Starbuck Offer Study
Data Scientist Capstone Report

by Qiusha Fan
26.04.2020

1. Project Definition

1.1 Project Overview

This project uses simulated data from Starbucks promotions. The Starbucks platform sends out every couple of days some promotion or information about their products to the customers through email, mobile phone, social media or via web. The platform will keep following up the status of the offer, when it is received, if and when it has been viewed by a customer, and finally, if and when it has been completed through purchase in Starbucks shop.

These records reflect customers' behaviour, they could be helpful to the company to better target and send the most suitable Ads or promotions to their customers.

Three datasets are provided:

- promotion list: collect details about promotions. There are three types of promotions: BOGO (buy one get one), discount, and informational offers. Details of offer, such as valid days, minimum spendings, rewards, are also provided,
- customer profile: age, income, gender, the date of registration as a member are saved in this dataset,
- transaction & offer records: transaction records when a customer purchases a product and offer status updates are recorded in this dataset.

1.2 Problem Statement

The task of this problem is to combine transaction, demographic and offer data to determine which demographic groups respond best to which offer type.

This looks like a multi-classification problem ion the first glance. We have three offers, we need to decide for each customer which offer type fits them. Or we can use clustering, to split our customer to three clusters.

But considering one customer may have more than one type of offer that fits him, or even all three types of offers all fit him. From the business aspect, we want such a customer to get all offers. So multi classification or clustering will not give such a possibility.

So I choose to train three binary classifiers for each offer type:

1. who will be the target customer of the bogo?
2. who will be the target customer of the discount?
3. who will be the target of informational ads?

Features will be mainly from demographic and offer data, labels will be derived from records.

To label the data, I need to define what is a correct match between customer and offer type. The promotion all start from the event "offer received", but end differently. For each offer that has been sent to each customer, the result of the promotion will be evaluated by a transaction followed by offer received, offer viewed, and offer completed (completed not required for informational offer). As shown following:

received -> viewed -> transaction -> completed
received -> viewed -> transaction (informational)

For informational offer, the time period between offer receive and transaction shall be shorter than the duration of the offer.

Since the goal is to learn the effectiveness of the promotion, any transactions or completed offers that happened without viewing the offer, or before viewing the offer will be filtered out.

Any "offer viewed" event that fulfils these conditions, will be labeled as "positive". Those got viewed but no valid transaction followed belong to the "negative" group. The offers not got viewed cannot be labeled, thus will be excluded from the training or test.

After the data is labeled, I will feed them into a machine learning model to learn the features. I choose three kinds of models: logistic regression, decision tree, and random forest. Logistic regression is a simple approach to the problem of binary classification, so serves as the basis. I expect the random forest approach will perform the best, being an ensemble method. This is most common approach nowadays to apply random forest since it combines power of weak learners, is fast and offers relatively good interpretability of the features (important for improvements).

1.3 Metrics

The problem stated is a binary classification problem (for each of the 3 offers, as mentioned in the previous section). Thus, I will use accuracy and F1 score (combining precision and recall) to evaluate my model.

Accuracy is the percentage of correct predictions over all predictions. It is a general indicator of how much percentage predictions made are correct. So the higher accuracy on both training and validation/testing set, shows the model is more accurate.

Precision, recall and F1 score are the metrics that evaluate the ratio of true positive and true negative we have in the prediction. They are especially useful if a dataset is imbalanced with respect to classes, i.e. where accuracy fails to properly assess the performance.

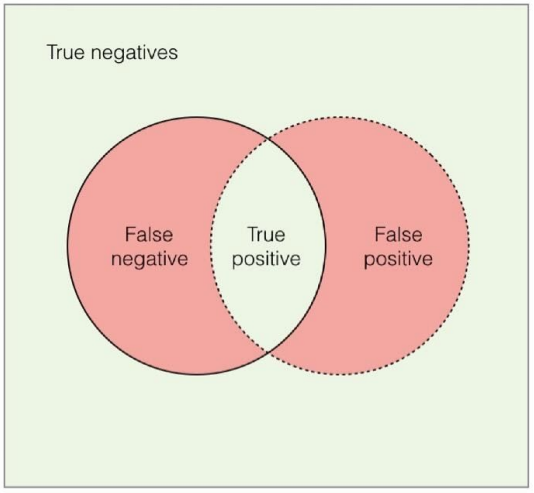
$$Precision = (True_{pos}) / (True_{pos} + False_{pos})$$

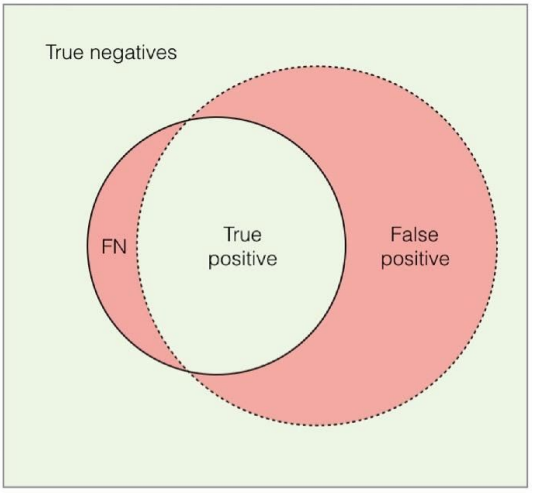
$$Recall = (True_{pos}) / (True_{pos} + False_{neg})$$

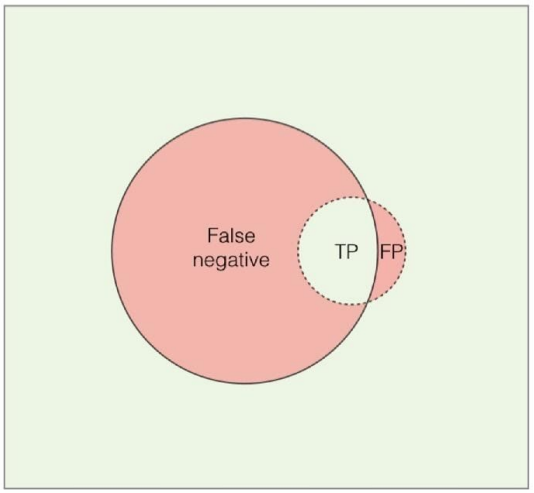
F1 score is a combining precision and recall (as a harmonic mean of them).

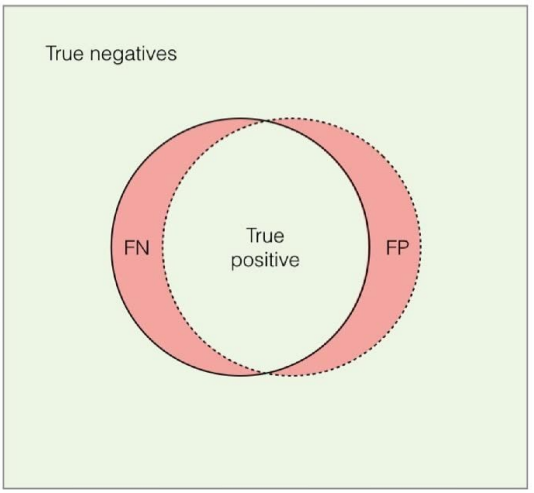
$$F1 - score = 2 * (Precision * Recall) / (Precision + Recall)$$

In our case, since sending the offer is rather low cost, and with high reward, we prefer to have a higher recall. A few scenarios of precision and recall as in the following picture.









2. Analysis (Data Explore & Visualization)

In this part, I will have a look at the imported data. Try to answer these questions:

- What are the dimensions of the datasets?
- What features does each of them have?
- What is the value range / value type of each feature?
- Is there a correlation between some features?
- Are the samples in the dataset equally distributed, or what's the distribution?

It is important to know the data that I'm dealing with well. By answering these questions, I will have an idea of what resources I can use to solve the problem, and further refine the strategy (Actually data exploration happens before I form a clear strategy).

2.1 Portfolio Dataset

Portfolio data saves the types of offers that have been dealt with during this time period. It is not long, so I print the whole dataframe. There are three major types of offers, 4 BOGO offers, 4 discount offers and 2 informational offers, sent through different channels. These offers vary in difficulty, reward and duration.

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

A description table below shows how difficulty, duration and reward vary between the offer types. The "bogo" offers have lower difficulty and higher reward, but shorter duration compared with the "discount" offers. Informational offer has no certain difficulty or reward, the duration is smaller than the other two kinds of offers.

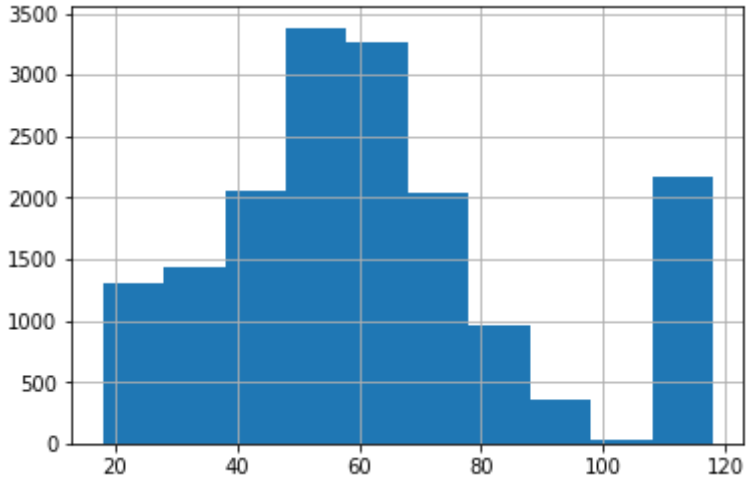
	difficulty						duration						reward					
	count	mean	std	min	50%	max	count	mean	std	min	50%	max	count	mean	std	min	50%	max
offer_type																		
bogo	4.0	7.50	2.886751	5.0	7.5	10.0	4.0	6.0	1.154701	5.0	6.0	7.0	4.0	7.5	2.886751	5.0	7.5	10.0
discount	4.0	11.75	5.678908	7.0	10.0	20.0	4.0	8.5	1.732051	7.0	8.5	10.0	4.0	3.0	1.414214	2.0	2.5	5.0
informational	2.0	0.00	0.000000	0.0	0.0	0.0	2.0	3.5	0.707107	3.0	3.5	4.0	2.0	0.0	0.000000	0.0	0.0	0.0

2.2 Profile Dataset

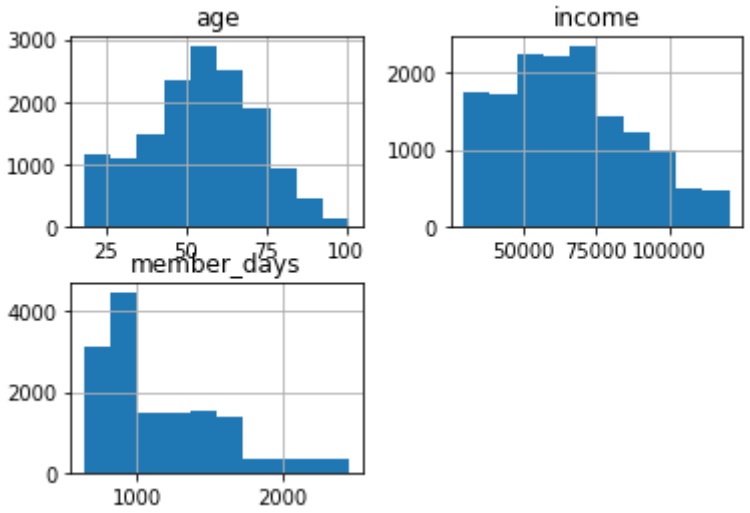
Profile dataset stores the demographic information of the customers. The table below shows the first five data samples in it. Notice that there are a few abnormalities (age 118) and some missing data.

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

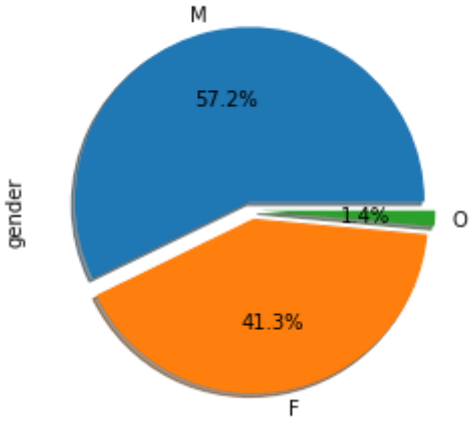
Plot the histogram of age distribution: there are 2k people who are 118 years old. All those people have no information about gender and income category. Due to lack of features I will drop those user informations.



After removing the abnormalities, plot again the distribution of these features. The distribution of age is most close to normalized distribution, the income and days since a member are all a bit skewed.



The gender pie chart shows that the male-female ratio is neutral (close to 50%), input data is balanced.



I draw a correlation chart to see if these features are related. The member_days are not related with age or income, age and income are a bit correlated, which is understandable, but the value is not very high (0.31).

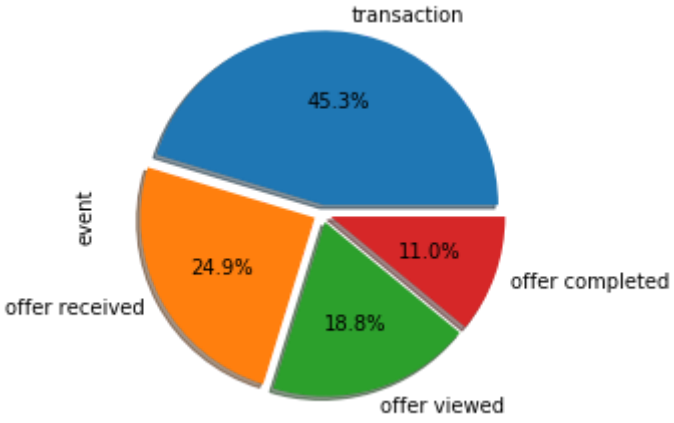
2.3 Transcript Dataset

Transcript dataset is a backend database that records the purchases a user made through a transaction, and updates the offer status. It has four columns: event, person, time, value.

Event column says the type of the record, it can be a transaction or something related with offer. There are four types of possible events: transaction, offer received, offer viewed, offer completed. Each event has a person id, shows the customer who is related, a time stamp, and a value. Value is in the dictionary datatypes, offer id, reward, amount of transactions can be packed inside.

	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'}

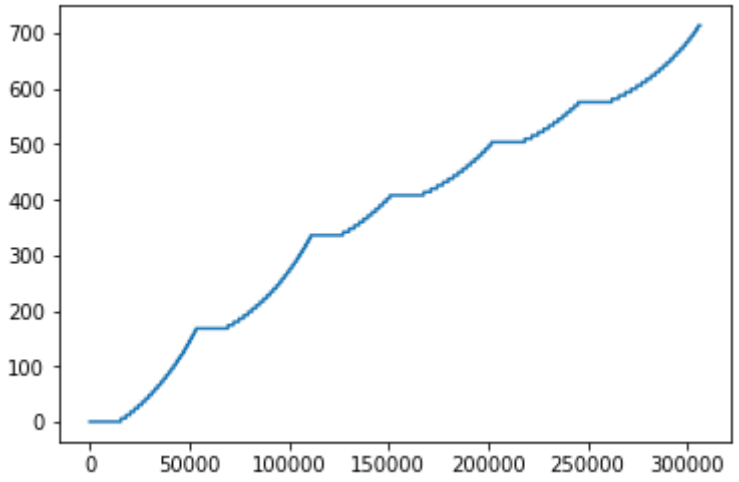
I draw the pie chart of the percentage of each event type. Almost half of the records are transactions (45%), 25% records are offer received, then 19% offer viewed, 11% offer completed. Only less than half of the offer that has been sent out is completed. Though, I suppose in real life it's already a perfect rate. Most of the transactions have nothing to do with the offers.



I'm also curious, how balanced is the data. I print out the amount of offer sent in each category. We can see that each offer id has been sent out around 7.5k times. It is balanced.

```
transcript[transcript['event'] == 'offer received']['value'].value_counts()
```

```
{'offer id': '9b98b8c7a33c4b65b9aebfe6a799e6d9'}    7677
{'offer id': '0b1e1539f2cc45b7b9fa7c272da2e1d7'}    7668
{'offer id': 'ae264e3637204a6fb9bb56bc8210ddfd'}    7658
{'offer id': '2298d6c36e964ae4a3e7e9706d1fb8c2'}    7646
{'offer id': '2906b810c7d4411798c6938adc9daaa5'}    7632
{'offer id': '5a8bc65990b245e5a138643cd4eb9837'}    7618
{'offer id': '3f207df678b143eea3cee63160fa8bed'}    7617
{'offer id': 'fafdcd668e3743c1bb461111dcafc2a4'}    7597
{'offer id': '4d5c57ea9a6940dd891ad53e9dbe8da0'}    7593
{'offer id': 'f19421c1d4aa40978ebb69ca19b0e20d'}    7571
Name: value, dtype: int64
```

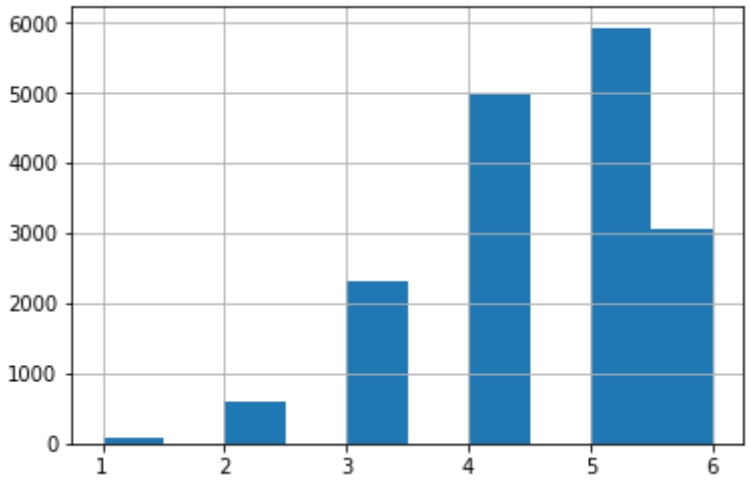


By plotting the column "time", it can be seen that the transcript dataset is organized according to the increasing time. While the maximum of time is 714 hours, it gives us the time window of 30 days (714/24) of this observation.

I want to have a look at how the company is sending the promotion infos, mainly the frequency, do they send it every hour, or every few days? Count the timestamp of each “offer received” event, we can see that Starbuck has sent 6 times the promotions. Each time around 12K sendings, but not in a constant period. I would like to use this sending event as a dataframe. It is also balanced.

	sent_time	sent_cnt
0	0	12650
1	168	12669
2	336	12711
3	408	12778
4	504	12704
5	576	12765

The bar plot below shows the counts of events each person gets over time being. Customers can get a maximum 6 times offer, and minimum 1 times (very rare).



At last, I want to look at the relationship between event, value and time for one customer. I picked a few customers randomly, these can represent common scenarios. By observing their records, it can represent how the system records one person’s behaviour.

	event	person	time	value
0	offer received	78afa995795e4d85b5d9ceeca43f5fef	0	{'offer id': '9b98b8c7a33c4b65b9aebfe6a799e6d9'}
15561	offer viewed	78afa995795e4d85b5d9ceeca43f5fef	6	{'offer id': '9b98b8c7a33c4b65b9aebfe6a799e6d9'}
47582	transaction	78afa995795e4d85b5d9ceeca43f5fef	132	{'amount': 19.89}
47583	offer completed	78afa995795e4d85b5d9ceeca43f5fef	132	{'offer_id': '9b98b8c7a33c4b65b9aebfe6a799e6d9...
49502	transaction	78afa995795e4d85b5d9ceeca43f5fef	144	{'amount': 17.78}
53176	offer received	78afa995795e4d85b5d9ceeca43f5fef	168	{'offer id': '5a8bc65990b245e5a138643cd4eb9837'}
85291	offer viewed	78afa995795e4d85b5d9ceeca43f5fef	216	{'offer id': '5a8bc65990b245e5a138643cd4eb9837'}
87134	transaction	78afa995795e4d85b5d9ceeca43f5fef	222	{'amount': 19.67}
92104	transaction	78afa995795e4d85b5d9ceeca43f5fef	240	{'amount': 29.72}
141566	transaction	78afa995795e4d85b5d9ceeca43f5fef	378	{'amount': 23.93}
150598	offer received	78afa995795e4d85b5d9ceeca43f5fef	408	{'offer id': 'ae264e3637204a6fb9bb56bc8210ddfd'}
163375	offer viewed	78afa995795e4d85b5d9ceeca43f5fef	408	{'offer id': 'ae264e3637204a6fb9bb56bc8210ddfd'}
201572	offer received	78afa995795e4d85b5d9ceeca43f5fef	504	{'offer id': 'f19421c1d4aa40978ebb69ca19b0e20d'}
218393	transaction	78afa995795e4d85b5d9ceeca43f5fef	510	{'amount': 21.72}
218394	offer completed	78afa995795e4d85b5d9ceeca43f5fef	510	{'offer_id': 'ae264e3637204a6fb9bb56bc8210ddfd...
218395	offer completed	78afa995795e4d85b5d9ceeca43f5fef	510	{'offer_id': 'f19421c1d4aa40978ebb69ca19b0e20d...
230412	transaction	78afa995795e4d85b5d9ceeca43f5fef	534	{'amount': 26.56}
262138	offer viewed	78afa995795e4d85b5d9ceeca43f5fef	582	{'offer id': 'f19421c1d4aa40978ebb69ca19b0e20d'}

3. Methodology

3.1 Data Preprocessing

First I need to work on three datasets, then merge them into one.

1. make a copy of each dataset, then work on the copy.
2. profile:
    - change date to an integer (days of being a member),
    - drop NaNs,
    - one hot encode gender.
3. transcript:
    - splite value dict to: offer id, amount,
    - drop value column.
4. one hot encode channels
5. merge transcript_clean with profile_clean, then further merge with portfolio
6. split transaction data and offer data.
7. split the one merged data frame to three small data frames according to offer types.

3.2 Implementation

First need to label the data correctly, based on the strategy explained in chapter 1.2. One tricky part to choose to filter in implementation of this strategy is to take the “offer viewed” as a mask. This filters out all the cases where the offer has never been seen. Then implement one function to exclude the case when the offer has been viewed, but after the offer is completed. Because in this scenario, the offer also didn’t influence the decision of the customer, so I will filter it out.

By implementation, df.shift() is always used, to compare the rows next to each other. It saves a lot of time compared with iterating rows. This is my major code to exclude the “first completed then viewed” senario:

req1 = req1 = df.shift(1).loc[mask_viewed]['event'] == 'offer completed'

main line to label bogo and discount dataset:

req1 = df.shift(-1).loc[mask_viewed]['event'] == 'offer completed'

To label informational data is a bit different. I first removed all the transactions due to BOGO and discount offer completion, then checked in the window of receiving informational offers, if there are transactions from the same user. If there are transactions, mark as True, else False.

The learning implementation consists of three main functions:

prepare_input(df):

- prepare X, y.
- drop non numeric columns,
- change the labels from True/False to 1/0.
- do random train_test_split (random seed 42),
- standardize train test set with StandardScaler().
- return X_train, X_test, y_train, y_test.

train_func(df):

- init model with model from sklearn; here i choose lre, tree and rf.
- I initialize lre with LogisticRegression(), limited max_iter=100;
- for decision tree: criterion='entropy', max_depth = 6, min_samples_split = 90, min_samples_leaf = 50, random_state = 10;
- for rf: n_estimators = 20, criterion='entropy', max_depth = 11, max_features= 'auto', min_samples_split = 10, min_samples_leaf = 20, random_state = 2

- get input data prepared for training, send to function train_evaluate_model, get and return results metrics and models.

train_evaluate_model(model, X_train, y_train, X_test, y_test):

- train model with scikit learn model.fit
- get & return metrics (train accuracy, test accuracy, precision, recall, f1-score.

3.3 Refinement

To do a bit of refinement of the model, I look back at the data, see if I can add some more features to train/test set. Till now I only used features that have been provided, age, income, time since a member as the feature of the customer.

I try to generate some features from the transcript record. But one thing has to be kept in mind: since the training and test data are also produced from the same dataset, I should avoid to include those information that's related to offer completion to features.

Finally i added two features:

- the frequency of transactions been made during this time being,
- the total amount of transactions.

These two features are totally independent from the offer received or completed, so they are less possible to cause an overfitting problem.

4. Results

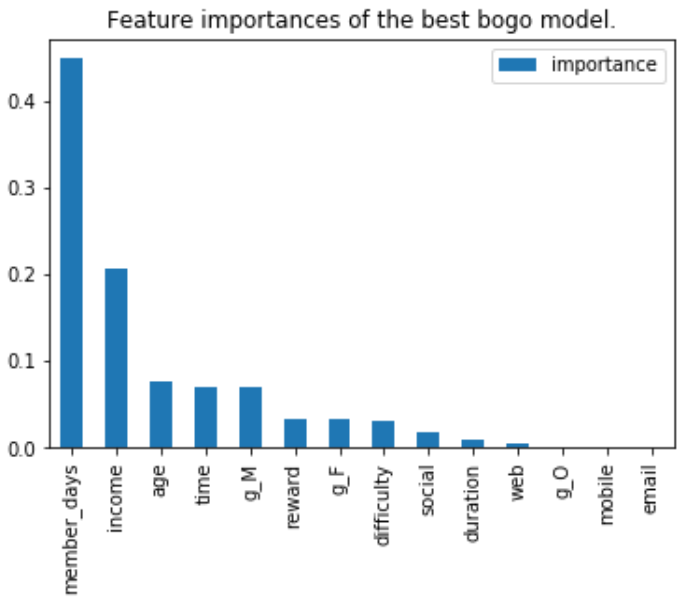
4.1 Model Evaluation and Validation

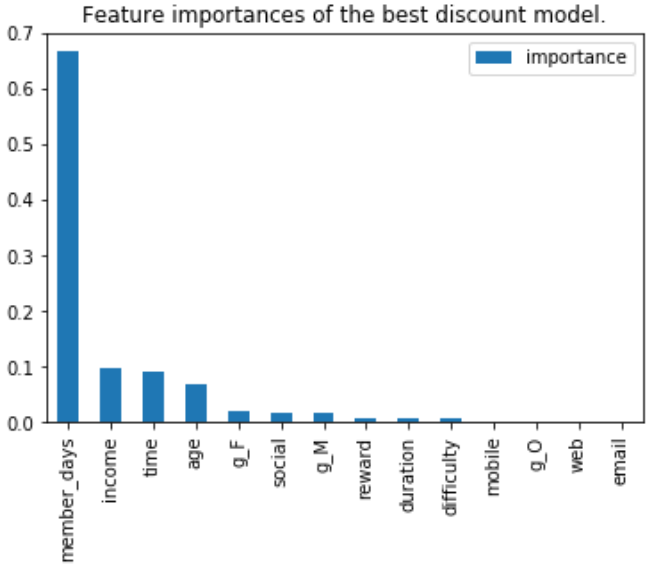
The results of the first try of original features (in the order: BOGO, discount, informational):

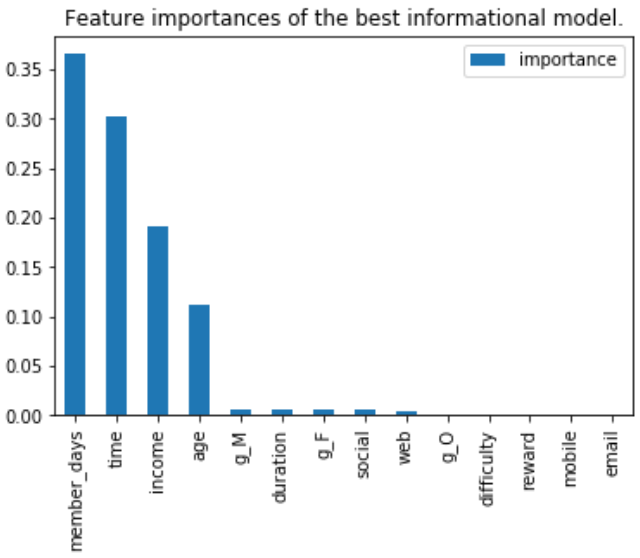
	lre	tree	rf	lre	tree	rf	lre	tree	rf
accuracy_train	0.666350	0.725047	0.753004	0.736886	0.748191	0.779696	0.606838	0.669872	0.713942
accuracy_test	0.665570	0.712370	0.719707	0.730479	0.739524	0.740730	0.625000	0.650641	0.662393
precision	0.664999	0.711626	0.719058	0.680304	0.689063	0.690043	0.625204	0.651824	0.662228
recall	0.662769	0.711148	0.718284	0.602503	0.633326	0.638486	0.625245	0.651403	0.661855
f1score	0.662913	0.711324	0.718535	0.606668	0.643575	0.649101	0.624993	0.650539	0.661893

The training accuracy and test accuracy are at the similar level, no overfitting.
The highest accuracy is achieved by random forest.

Plot the feature_importance from random forest,


Feature importances of the best bogo model.
importance
0.4
0.3
0.2
0.1
0.0
member_days
income
age
time
g_M
reward
g_F
difficulty
social
duration
web
g_O
mobile
email


Feature importances of the best discount model.
importance
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
member_days
income
time
age
g_F
social
g_M
reward
duration
difficulty
mobile
g_O
web
email



The most important feature is the number of days since becoming a member, it is much more important in bogo and discount. Then comes income, age. It's interesting to see the time of viewing the offer is in all three cases more important than gender,

After refinement and added two features, the result got better, after add the frequency of transaction, the accuracy of BOGO and discount increased around 5%, but for informational did not increase:

	lre	tree	rf
accuracy_train	0.730108	0.765022	0.796964
accuracy_test	0.728814	0.756641	0.768024
precision	0.727886	0.757690	0.767362
recall	0.724521	0.751280	0.764637
f1score	0.725396	0.752635	0.765526

lre	tree	rf
0.825068	0.831776	0.843458
0.805547	0.817606	0.817305
0.772726	0.784334	0.787602
0.740921	0.766893	0.757238
0.753168	0.774518	0.769307

lre	tree	rf
0.666800	0.697249	0.733841
0.646368	0.642094	0.666132
0.648059	0.642619	0.665594
0.647931	0.642862	0.665795
0.646361	0.642035	0.665652

Further add the other feature (total amount of transaction) on top of this. The accuracy and F1 further increased for BOGO and discount, but still has no obvious increase on informational offer (increased 2%):

	lre	tree	rf
accuracy_train	0.804048	0.831626	0.844782
accuracy_test	0.812041	0.824184	0.826208
precision	0.811393	0.825261	0.827560
recall	0.813214	0.820506	0.822382
f1score	0.811588	0.821979	0.823947

lre	tree	rf
0.842780	0.850995	0.863883
0.829967	0.826952	0.833585
0.799293	0.810927	0.814174
0.783934	0.754059	0.769686
0.790816	0.773358	0.786244

lre	tree	rf
0.668937	0.697115	0.749332
0.647436	0.655449	0.682692
0.648727	0.660232	0.682068
0.648756	0.658433	0.681087
0.647435	0.655033	0.681294

4.2 Justification

In the implementation, I compared three machine learning models: logistic regression, decision tree, random forest. Their performance ranks as follows: decision tree is better than logistic regression, random forest is better than decision tree.

The reason that random forest predicts better than tree and logistic regression is intuitive. Random forest is a boosting of trees, and as such an ensemble model outperforms single weak learners that it consists of.

The result of the original training was around 70%. By applying some feature engineering, the accuracy and F1 Score increased by 10% by BOGO and discount offer, while by informational offer didn't improve obviously. Probably these are not the right features for an informational offer.

The feature "days of being a member" has the highest feature importance. It suggests the most information gain (setting random forest criterion=entropy). The opposite case is channel email. It is true in every offer, that means, in training data, it is always true. It ranks the last at all the three offer types. If I was to remove features, those least important would be the first to remove.

5. Conclusion

5.1 Reflection

I enjoyed the process of solving this problem. When I first saw it, I thought it's easy to tackle. The strategy to solution, how to define a successful promotion has formed very fast.

But it got hard when it came to implement data preprocessing in pandas, with all the data. I'm not a specialist in pandas. In the beginning I always felt like writing a numpy array or query similar as in SQL, finally I tried a for loop to iterate over 20k or more rows of dataframe, which runs forever. Later I found out how to effectively filter data by mask, and how to use shift to compare, that makes things easy.

5.2 Improvement

I didn't tune the parameters a lot in this project, partly because somehow I have the opinion that features are more important than parameters. If a model is not accurate enough (underfitting), it is better to try to create more features, than focus on parameter tuning. However in furthers steps, tuning might bring some more improvements.

But one thing I should improve later is recall. Now the recall is similar with the precision. it means my model didn't have a good parameter to get the higher recall on cost of precision.