.

The data is contained in three json files:

- portfolio: containing offer ids and meta data about each offer (duration, type, etc.).
- profile: demographic data for each customer.
- transcript: records for transactions, offers received, offers viewed, and offers completed.

Problem statement:

If a customer completed an offer within the validity period, the customer responded to the offer successfully. To build a classification model and predict the customer response to offers, the key step is to label the data based on the customer transactions, separating successful responses (label '1') from unsuccessful ones (label '0'). This procedure could be tricky since certain customers might make a purchase without having received an offer or seen an offer. Besides, some demographic groups will make purchases even if they don't receive an offer. Different groups need to be assessed before we assign the labels. Afterwards, we will select features from the customer demographic and the offer type information so that the machine learning model can learn from it.

Metrics:

To evaluate the performance of a machine learning classification model, we will compare the train/test accuracy. Accuracy is defined as (true positives + true negatives)/dataset size. In general, a higher test

accuracy indicates a better classification model. However, accuracy scores are not always informative, especially when the data labels are imbalanced. Therefore, the f1-scores (precision, recall) will also be evaluated, where false positives and false negatives are also included.

Analysis

Data exploration:

A brief overview of the three json file were provided below. The 'portfolio' file contains offer ids and meta data about each offer (duration, type, etc.). 'Channels' specifies platforms where the offers were launched, 'difficulty' and 'reward' indicate the minimum amount in dollars needed to spend in order to use the offer and the reward to be expected from the offer, whereas the 'duration' shows the validity period of each offer. Dummy variables were assigned to the 'channels' column so we can use channels as part of the features to build our model.

Table I: The 'portfolio' file, containing offer ids and meta data about each offer (duration, type, etc.).

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

The 'profile' file shows demographic data of each customer, including the age, membership time, gender, id and income. There are missing values in the gender and income columns, it is also noted that ages like 118 are present. Further explorations to address these issues will be shown in the exploratory visualization section. I converted the entries in the 'became_member_on' from string format into datetime and extract the corresponding membership year, month and day information.

Table II: the 'profile' file that shows demographic data for each customer.

	age	became_member_on	gender	id	income	member_year	member_month	member_day
0	118	2017-02-12	None	68be06ca386d4c31939f3a4f0e3dd783	NaN	2017	2	Sunday
1	55	2017-07-15	F	0610b486422d4921ae7d2bf64640c50b	112000.0	2017	7	Saturday
2	118	2018-07-12	None	38fe809add3b4fcf9315a9694bb96ff5	NaN	2018	7	Thursday
3	75	2017-05-09	F	78afa995795e4d85b5d9ceeca43f5fef	100000.0	2017	5	Tuesday
4	118	2017-08-04	None	a03223e636434f42ac4c3df47e8bac43	NaN	2017	8	Friday

The ‘transcript’ file includes records for transactions, offers received, offers viewed, and offers completed from each customer. These four customer records are indicated in the ‘event’ column. The ‘person’ and ‘time’ columns specify the person id (same as the ‘id’ column in the profile data set) and time in hour (starting from 0) associated with each record. In the ‘value’ column, if the record is either offers received, offers viewed or offers completed, it will be linked to the ‘offer_id’. Offers completed will also have ‘reward’ shown from the purchases. If the record is from transaction, the event will provide the purchase ‘amount’ in dollars.

Table III: the ‘transcript’ file with records for transactions, offers received, viewed, and completed.

	event	person	time	value
0	offer received	78afa995795e4d85b5d9ceeca43f5fef	0	{'offer id': '9b98b8c7a33c4b65b9aebfe6a799e6d9'}
1	offer received	a03223e636434f42ac4c3df47e8bac43	0	{'offer id': '0b1e1539f2cc45b7b9fa7c272da2e1d7'}
2	offer received	e2127556f4f64592b11af22de27a7932	0	{'offer id': '2906b810c7d4411798c6938adc9daaa5'}
3	offer received	8ec6ce2a7e7949b1bf142def7d0e0586	0	{'offer id': 'fafdcd668e3743c1bb461111dcafc2a4'}
4	offer received	68617ca6246f4fbc85e91a2a49552598	0	{'offer id': '4d5c57ea9a6940dd891ad53e9dbe8da0'}

Exploratory visualization:

In ‘profile’ file, there are 17000 unique person ids, and 2175 missing values from both the ‘gender’ and ‘income’ columns. From the age histogram, there is an anomaly at the age of 118. If we check the missing values from a subset of ‘profile’ where we specify the age to be 118, exactly 2175 missing values in ‘gender’ and ‘income’ columns were found. The hypothesis is, when customers register in the Starbuck app, their age, gender and income information are not mandatory to enter, unlike the ‘gender’ or ‘income’ being assigned to a ‘NaN’ value, the age will be default set to 118.

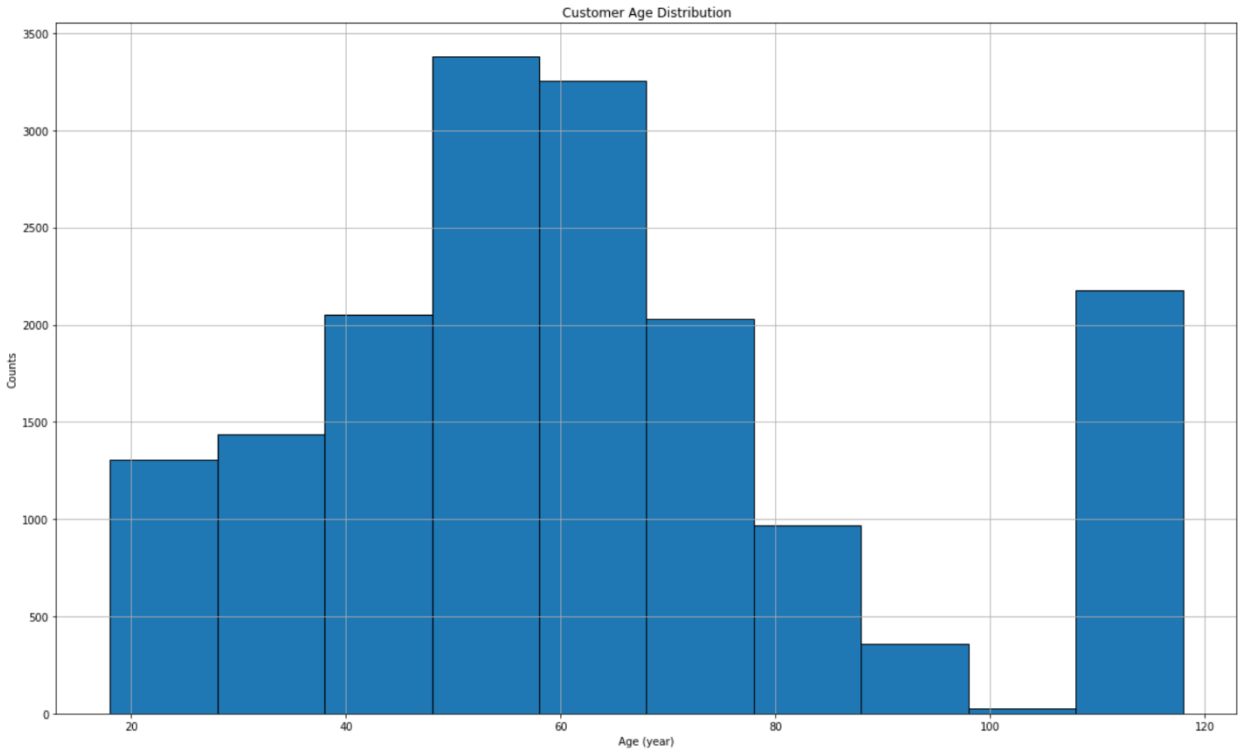

Figure 1: the customer age distribution.

There are over 2000 more female customers than males in our data set. Their income and age distributions by gender were further looked at (they are not affected by the missing values). From the income distribution, the most frequent incomes are from those who annually earn $50000 to $70000. From the age distribution, more customers are from the 50 to 60 age group.

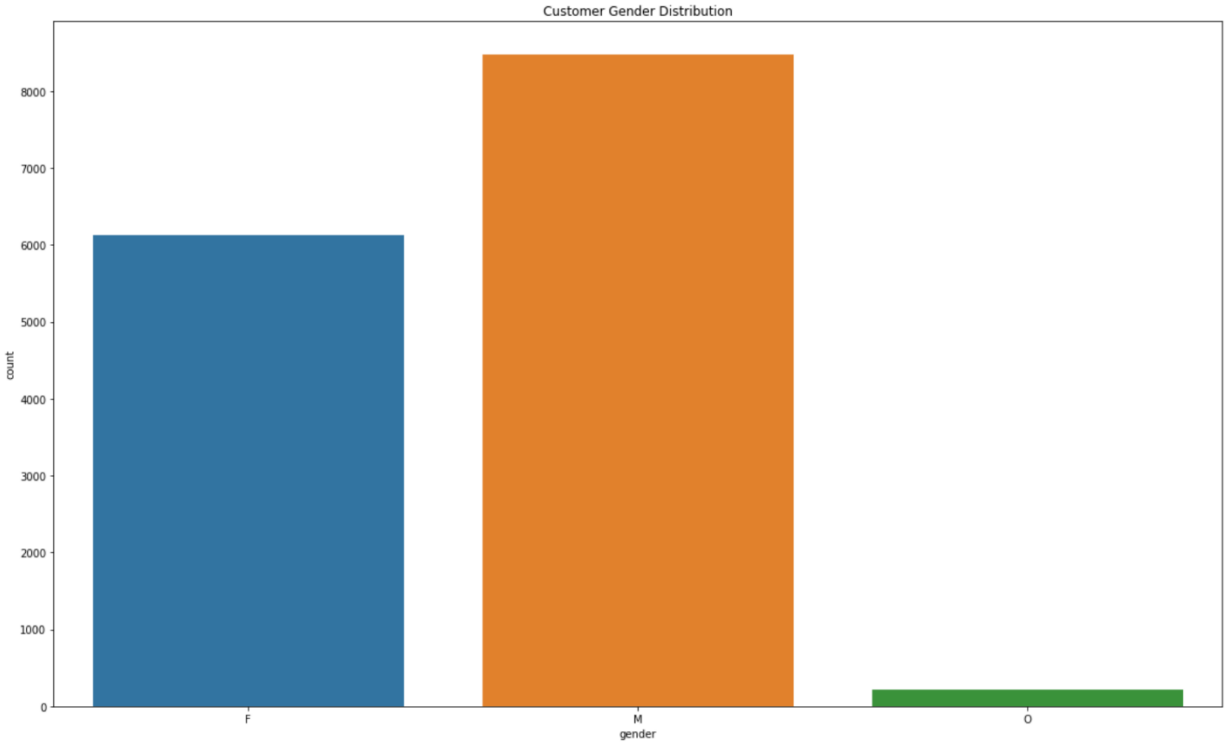

Figure 2: the customer gender distribution.

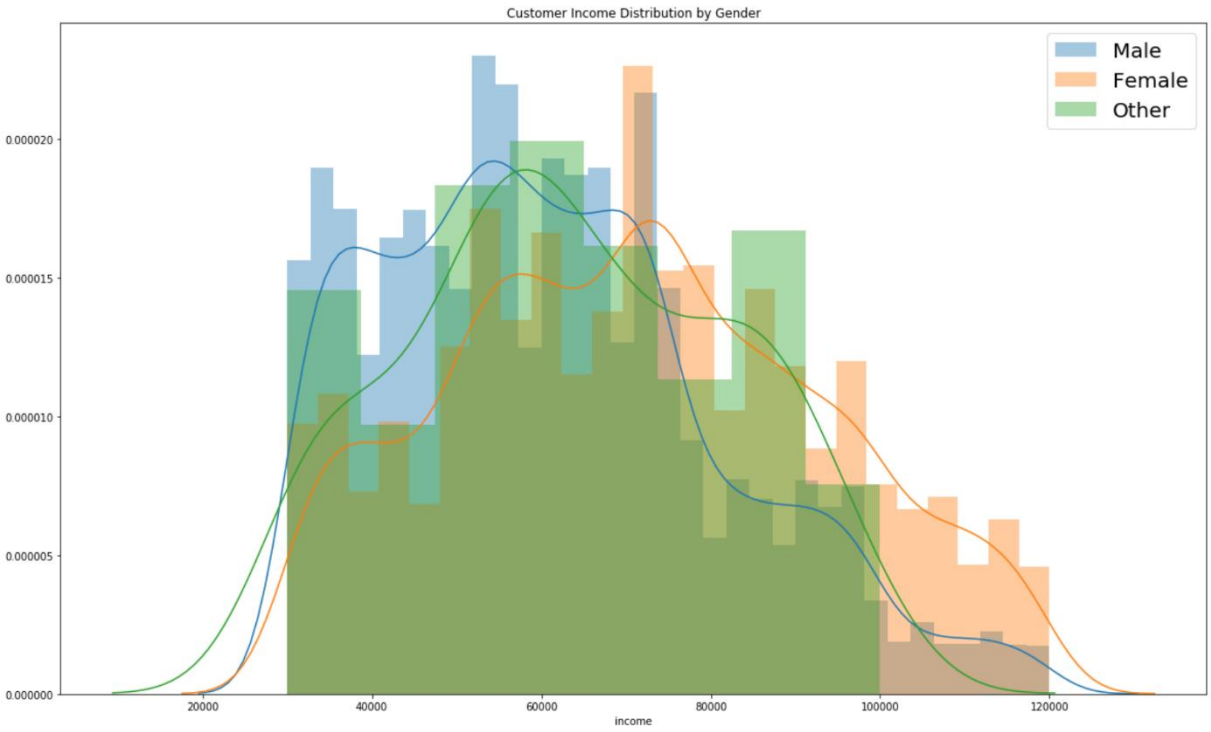

Figure 3: the customer income distribution by gender.

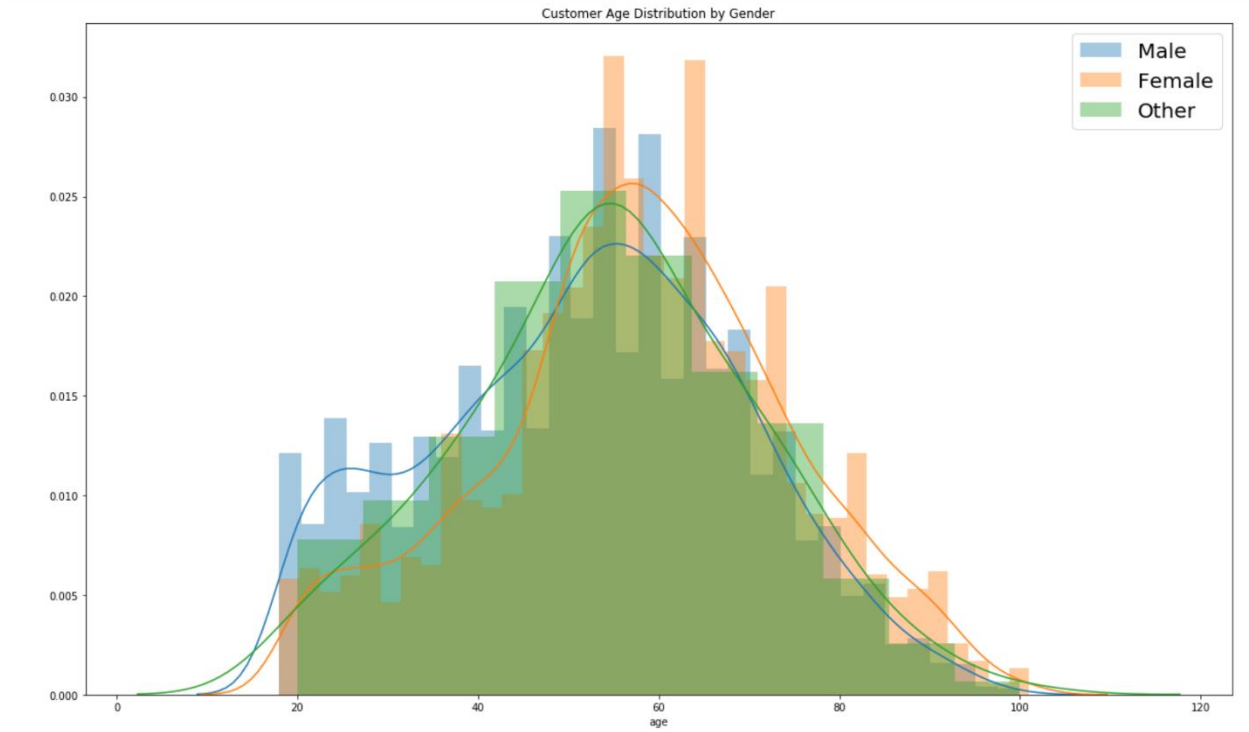

Figure 4: the customer age distribution by gender.

From the membership starting year distribution by gender (extracted from the 'became_member_on' column mentioned above), there is definitely an increase in the number of customers since 2013. Year 2018 does not show the trend since we only have data from the first few months.

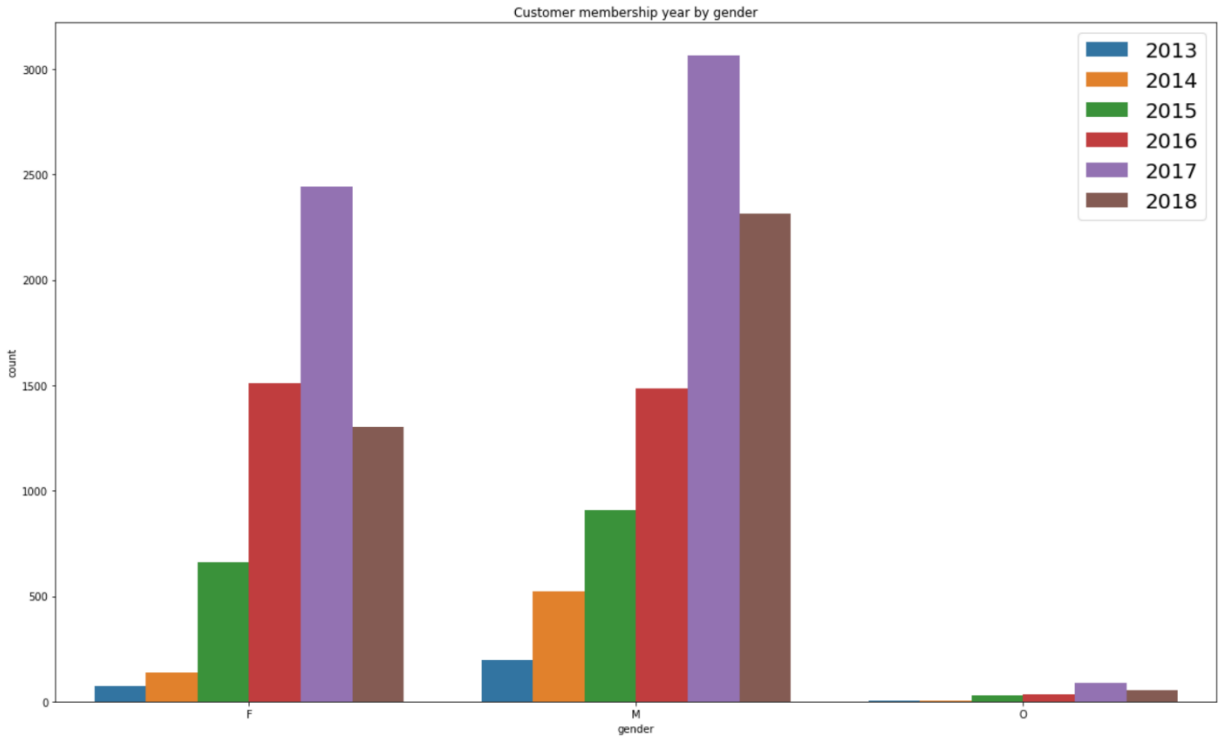

Figure 5: customer membership starting year by gender.

From the ‘transcript’ file, about 140000 transaction events, 75000 offer received events, 60000 offer viewed events and 35000 offer completed events are recorded.

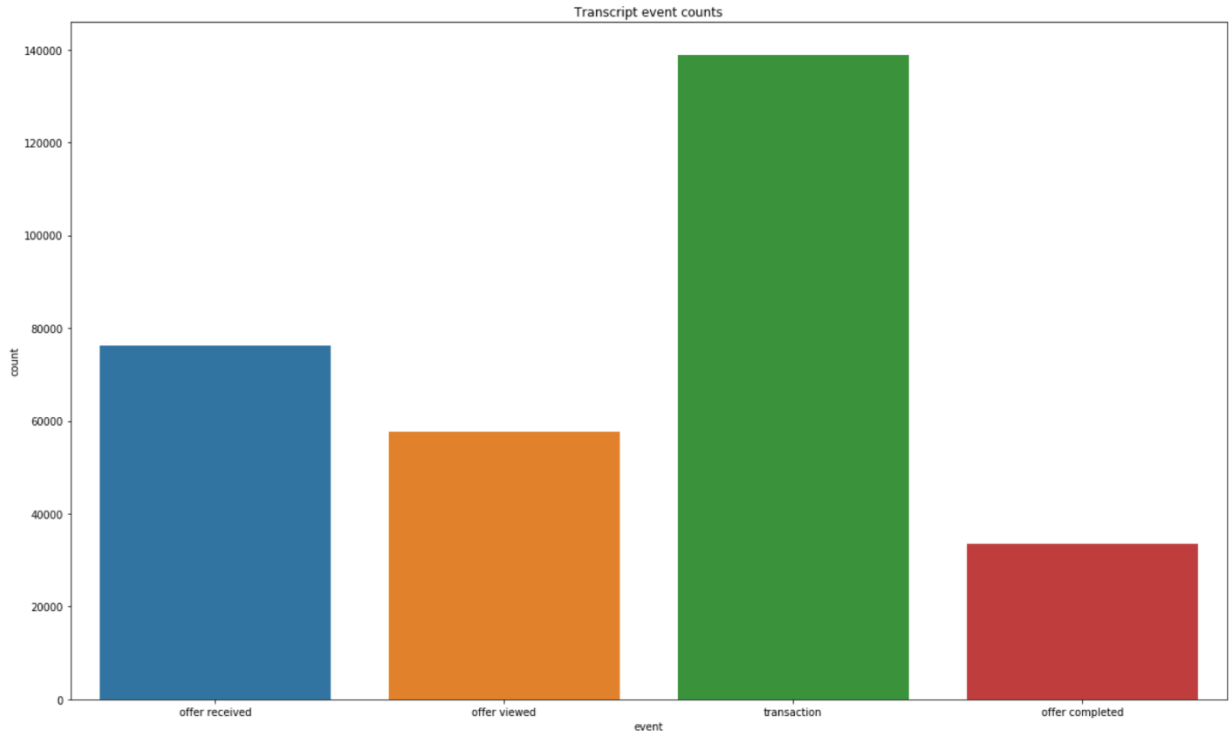

Figure 6: four type of transcript events and counts.

From the counts of the four events as functions of time, it is noted that after an offer is released, there is a boost in the number of transactions within the first 24 to 48 hours, which indicates a typical time scale it takes to influence the customers.

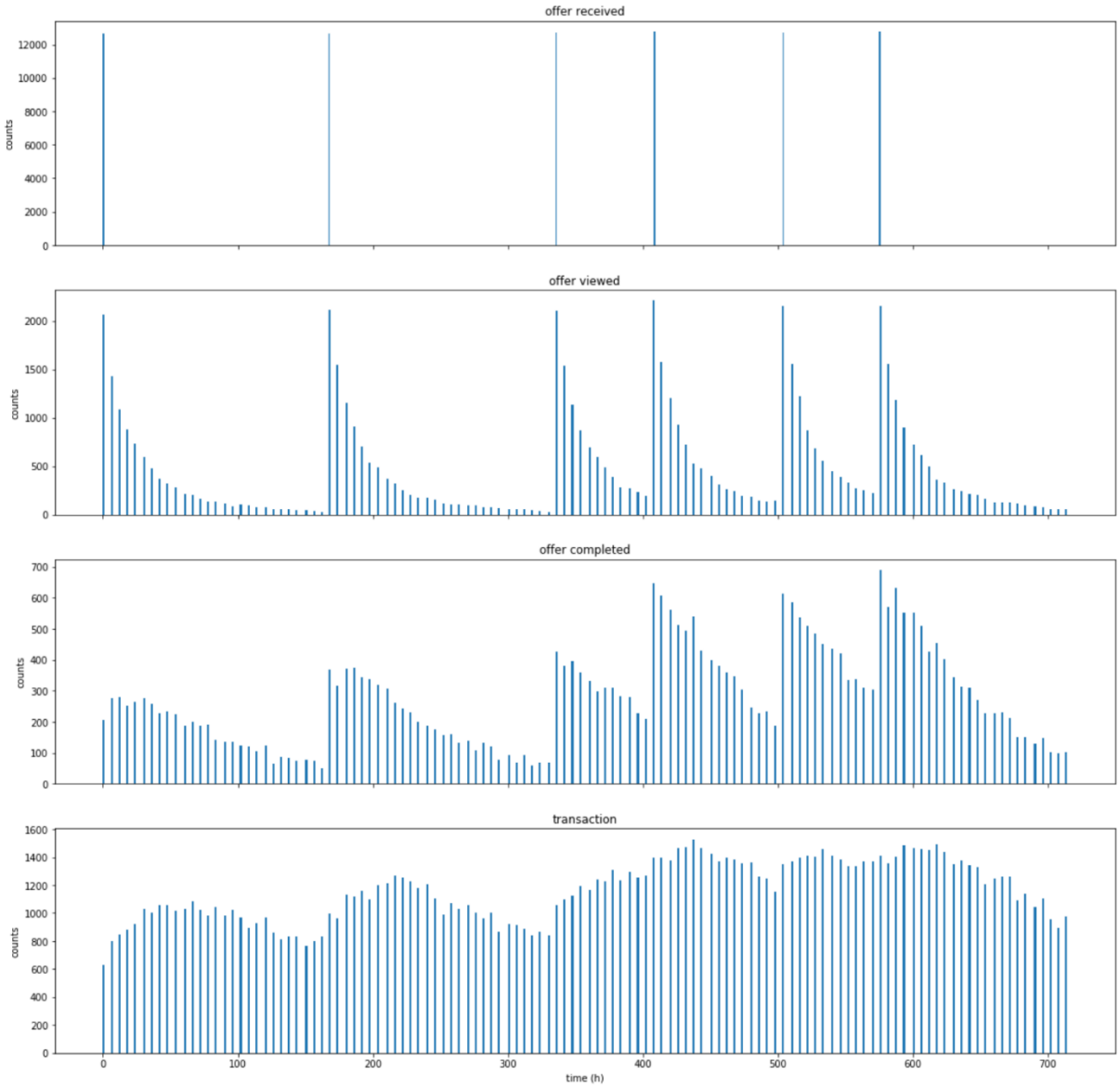

Figure 7: counts of four events as a function of time.

Using df.apply(pd.Series) method on the 'event' column in the 'transcript' data, we can unstack this column into 'offer_id', 'amount' and 'reward' (the meanings were explained in the data exploration section). Further visualizations were performed by first merging the 'transcript' file and 'portfolio' file on the shared 'offer_id' column, then counting the events based on either the offer type or channel. From the two figures below, it is noted that all the transactions are not associated with a certain offer type, and informational offers won't have an offer completed record once successfully responded. There are 30000 BOGO or discount offers, and 15000 informational offers sent to the customers, with more than half of BOGO and discount offers being completed. Therefore, there is room for improvement in terms of increasing the offer conversion rate (successful completion / total number of offers sent).

Comparing the event counts between channels, Starbucks send out more offer to the customers via email than mobile, web or social.

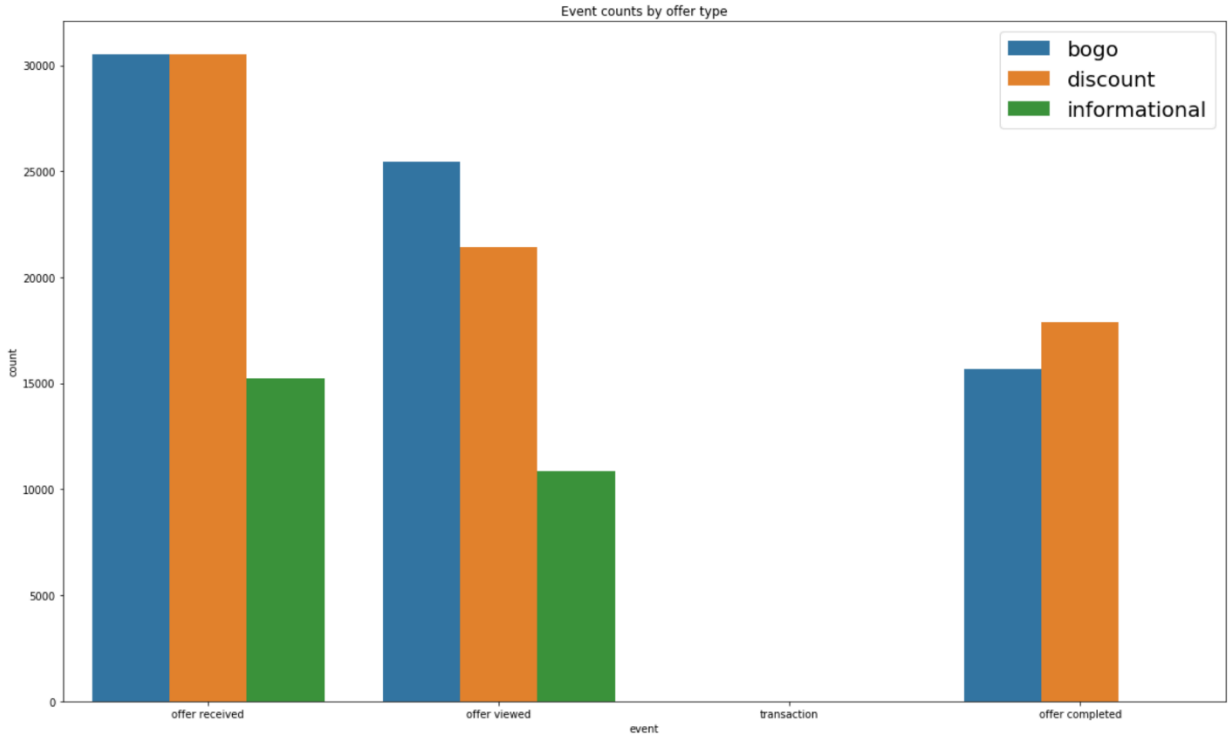

Figure 8: event counts by offer type.

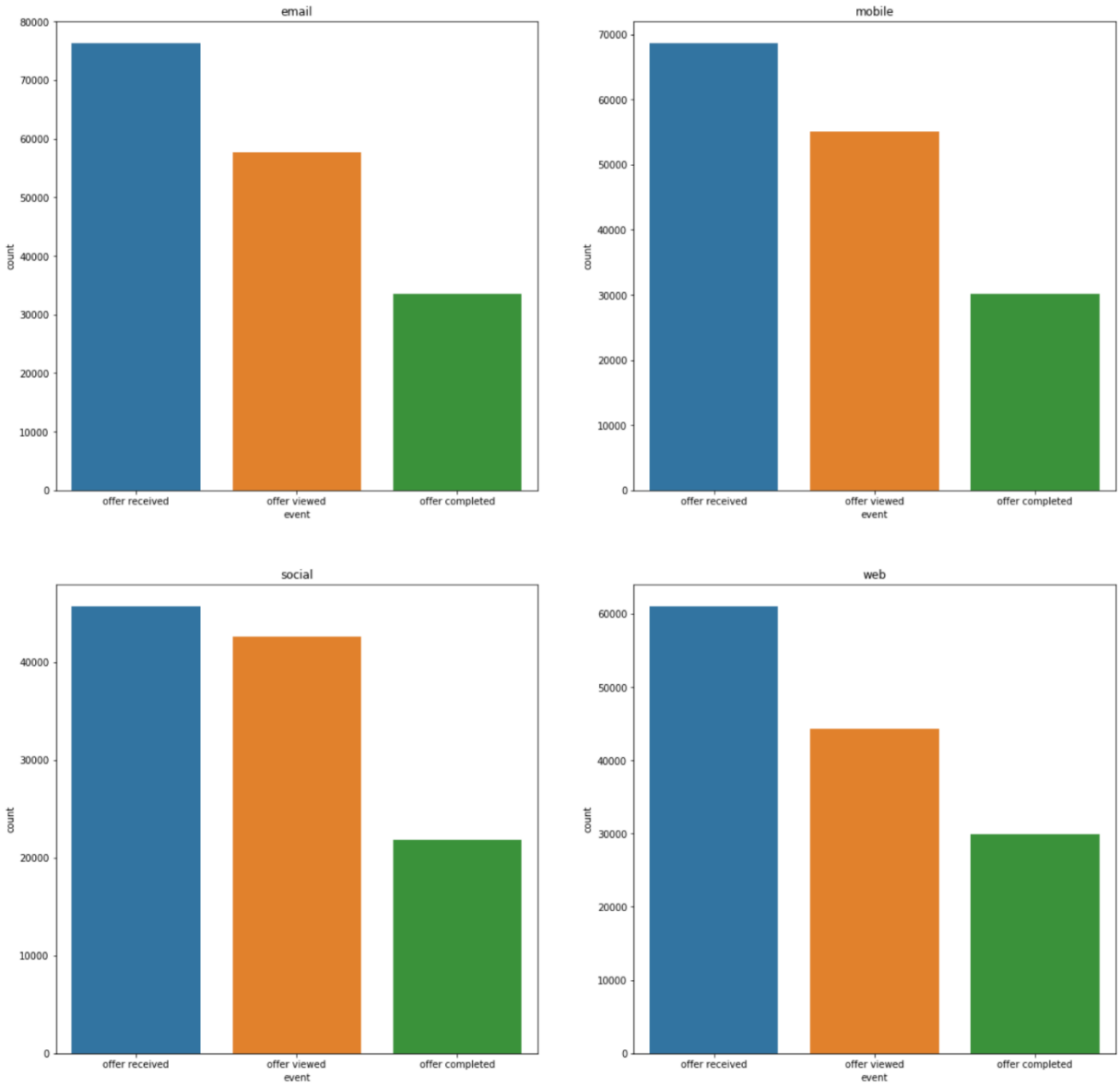

Figure 9: event counts by channel.

Algorithms and techniques:

To build a machine learning classification model to predict whether a customer will response to an offer, the random forest classifier will be used, which is a very popular supervised machine learning algorithm. The random forest classifier will take features like the customer and offer information as input and predict a success label '1' or '0', meaning the customer has /hasn't successfully completed an offer. We will need to first label each person offer_id pair from the transcript data so the classifier can learn from. Meanwhile before actually training the model, we need to clean the data and engineer some features. For numerical variables, we can use the values as features, while for categorical variables, we will create the corresponding dummy variables using one-hot encoding. Besides, it is important for us to properly

scale the features because we would like to let all the features have similar influence on the model (not biased by one or two variables). These techniques will help improve the model's overall performance. The details of each step will be discussed in the following sections.

Benchmark:

To create an initial benchmark for the classifier, I decided to use a random forest classifier and manually set the classifier parameter as follows: the 'max_depth' will be set to 10, the 'max_features' will be set as 'auto', the 'min_samples_split' will be set to 10, the 'n_estimators' will be set to 10, and the 'min_samples_leaf' will be set to 10. The basic idea is that we want to improve the performance of the model by tuning these parameters and the implementation will include the use of grid search. For the improved models, we will compare the scores with those from the benchmark classifier specified here.

Methodology

Data preprocessing:

To accurately label the successful responses from customers to offers, let's first understand the time series of the transcript events. From the previously merged transcript_portfolio data shown below, a successful response consists of offer received (R), offer viewed (V), transaction (T) and offer completed (C). it is noted that the last step is not needed for the informational offer. Such process indicates that the customer is aware and committed to the offer, and we will give these responses, if in the validity period, a success label '1'. There are three other circumstances. First, R-T-C, R-T, T-C or T, with an optional V after T or C. These behaviors indicate that the customers will make purchases even if not viewing the offer. In other words, these customers are loyal to the Starbucks products anyway. From the business perspective, we would not like to send them offers. Second, R, which means the customers received the offer, but not viewing it. And last, R-V, which means the customers received and viewed the offer, but never made the purchases. It is understandable that the last type of response will be labeled '0', in contrast to the successful ones.

Table IV: transcript merged with portfolio (first few rows and columns)

	event	person	time	transaction_amount	reward_received	offer_id	channels	difficulty
55972	offer received	0009655768c64bdeb2e877511632db8f	168	NaN	NaN	5a8bc65990b245e5a138643cd4eb9837	[email, mobile, social]	0.0
77705	offer viewed	0009655768c64bdeb2e877511632db8f	192	NaN	NaN	5a8bc65990b245e5a138643cd4eb9837	[email, mobile, social]	0.0
89291	transaction	0009655768c64bdeb2e877511632db8f	228	22.16	NaN	NaN	NaN	NaN
113605	offer received	0009655768c64bdeb2e877511632db8f	336	NaN	NaN	3f207df678b143eea3cee63160fa8bed	[web, email, mobile]	0.0
139992	offer viewed	0009655768c64bdeb2e877511632db8f	372	NaN	NaN	3f207df678b143eea3cee63160fa8bed	[web, email, mobile]	0.0
153401	offer received	0009655768c64bdeb2e877511632db8f	408	NaN	NaN	f19421c1d4aa40978ebb69ca19b0e20d	[web, email, mobile, social]	5.0
168412	transaction	0009655768c64bdeb2e877511632db8f	414	8.57	NaN	NaN	NaN	NaN
168413	offer completed	0009655768c64bdeb2e877511632db8f	414	NaN	5.0	f19421c1d4aa40978ebb69ca19b0e20d	[web, email, mobile, social]	5.0
187554	offer viewed	0009655768c64bdeb2e877511632db8f	456	NaN	NaN	f19421c1d4aa40978ebb69ca19b0e20d	[web, email, mobile, social]	5.0
204340	offer received	0009655768c64bdeb2e877511632db8f	504	NaN	NaN	fafdcd668e3743c1bb461111dcafc2a4	[web, email, mobile, social]	10.0
228422	transaction	0009655768c64bdeb2e877511632db8f	528	14.11	NaN	NaN	NaN	NaN
228423	offer completed	0009655768c64bdeb2e877511632db8f	528	NaN	2.0	fafdcd668e3743c1bb461111dcafc2a4	[web, email, mobile, social]	10.0

From the explanations above, the key is to understand offer viewed events and transaction events. If there is a transaction happed after an offer viewed from a specific customer to an offer within its validity period, it should be assigned ‘1’, if there is no following offer viewed, or if the offer viewed happened outside the validity period of that offer, it should be labeled ‘0’.

It is noted that transactions do not have associated offer id, therefore, to link transactions to offer viewed events, I will first subset the data to include only offer viewed and transaction event, then forward fill the offer id column for transactions from the previous offer viewed.

Table V: forward fill offer_id for transaction from the previous offer viewed event.

	event	person	time	offer_id
77705	offer viewed	0009655768c64bdeb2e877511632db8f	192	5a8bc65990b245e5a138643cd4eb9837
89291	transaction	0009655768c64bdeb2e877511632db8f	228	5a8bc65990b245e5a138643cd4eb9837
139992	offer viewed	0009655768c64bdeb2e877511632db8f	372	3f207df678b143eea3cee63160fa8bed
168412	transaction	0009655768c64bdeb2e877511632db8f	414	3f207df678b143eea3cee63160fa8bed
187554	offer viewed	0009655768c64bdeb2e877511632db8f	456	f19421c1d4aa40978ebb69ca19b0e20d
228422	transaction	0009655768c64bdeb2e877511632db8f	528	f19421c1d4aa40978ebb69ca19b0e20d
233413	offer viewed	0009655768c64bdeb2e877511632db8f	540	fafdcd668e3743c1bb461111dcafc2a4
237784	transaction	0009655768c64bdeb2e877511632db8f	552	fafdcd668e3743c1bb461111dcafc2a4
258883	transaction	0009655768c64bdeb2e877511632db8f	576	fafdcd668e3743c1bb461111dcafc2a4

Then we compare the offer_id of a transaction or an offer competed event with the offer_id from its previous offer viewed event, if they are the same, it means the customer responded to the offer. It is noted that this is not equivalent to a successful response for the case of informational offer transaction since we also need to consider the validity period. Therefore, I will give such response an 'effectiveness' value of '1' or '0'. The case when an offer completed event received a '0' is when the customer purchases a product without viewing the offer. Technically, shift method is used to obtain the previous id for comparison, with details shown in the notebook.

Table VI: an 'effectiveness' variable indicating a customer might respond to an offer.

	event	person	time	offer_id	effectiveness
1	offer viewed	0009655768c64bdeb2e877511632db8f	192	5a8bc65990b245e5a138643cd4eb9837	0
2	transaction	0009655768c64bdeb2e877511632db8f	228	5a8bc65990b245e5a138643cd4eb9837	1
4	offer viewed	0009655768c64bdeb2e877511632db8f	372	3f207df678b143eea3cee63160fa8bed	0
6	transaction	0009655768c64bdeb2e877511632db8f	414	3f207df678b143eea3cee63160fa8bed	1
7	offer completed	0009655768c64bdeb2e877511632db8f	414	f19421c1d4aa40978ebb69ca19b0e20d	0

Next, we need to separately treat BOGO/discount and informational offers. For BOGO/discount, if the effectiveness value of the offer completed event is '1', then it is a successful response. We will pick up the person and offer_id columns from these qualified customers and give them a success label '1'.

Table VII: person and offer id pairs showing successful response to an offer.

	person	offer_id	success
0	0011e0d4e6b944f998e987f904e8c1e5	9b98b8c7a33c4b65b9aebfe6a799e6d9	1
1	0020c2b971eb4e9188eac86d93036a77	4d5c57ea9a6940dd891ad53e9dbe8da0	1
2	0020ccbbb6d84e358d3414a3ff76cffd	9b98b8c7a33c4b65b9aebfe6a799e6d9	1
3	0020ccbbb6d84e358d3414a3ff76cffd	f19421c1d4aa40978ebb69ca19b0e20d	1
4	004b041fbfe44859945daa2c7f79ee64	f19421c1d4aa40978ebb69ca19b0e20d	1

To find the customers who received and viewed the offer but not making purchases, first, a right merge between transaction or offer completed subset and offer received subset was performed. If we set the merge indicator on, 'right only' will help separate the customers who only received offers without further purchase. These customers may or may not have seen the offers. Then we merge this 'right only' set back to the transcript_portfolio data and further subset out the offer viewed events. Eventually, these events correspond to customers who have failed responses to the offers, and we label the person offer_id pair a success label '0'. The details can be found in the notebook.

For informational offers, since there is no offer completed event associated, we need to find whether the transactions are valid by comparing the transaction time and offer duration. First, we convert all the event time in hours to days, then we group by person and offer_id the offer received and transaction subset and use the diff method to compute the time it takes from receiving an offer to making a transaction, and compare it with the duration column, if the diff value is less than or equal to duration, then we will give the transaction a 'valid' label of '1'.

Table VIII: valid transaction indicator (valid) based on transaction time (diff) for informational offers.

	offer_id	channels	difficulty	duration	offer_type	...	mobile	social	web	bogo	discount	informational	effectiveness	time_in_day	diff	valid
	138643cd4eb9837	[email, mobile, social]	0	3	informational	...	1	1	0	0	0	1	0.0	7.0	NaN	0.0
	138643cd4eb9837	[email, mobile, social]	0	3	informational	...	1	1	0	0	0	1	0.0	8.0	NaN	0.0
	138643cd4eb9837	[email, mobile, social]	0	3	informational	...	1	1	0	0	0	1	1.0	9.5	2.5	1.0
	ı3cee63160fa8bed	[web, email, mobile]	0	4	informational	...	1	0	1	0	0	1	0.0	14.0	NaN	0.0
	ı3cee63160fa8bed	[web, email, mobile]	0	4	informational	...	1	0	1	0	0	1	0.0	15.5	NaN	0.0

If an informational offer transaction received an effectiveness label of '1' and a 'valid' label of '1', then it is a successful response. Similarly, we can use the same 'right merge' technique to find customers who only received and viewed the offer. And finally, we assign success label of '1' and '0' for each case.

Let's think about what we have achieved so far. For each type of offer, we have generated a new data frame including the person id, offer id and a column named success. A success value of 1 or 0 indicates that a person successfully / unsuccessfully responded to a certain offer. Therefore, the success column will be used as labels in our classification model.

The next thing we need to do is feature engineering. Features come from two sources, 'profile' which contains the customers' demographic information, 'portfolio' which contains the offer information. Let's focus on the customers first. From the member year, month and day columns, the membership days for all the customers were computed, this is an important feature because I speculate that a senior member will be more likely to complete offers than a new member who uses the app less often. This variable is named 'member_days'. Aside from membership days, other personal information I used as features are: the customers age, gender (dummy variables created), income.

In terms of offer information, one feature I would like to engineer and explore is the total number of offers a customer received. I suspect that the more offers one customer received (regardless of the type), the more likely the customer will be influenced, which will encourage the customer to make more purchases. Therefore, I counted the offer number by grouping the offer received events for each customer and created a new variable called 'num_offer_received', and I found that aside from 6 customers who never received any offers, a typical customer will receive 1 to 6 offers in total during the data collection period. Besides, I also used channels (dummy variables created), difficulty, duration, reward from the portfolio file as other features.

Implementation:

To recap what we have so far, for a person offer_id pair, we labeled using a 'success' variable whether the customer has successfully responded to the offer. We have engineered some features for the classification model. The random forest classifier will be implemented to train on our data.

Two functions are defined. The 'feature_terget' function is used to create feature and target variables for the classifier. 'success' is selected as the target label and other variables are used as features with an optional parameter 'columns_to_drop' to exclude variables you don't want to use in the classifier. The 'train_test' function is implemented to split the data into train and test sets. StandardScaler is used to normalize the numerical variables and standardize features by removing the mean and scaling to unit variance.

```
def features_target(df, columns_to_drop):
    target = df['success']
    features = df.drop(columns_to_drop, axis=1, inplace=False)
    return features, target
```

Function 1: feature target selection function.

```
def train_test(features, target):
    X_train, X_test, y_train, y_test = train_test_split(features, target, test_size=0.2, random_state=23)

    scaler = StandardScaler()
    X_train = scaler.fit_transform(X_train)
    X_test = scaler.transform(X_test)

    return X_train, X_test, y_train, y_test
```

Function 2: train test split function.

For the training and prediction process, I defined a function called 'train_predict_model', which takes the model used and train test data as input, trains the model on the data and returns the train and test accuracies and f1-scores.

```
def train_predict_model(model, X_train, y_train, X_test, y_test):
    results = {}

    start = time()
    model = model.fit(X_train, y_train)
    end = time()
    results['train_time'] = end - start

    start = time()
    pred_train = model.predict(X_train)
    pred_test = model.predict(X_test)
    end = time()
    results['pred_time'] = end - start

    results['training_score'] = model.score(X_train, y_train)
    results['testing_score'] = model.score(X_test, y_test)

    print("{} trained on {} samples.".format(model.__class__.__name__, len(y_train)))
    print("{} tested on {} samples.".format(model.__class__.__name__, len(y_test)))
    print(classification_report(y_train, pred_train, digits=4))
    print(classification_report(y_test, pred_test, digits=4))
    return results
```

Function 3: model training and prediction function.

For the benchmark comparison, I used a random forest classifier with the following parameters. I set the max_depth to 10, max_features as 'auto', min_samples_split to 10, n_estimators to 10, and min_samples_leaf to 10.

```
model_RF = RandomForestClassifier(random_state=2, max_depth=10, max_features='auto',
                                  min_samples_split=10, n_estimators=10, min_samples_leaf=10)
```

Function 4: the benchmark random forest classifier model.

The three functions were implemented in order onto data of each offer. From the three figures below, summary of results are as follows. For BOGO offers, the train accuracy is 0.836, the train f1-score is 0.781, the test accuracy is 0.821, the test f1-socre is 0.760. For discount offers, the train accuracy is 0.867, the train f1-score is 0.810, the test accuracy is 0.863, the test f1-socre is 0.803. For informational offers, the train accuracy is 0.771, the train f1-score is 0.700, the test accuracy is 0.729, the test f1-socre is 0.644.

```
In [135]: results_RF_bogo = train_predict_model(model_RF, X_train1, y_train1, X_test1, y_test1)

RandomForestClassifier trained on 9829 samples.
RandomForestClassifier tested on 2458 samples.
              precision    recall  f1-score   support

           0     0.7592    0.1069    0.1874      1740
           1     0.8379    0.9927    0.9087      8089

   micro avg     0.8359    0.8359    0.8359      9829
   macro avg     0.7985    0.5498    0.5481      9829
weighted avg     0.8239    0.8359    0.7810      9829

              precision    recall  f1-score   support

           0     0.5536    0.0697    0.1238       445
           1     0.8276    0.9876    0.9006      2013

   micro avg     0.8214    0.8214    0.8214      2458
   macro avg     0.6906    0.5286    0.5122      2458
weighted avg     0.7780    0.8214    0.7599      2458

In [136]: results_RF_bogo

Out[136]: {'train_time': 0.05391716957092285,
 'pred_time': 0.016538143157958984,
 'training_score': 0.8358937837012921,
 'testing_score': 0.8213995117982099}
```

Figure 10: classifier scores for the BOGO offer data.

```
In [138]: results_RF_discount = train_predict_model(model_RF, X_train2, y_train2, X_test2, y_test2)

RandomForestClassifier trained on 10179 samples.
RandomForestClassifier tested on 2545 samples.
              precision    recall  f1-score   support

           0     0.7609    0.0255    0.0493      1373
           1     0.8680    0.9988    0.9288      8806

   micro avg     0.8675    0.8675    0.8675     10179
   macro avg     0.8144    0.5121    0.4891     10179
weighted avg     0.8535    0.8675    0.8101     10179

              precision    recall  f1-score   support

           0     0.5000    0.0144    0.0279       348
           1     0.8647    0.9977    0.9265      2197

   micro avg     0.8633    0.8633    0.8633      2545
   macro avg     0.6823    0.5060    0.4772      2545
weighted avg     0.8148    0.8633    0.8036      2545

In [139]: results_RF_discount

Out[139]: {'train_time': 0.04492902755737305,
           'pred_time': 0.018595218658447266,
           'training_score': 0.8674722467825916,
           'testing_score': 0.8632612966601179}
```

Figure 11: classifier scores for the discount offer data.

```
In [141]: results_RF_informational = train_predict_model(model_RF, X_train3, y_train3, X_test3, y_test3)

RandomForestClassifier trained on 5780 samples.
RandomForestClassifier tested on 1445 samples.
              precision    recall  f1-score   support

           0     0.7512    0.1117    0.1945      1432
           1     0.7715    0.9878    0.8664      4348

   micro avg     0.7708    0.7708    0.7708      5780
   macro avg     0.7613    0.5498    0.5304      5780
weighted avg     0.7665    0.7708    0.6999      5780

              precision    recall  f1-score   support

           0     0.4906    0.0665    0.1171       391
           1     0.7378    0.9744    0.8397      1054

   micro avg     0.7287    0.7287    0.7287      1445
   macro avg     0.6142    0.5204    0.4784      1445
weighted avg     0.6709    0.7287    0.6442      1445

In [142]: results_RF_informational

Out[142]: {'train_time': 0.03624677658081055,
           'pred_time': 0.011182785034179688,
           'training_score': 0.7707612456747405,
           'testing_score': 0.728719723183391}
```

Figure 12: classifier scores for the informational offer data.

Refinement:

To improve the model performance, grid search was used to obtain the best parameters of the classifier. The grid search function is defined below. The function was run on the training set of data of each offer type. The optimal parameters were obtained, then the classifiers were retrained with these parameters. The optimal parameters and model scores are show in figures 13, 14, 15.

```python
# grid search for best prms
def RF_prm_search(X,y):
    param_grid={'max_features': ['auto', 'sqrt'],
                'max_depth' : [10, 15],
                'n_estimators': [10, 20, 25, 30],
                'min_samples_split': [10, 20],
                'min_samples_leaf': [10, 15],
                }
    grid_search = GridSearchCV(RandomForestClassifier(random_state=2), param_grid)
    grid_search.fit(X, y)
    grid_search.best_params_
    return grid_search.best_params_
```

Function 5: grid search for the best parameters of the random forest classifier.

```
Out[144]: {'max_depth': 10,
           'max_features': 'auto',
           'min_samples_leaf': 10,
           'min_samples_split': 10,
           'n_estimators': 30}
```

```
In [145]: model_RF_bogo_best = RandomForestClassifier(random_state=2, max_depth=10, max_features='auto', min_samples_split= 10, n_estir
```

```
In [146]: results_RF_bogo_best = train_predict_model(model_RF_bogo_best, X_train1, y_train1, X_test1, y_test1)
```

```
RandomForestClassifier trained on 9829 samples.
RandomForestClassifier tested on 2458 samples.
              precision    recall  f1-score   support

           0     0.7626    0.0868    0.1558      1740
           1     0.8350    0.9942    0.9077      8089

   micro avg     0.8336    0.8336    0.8336      9829
   macro avg     0.7988    0.5405    0.5318      9829
weighted avg     0.8222    0.8336    0.7746      9829

              precision    recall  f1-score   support

           0     0.5455    0.0539    0.0982       445
           1     0.8256    0.9901    0.9004      2013

   micro avg     0.8206    0.8206    0.8206      2458
   macro avg     0.6855    0.5220    0.4993      2458
weighted avg     0.7749    0.8206    0.7551      2458
```

```
In [147]: results_RF_bogo_best
```

```
Out[147]: {'train_time': 0.12049317359924316,
           'pred_time': 0.04210090637207031,
           'training_score': 0.8335537694577272,
           'testing_score': 0.8205858421480878}
```

Figure 13: classifier scores for the BOGO offer data (optimal).

```
Out[148]: {'max_depth': 10,
           'max_features': 'auto',
           'min_samples_leaf': 15,
           'min_samples_split': 10,
           'n_estimators': 10}
```

```
In [149]: model_RF_discount_best = RandomForestClassifier(random_state=2, max_depth=10, max_features='auto', min_samples_split= 10, n_e
```

```
In [150]: results_RF_discount_best = train_predict_model(model_RF_discount_best, X_train2, y_train2, X_test2, y_test2)
```

```
RandomForestClassifier trained on 10179 samples.
RandomForestClassifier tested on 2545 samples.
              precision    recall  f1-score   support

           0     0.7812    0.0182    0.0356      1373
           1     0.8672    0.9992    0.9285      8806

   micro avg     0.8669    0.8669    0.8669     10179
   macro avg     0.8242    0.5087    0.4820     10179
weighted avg     0.8556    0.8669    0.8081     10179

              precision    recall  f1-score   support

           0     0.0000    0.0000    0.0000       348
           1     0.8632    0.9991    0.9262      2197

   micro avg     0.8625    0.8625    0.8625      2545
   macro avg     0.4316    0.4995    0.4631      2545
weighted avg     0.7451    0.8625    0.7995      2545
```

```
In [151]: results_RF_discount_best
```

```
Out[151]: {'train_time': 0.04224586486816406,
           'pred_time': 0.01970529556274414,
           'training_score': 0.8668827979172806,
           'testing_score': 0.862475442043222}
```

Figure 14: classifier scores for the discount offer data (optimal).

```
Out[152]: {'max_depth': 10,
           'max_features': 'auto',
           'min_samples_leaf': 10,
           'min_samples_split': 10,
           'n_estimators': 20}
```

```
In [153]: model_RF_informational_best = RandomForestClassifier(random_state=2, max_depth=10, max_features='auto', min_samples_split= 10
```

```
In [154]: results_RF_informational_best = train_predict_model(model_RF_informational_best, X_train3, y_train3, X_test3, y_test3)
```

```
RandomForestClassifier trained on 5780 samples.
RandomForestClassifier tested on 1445 samples.
              precision    recall  f1-score   support

           0     0.7700    0.1145    0.1994      1432
           1     0.7722    0.9887    0.8672      4348

   micro avg     0.7721    0.7721    0.7721      5780
   macro avg     0.7711    0.5516    0.5333      5780
weighted avg     0.7717    0.7721    0.7017      5780

              precision    recall  f1-score   support

           0     0.5000    0.0691    0.1213       391
           1     0.7383    0.9744    0.8401      1054

   micro avg     0.7294    0.7294    0.7294      1445
   macro avg     0.6192    0.5217    0.4807      1445
weighted avg     0.6738    0.7294    0.6456      1445
```

```
In [155]: results_RF_informational_best
```

```
Out[155]: {'train_time': 0.05941605567932129,
           'pred_time': 0.013134956359863281,
           'training_score': 0.7721453287197232,
           'testing_score': 0.7294117647058823}
```

Figure 15: classifier scores for the informational offer data (optimal).

Result

Model evaluation and validation:

Here is a brief summary of the benchmark model scores and optimal model scores trained on data of each offer. The comparison will be discussed in the justification section.

Table IX: train test scores of models on each offer type with benchmark parameters applied.

	BOGO	Discount	Informational
Train accuracy	0.836	0.867	0.771
Test accuracy	0.821	0.863	0.729
Train f1-score	0.781	0.810	0.700
Test f1-score	0.760	0.803	0.644

Table X: train test scores of models on each offer type with optimal parameters applied.

	BOGO	Discount	Informational
Train accuracy	0.834	0.867	0.772
Test accuracy	0.821	0.862	0.729
Train f1-score	0.775	0.808	0.701
Test f1-score	0.755	0.800	0.649

Justification:

If we compare the optimal classifiers with the benchmark classifiers, here are some observations. For BOGO, discount and informational offers, there are little improvements in the train and test accuracies, also the train and test f1-scores, because the benchmark model has almost the same train and test accuracies as the optimal model.

However, from these results, I believe the classification models are actually a big improvement in terms of helping the company better send out offers. Recall from the exploratory visualization section. Only 50% to 60% of customers successfully responded to the offers they received, in other words, the baseline conversion rate is 50% to 60%. For example, if we send out 10000 offers, then only 5000 of them are responded, and the remaining 5000 offers are actually ineffective. Meanwhile, if the classification models were implemented, the company will already know before sending out offers that which offers might be ineffective, with a fairly confidence level of about 80%. Instead of sending all 10000 offers, the company can target on the ‘success’ customers and save expenses by not sending offers to the ‘not success’ customers.

Conclusion

Free form visualization:

One thing I'm interested in is the feature importance. For example, if we have evidence that females are more responsive to the offers than males, the company can put more emphasis on female customers. I plotted the feature importance for each offer types and they are shown below.

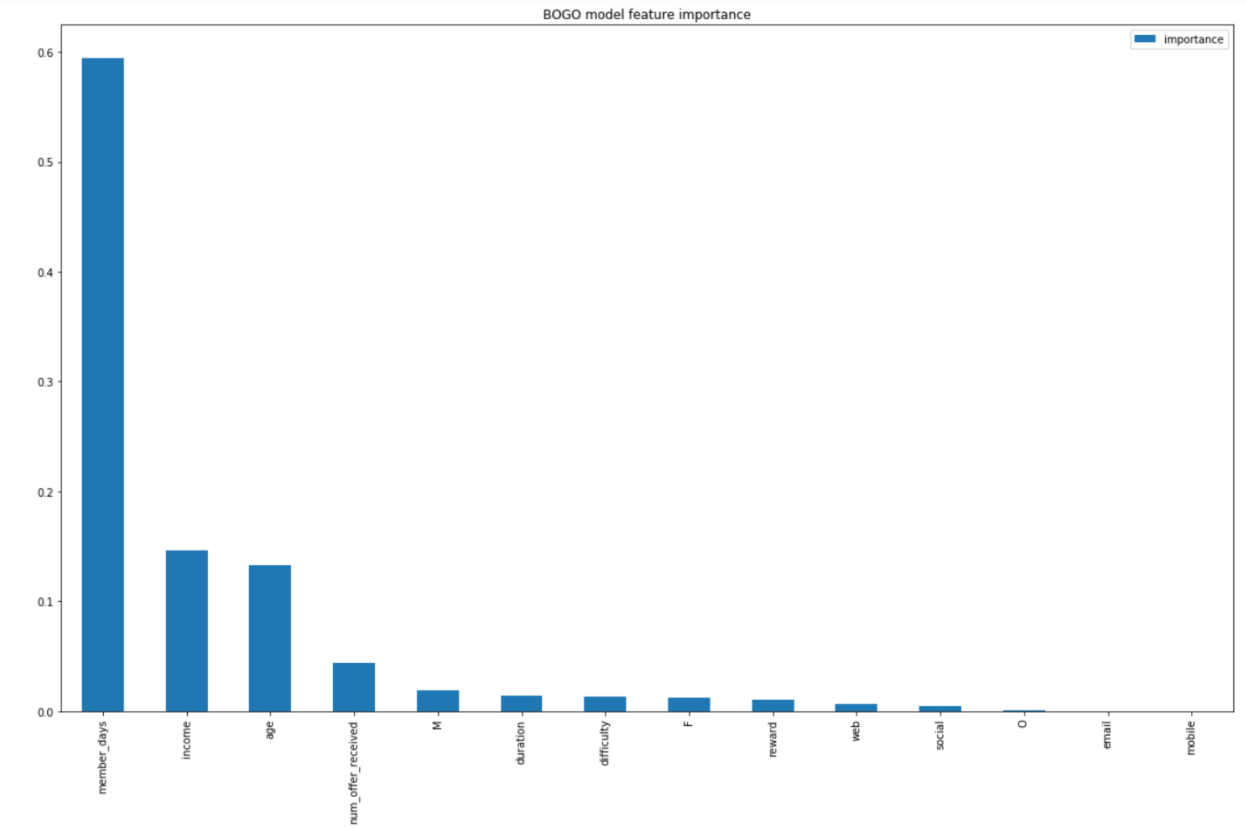

Figure 16: feature importance for the BOGO offer.

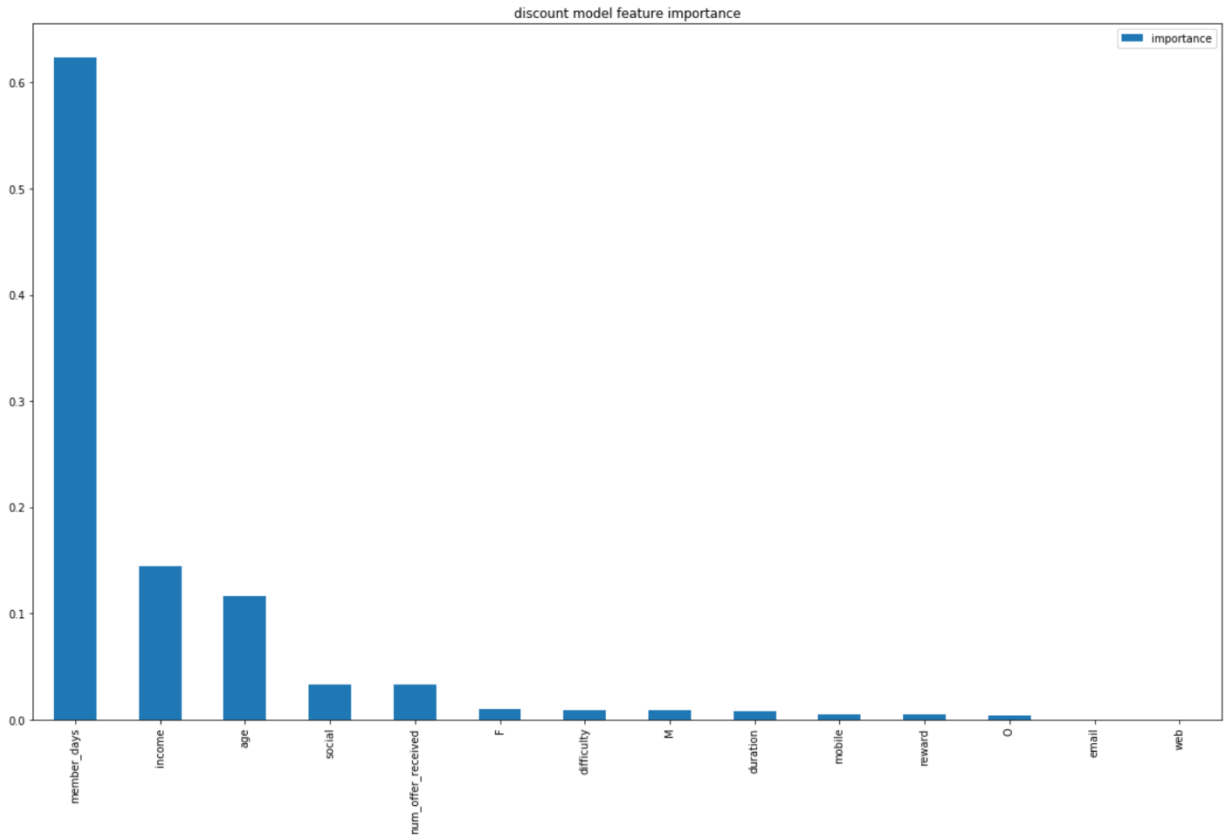

Figure 17: feature importance for the discount offer.

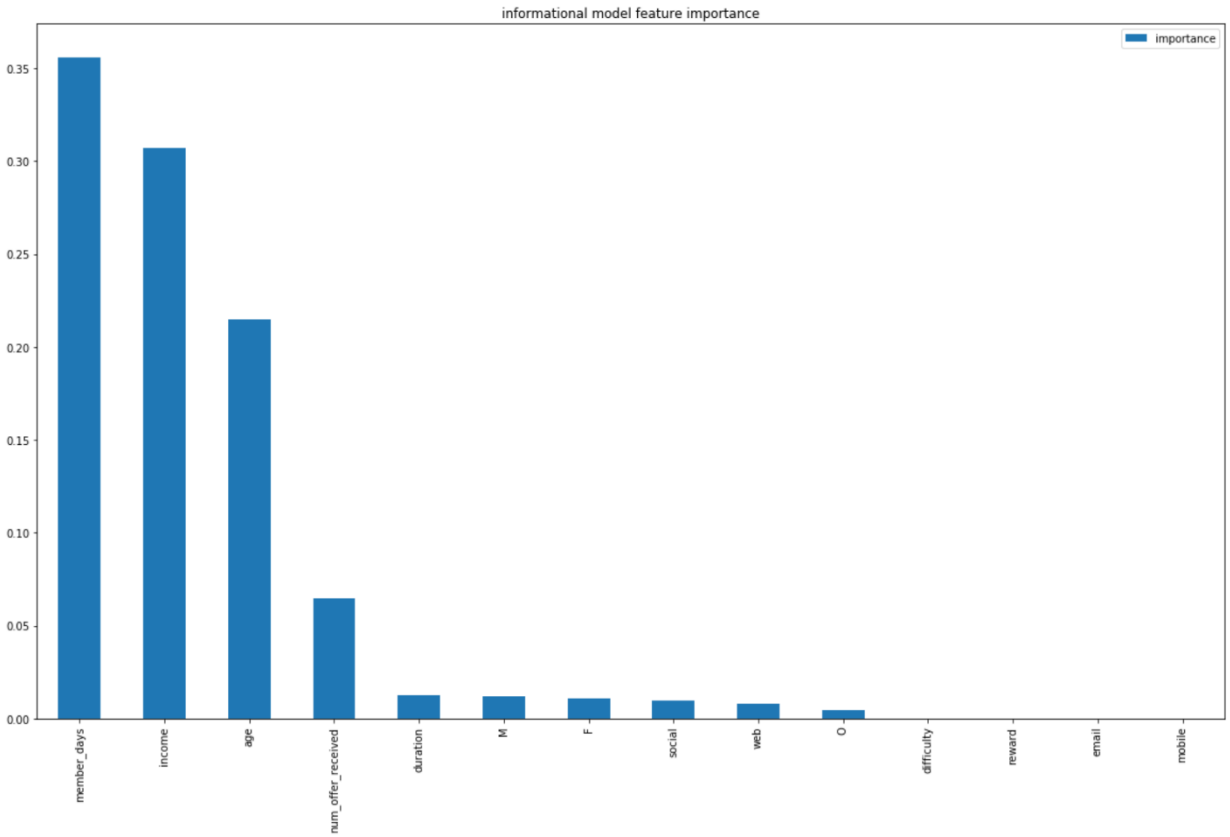

Figure 18: feature importance for the informational offer.

For BOGO, discount and informational offers, membership day, income and age are the top 3 most important features, which actually confirms my assumption before that the longer the customers have used the app, the more likely they will make more purchases with the offer. It is noted that, while membership day dominates the BOGO and discount offer models, income and age are much more important for the informational offer model. This is understandable because both BOGO and discount offers have actual money reward, which means they provide cheaper products that are more affordable to the customers. Whereas for the informational offer, it seems that the customers will think about whether the 'expensive' products are worth their money.

Reflection:

At the beginning of the project, I plan to build a machine learning classification model to predict whether a Starbuck customer will respond to a certain type of offer. We successfully created a classification model to predict whether a customer will respond to an offer, and we have achieved an accuracy score from 72.9% to 86.7%, and a f1-score from 64.9% to 80.8%, depending on the offer type. Aside from this, we also looked at the features importance, the results help the company better target the customers they should send out the offers and increase the revenue.

Improvement:

There is always room for improvements. For example, we can use other classification models, such as decision tree to classify and compare with the random forest model. We can further engineer other features. For example, we can set up several age bins and explore the age dependence. Besides, we can also build machine learning regression models and predict the amounts the customers will spend when making purchases.

Last but not least, in this project, I focused on the customers who successfully (R-V-T-C) respond to the offers, or who fail to respond (R-V). Recall I mentioned before that there are other types of customers, such as customers who received the offer but didn't view it, and loyal customers who would purchase the products regardless of offers. Exploratory analysis can be performed on those customers, or we can build classification models as well to identify them.