Starbucks Project - Predicting whether users will see the offer or not

Capstone Project - Udacity Data Scientist Nanodegree

Heegyu Kim
October 24th, 2020

Project Definition

Overview

The reward program is a representative marketing method that induces users to purchase. But programs that users are not interested in, users may unsubscribe corporate marketing channels or have a negative image. That's why it's important to send an SMS, email or reward program to interested users. In this case, the user's interest is determined by whether or not the user has viewed the program. By creating a model that predicts whether users will see the program, you can deliver the right program to the right users.

Statement

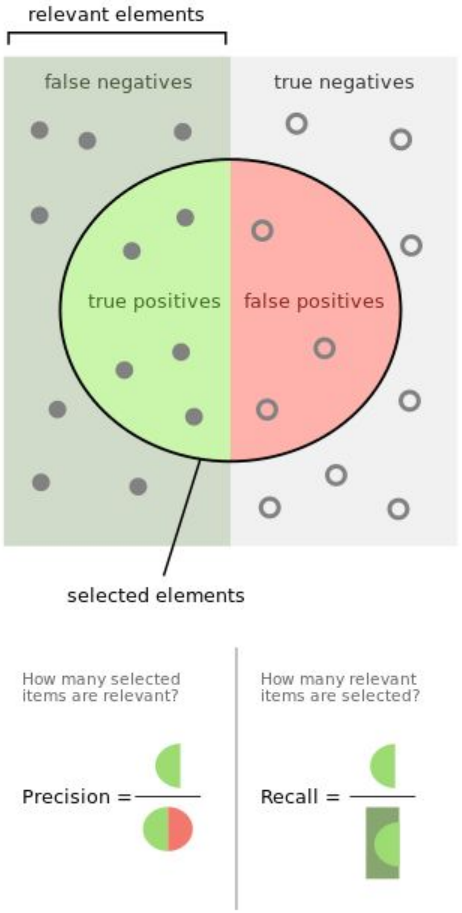
The goal is create a model to predict whether users will see the offer or not. The tasks involved are the following.

1. Download and preproces the "Starbucks app customer rewards program data" from https://www.kaggle.com/blacktile/starbucks-app-customer-reward-program-data
2. Train a classifier that can predict whether users will see the offer or not (binary classification) by using data whether past users have seen or not seen
3. Validate the model whether it can predict well for future data which was not used for training

Metrics

Accuracy is a common metric for almost all classification models. In unbalanced data, However, F1 score is a widely used metric. The F1 score is the harmonic mean of the precision and recall. You can see the definition in the image below. It is a better indication of accuracy in unbalanced data

$$\text{Accuracy} = \frac{tp + tn}{tp + tn + fp + fn}$$



$$F_1 = \frac{2}{\mathrm{recall}^{-1} + \mathrm{precision}^{-1}} = 2 \cdot \frac{\mathrm{precision} \cdot \mathrm{recall}}{\mathrm{precision} + \mathrm{recall}} = \frac{\mathrm{tp}}{\mathrm{tp} + \frac{1}{2}(\mathrm{fp} + \mathrm{fn})}.$$

Image From: https://en.wikipedia.org/wiki/F-score

Analysis

Data Exploration

Portfolio.json

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

- Contains information on 10 offers that users can receive.
- There are 3 types of programs— BOGO(Buy-One, Get-One), discount, informational.
- Informational offer don't have reward.
- Offers are delivered through multiple channels

Profile.json

	gender	age	id	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
...	...	...	...	...	...
16995	F	45	6d5f3a774f3d4714ab0c092238f3a1d7	20180604	54000.0
16996	M	61	2cb4f97358b841b9a9773a7aa05a9d77	20180713	72000.0
16997	M	49	01d26f638c274aa0b965d24cefe3183f	20170126	73000.0
16998	F	83	9dc1421481194dcd9400aec7c9ae6366	20160307	50000.0
16999	F	62	e4052622e5ba45a8b96b59aba68cf068	20170722	82000.0

17000 rows × 5 columns

- Profile is the information of users.
- Users of unknown age are 118 years old
- User's gender, age, income are all unknown or all exist.

- 2175 of users(12%) have no age, gender, income data(NaN). They only have event history.

Transcript.json

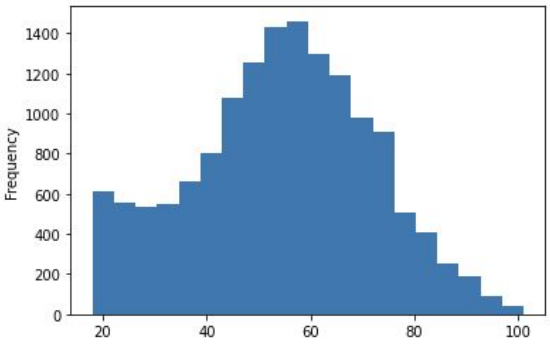
	person	event	value	time
0	78afa995795e4d85b5d9ceeca43f5fef	offer received	{'offer id': '9b98b8c7a33c4b65b9aebfe6a799e6d9'}	0
1	a03223e636434f42ac4c3df47e8bac43	offer received	{'offer id': '0b1e1539f2cc45b7b9fa7c272da2e1d7'}	0
2	e2127556f4f64592b11af22de27a7932	offer received	{'offer id': '2906b810c7d4411798c6938adc9daaa5'}	0
3	8ec6ce2a7e7949b1bf142def7d0e0586	offer received	{'offer id': 'fafdcd668e3743c1bb461111dcafc2a4'}	0
4	68617ca6246f4fbc85e91a2a49552598	offer received	{'offer id': '4d5c57ea9a6940dd891ad53e9dbe8da0'}	0
...	...	...	...	...
306529	b3a1272bc9904337b331bf348c3e8c17	transaction	{'amount': 1.5899999999999999}	714
306530	68213b08d99a4ae1b0dcb72aebd9aa35	transaction	{'amount': 9.53}	714
306531	a00058cf10334a308c68e7631c529907	transaction	{'amount': 3.61}	714
306532	76ddbd6576844afe811f1a3c0fbb5bec	transaction	{'amount': 3.5300000000000002}	714
306533	c02b10e8752c4d8e9b73f918558531f7	transaction	{'amount': 4.05}	714

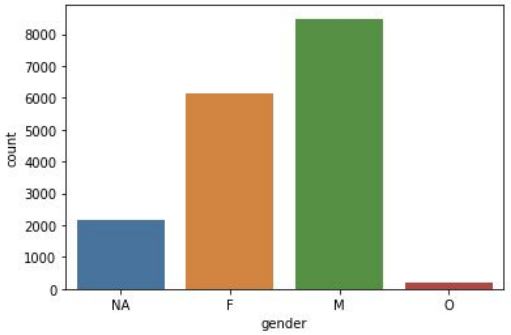
306534 rows × 4 columns

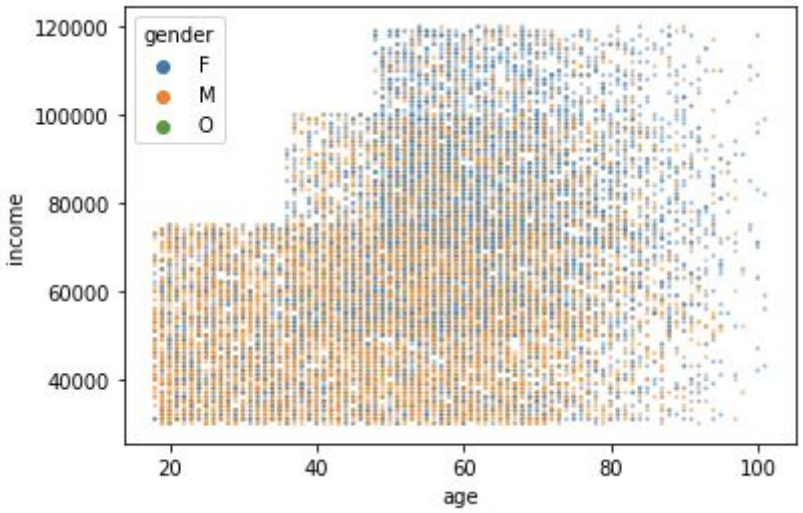
- Transcript are events in which a user received, viewed, completed, or transacted an offer. person column represents the user's ID in profile.
- Completed offer events have reward gain.
- Informational offers don't have complete events.

Data Visualization

User profile

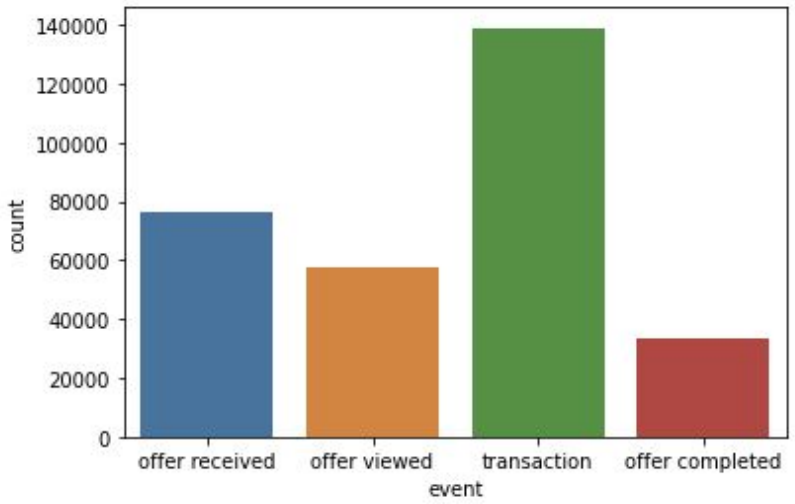




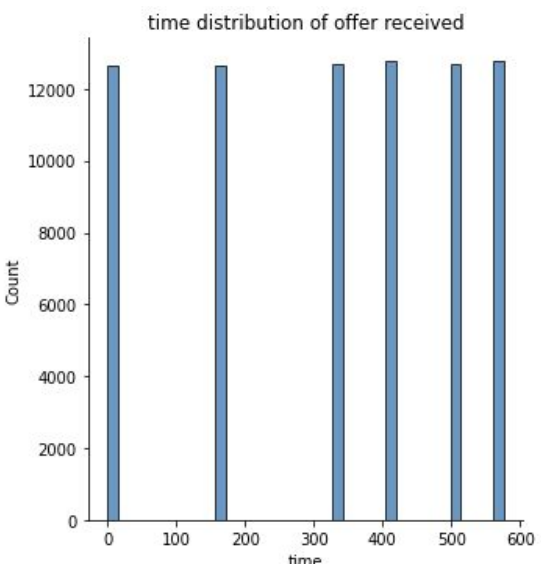


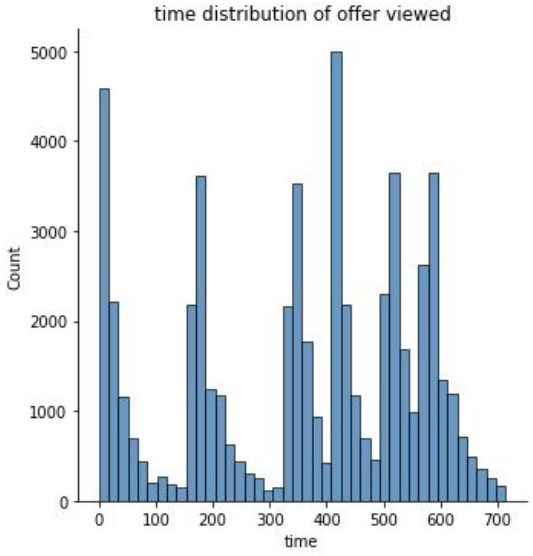
Most users are between 50-70 years old. Distribution of income by age was similar regardless of gender

Transcript

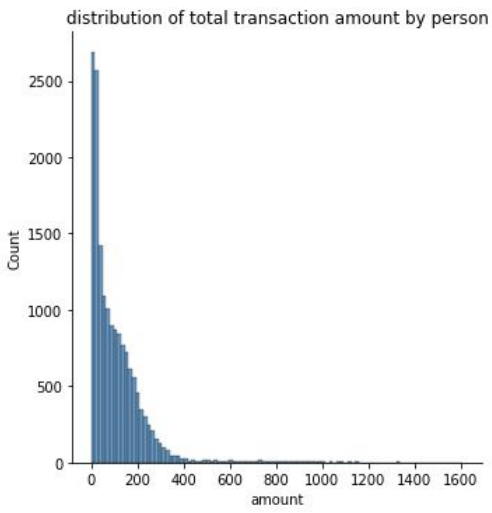


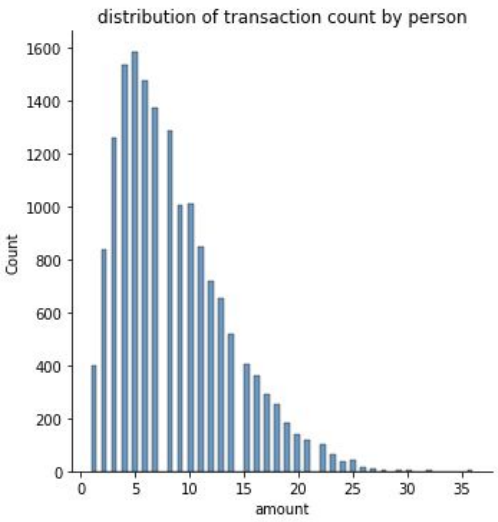
count by event type





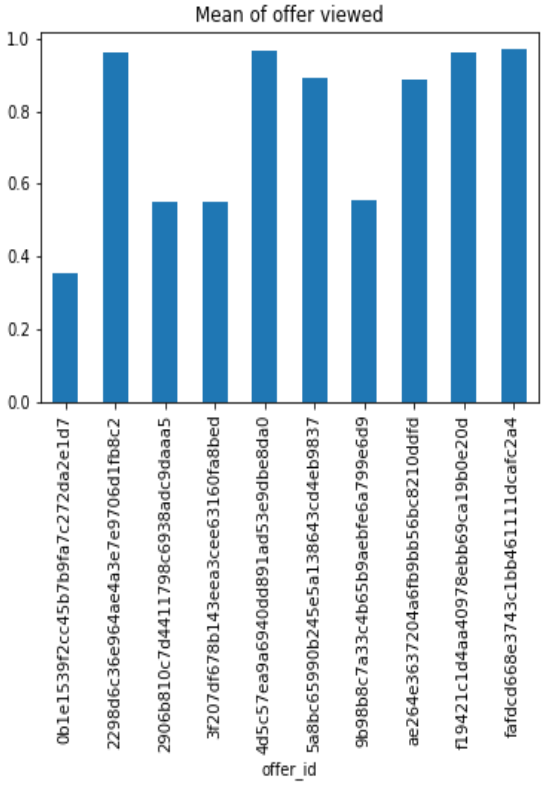
Above plot is histogram of offer received times. As you see that, User received an offer at a specific time(0, 168, 336, 408, 504, 576, unit is hour, initial value is zero). Offer views happened a lot right after receiving an offer. So I decided that events with a time value between 0 ~ 503 are train set, events with a time value more than 503 are test set. It means that we predict future user action from past history.

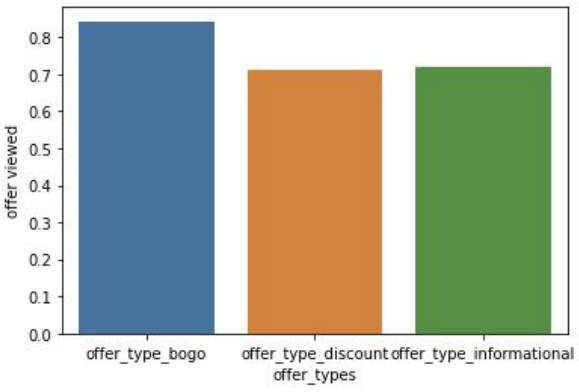




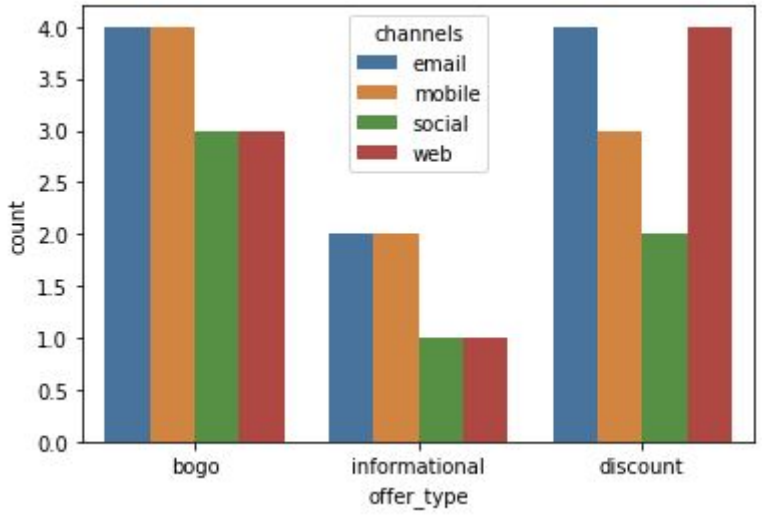
A lot of 0-amount transactions have occurred, but I need to check the exact reason. It may have purchased it for free using a coupon or event.

Portfolio

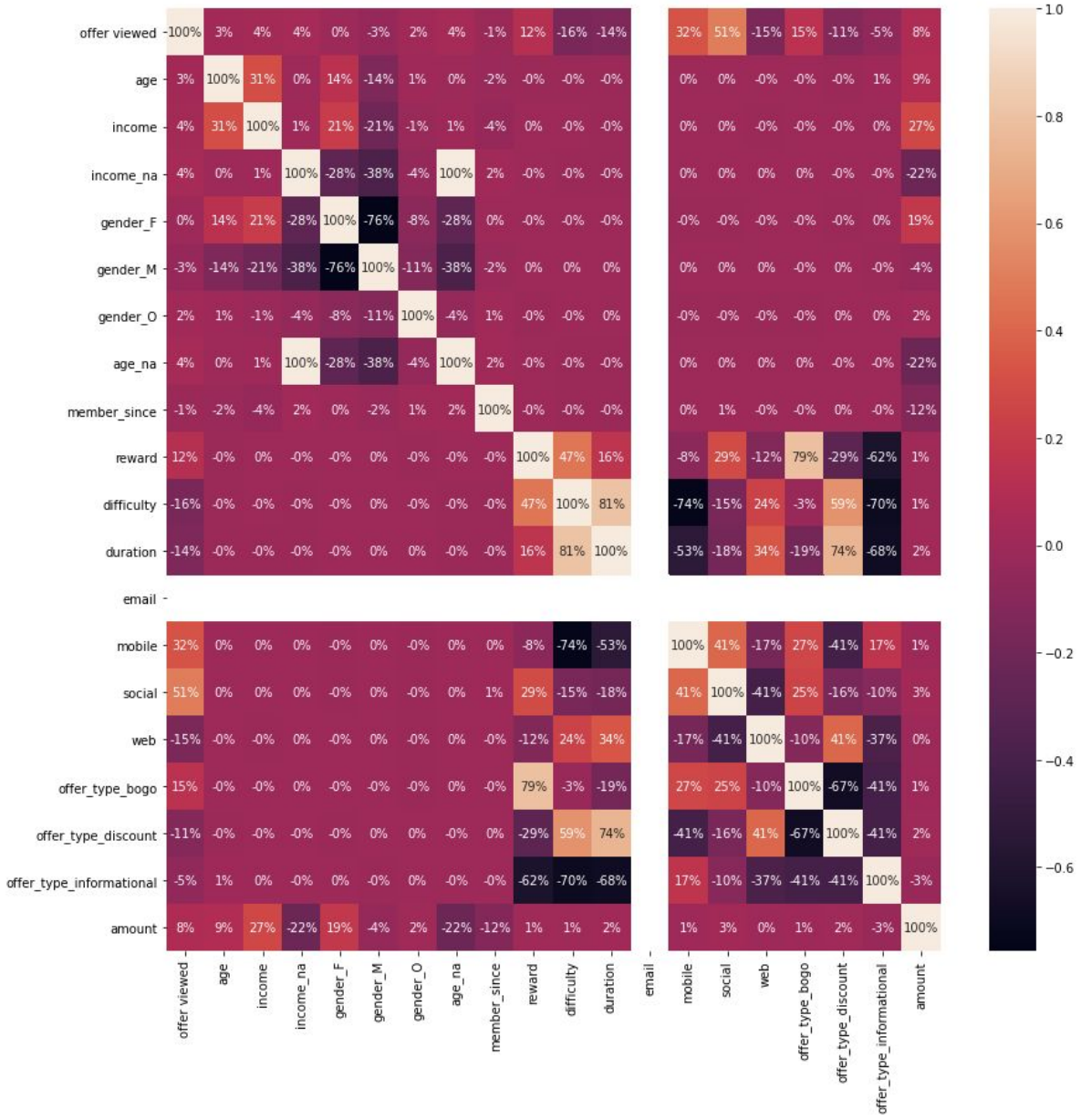




On average, the view probability was slightly higher in bogo, but when viewed by offer, there was a big difference even in the bogo offer type.



All offers are delivered through an email channel.



I combined user's information and offer that user received and whether user viewed or not. And plot heatmap of correlation matrix. mobile, social channels have a positive relationship with offers viewed. This was a surprising analysis. Reward have little positive relationships and difficulty, duration have little negative relationships. This was an easily predictable result, but it was less influential than expected.

Methodology

Data Preprocessing

Handle Missing Values

id	age	income	not_na	gender_F	gender_M	gender_O	member_since
68be06ca386d4c31939f3a4f0e3dd783	54.393524	65404.991568	0	0	0	0	2017
0610b486422d4921ae7d2bf64640c50b	55.000000	112000.000000	1	1	0	0	2017
38fe809add3b4fcf9315a9694bb96ff5	54.393524	65404.991568	0	0	0	0	2018
78afa995795e4d85b5d9ceeca43f5fef	75.000000	100000.000000	1	1	0	0	2017
a03223e636434f42ac4c3df47e8bac43	54.393524	65404.991568	0	0	0	0	2017
...	...	...	...	...	...	...	...
6d5f3a774f3d4714ab0c092238f3a1d7	45.000000	54000.000000	1	1	0	0	2018
2cb4f97358b841b9a9773a7aa05a9d77	61.000000	72000.000000	1	0	1	0	2018
01d26f638c274aa0b965d24cefe3183f	49.000000	73000.000000	1	0	1	0	2017
9dc1421481194dcd9400aec7c9ae6366	83.000000	50000.000000	1	1	0	0	2016
e4052622e5ba45a8b96b59aba68cf068	62.000000	82000.000000	1	1	0	0	2017

17000 rows × 7 columns

If the user's age and income are missing values, they are replaced with average values. Also, if age, income, and gender are missing values, not_na column is added—0 is na, and 0 is not.

Categorical to numeric

User	Channel
A	Email, Social
B	Email, SMS
C	Social

User	Email	Social	SMS
A	1	1	0
B	1	0	1
C	0	1	0

For categorical variables, a column designated as 1 when it occurs and 0 when it is not added was added. Channels in the portfolio can have multiple values.

id	age	income	not_na	gender_F	gender_M	gender_O	member_since
68be06ca386d4c31939f3a4f0e3dd783	54.393524	65404.991568	0	0	0	0	2017
0610b486422d4921ae7d2bf64640c50b	55.000000	112000.000000	1	1	0	0	2017
38fe809add3b4fcf9315a9694bb96ff5	54.393524	65404.991568	0	0	0	0	2018
78afa995795e4d85b5d9ceeca43f5fef	75.000000	100000.000000	1	1	0	0	2017
a03223e636434f42ac4c3df47e8bac43	54.393524	65404.991568	0	0	0	0	2017
...	...	...	...	...	...	...	...
6d5f3a774f3d4714ab0c092238f3a1d7	45.000000	54000.000000	1	1	0	0	2018
2cb4f97358b841b9a9773a7aa05a9d77	61.000000	72000.000000	1	0	1	0	2018
01d26f638c274aa0b965d24cefe3183f	49.000000	73000.000000	1	0	1	0	2017
9dc1421481194dcd9400aec7c9ae6366	83.000000	50000.000000	1	1	0	0	2016
e4052622e5ba45a8b96b59aba68cf068	62.000000	82000.000000	1	1	0	0	2017

17000 rows × 7 columns

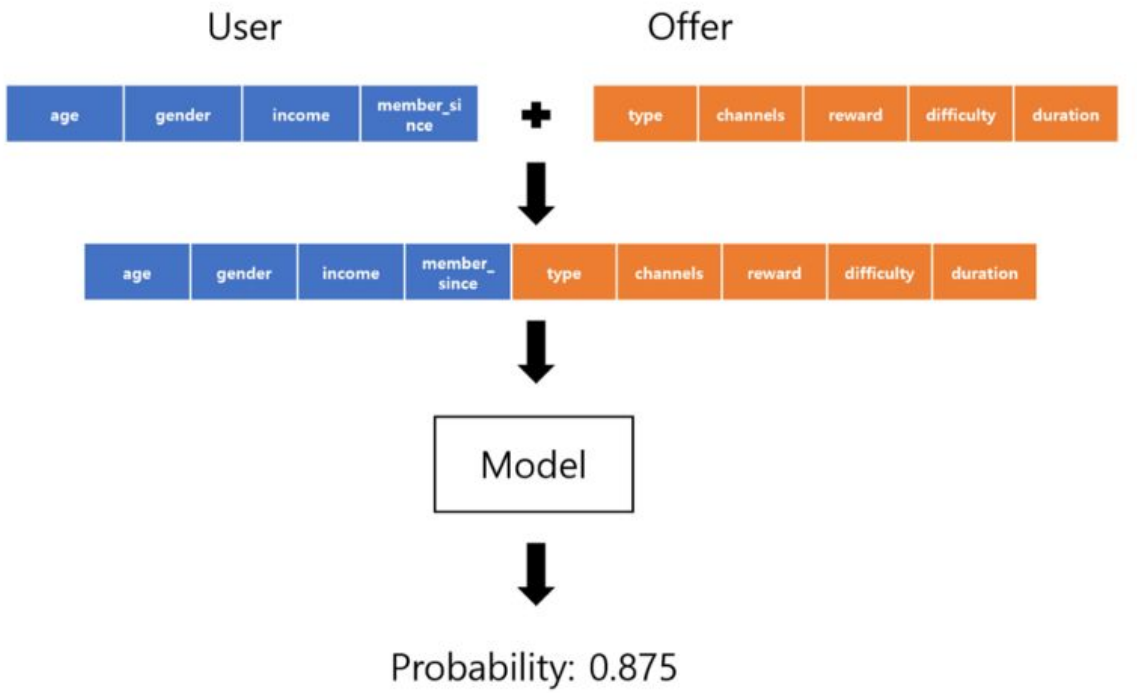
encoded profile data

Object to numeric

became_member is date format string(YYYYMMDD). I extracted the year and converted it to numeric value.

Create input data to use for prediction and output data.

Combine the offers received by the user and the user, and add an offer viewed column, which is 1 if the user viewed the offer, or 0 otherwise. Some users receive multiple events or view multiple times. To simplify the model, I omitted multiple events and set to 1.



The model receives user information and offer information received by the user and outputs the probability of viewing the offer received by the user.

Train/Test Split

Offer received/viewed events with a time value between 0 ~ 503 are train set, more than 503 are test set.

Normalization

Since the model values have different ranges, Scikit-Learn's StandardScaler unifies the distribution of values.

Benchmark

I couldn't find a similar project on the internet. Instead, I implemented a baseline model using RandomForestClassifier. In this baseline model, I found a model that maximizes the accuracy and the F1 score of the test dataset. Here's the performance of the baseline model. You can see that model is strongly overfitted.

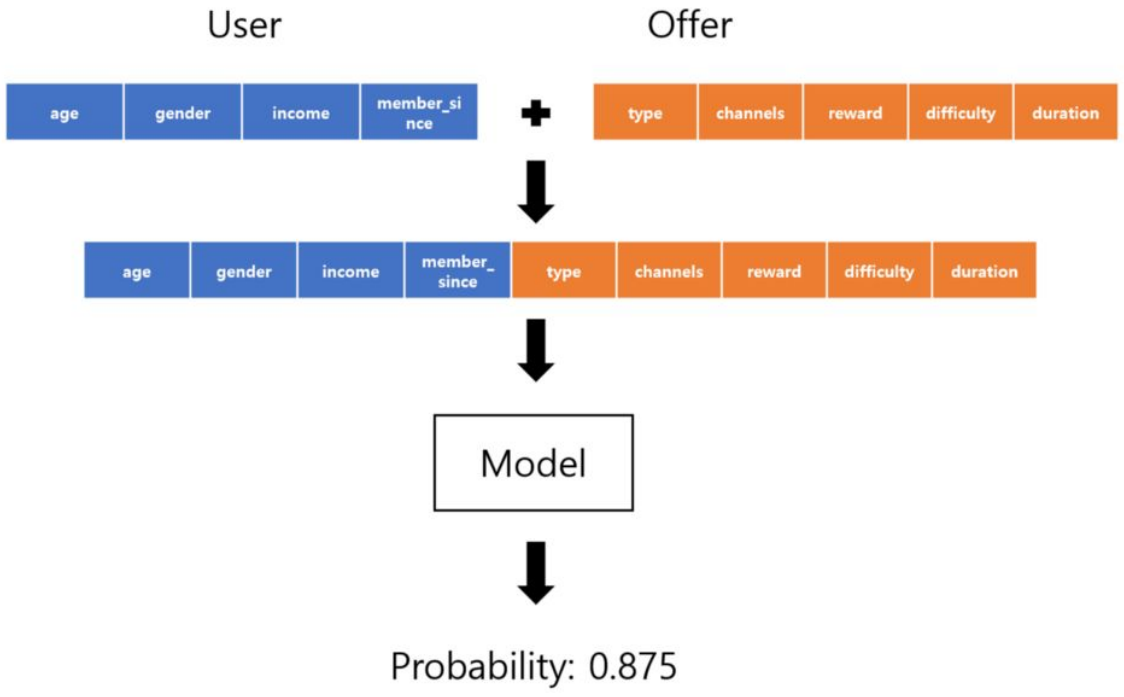
Report for baseline
Train Report>
precision recall f1-score support

0.0	0.96	0.89	0.92	10469
1.0	0.97	0.99	0.98	34906
accuracy			0.97	45375
macro avg	0.96	0.94	0.95	45375
weighted avg	0.97	0.97	0.96	45375

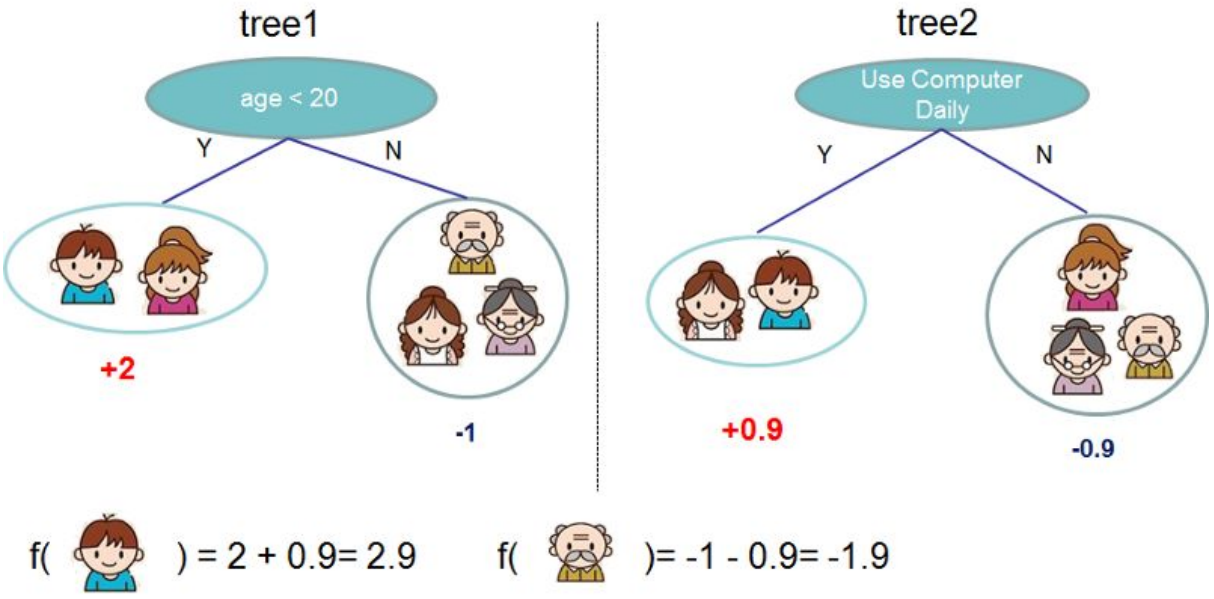
Test Report>

	precision	recall	f1-score	support
0.0	0.62	0.50	0.56	6102
1.0	0.85	0.90	0.87	18419
accuracy			0.80	24521
macro avg	0.73	0.70	0.71	24521
weighted avg	0.79	0.80	0.79	24521

Implementation

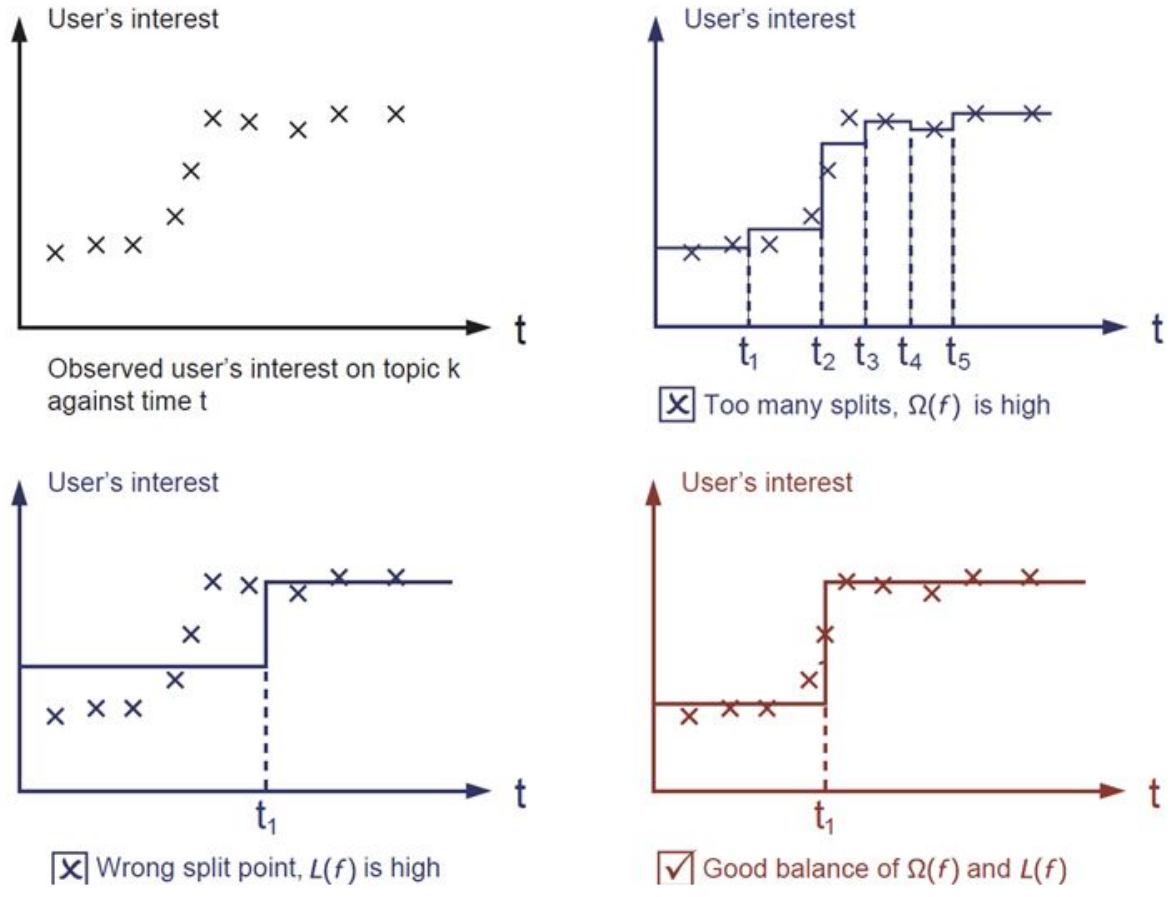


I used Scikit-Learn's StandardScaler for normalization, XGBoost as classifier.



XGBoost model from https://xgboost.readthedocs.io/en/latest/tutorials/model.html

XGBoost is a representative ensemble boosting model. Several decision trees conduct classification in parallel. First, the sampled values are put in tree 1, and then incorrectly predicted values are trained by weighting the tree 2. Repeat this over and over again for several trees.

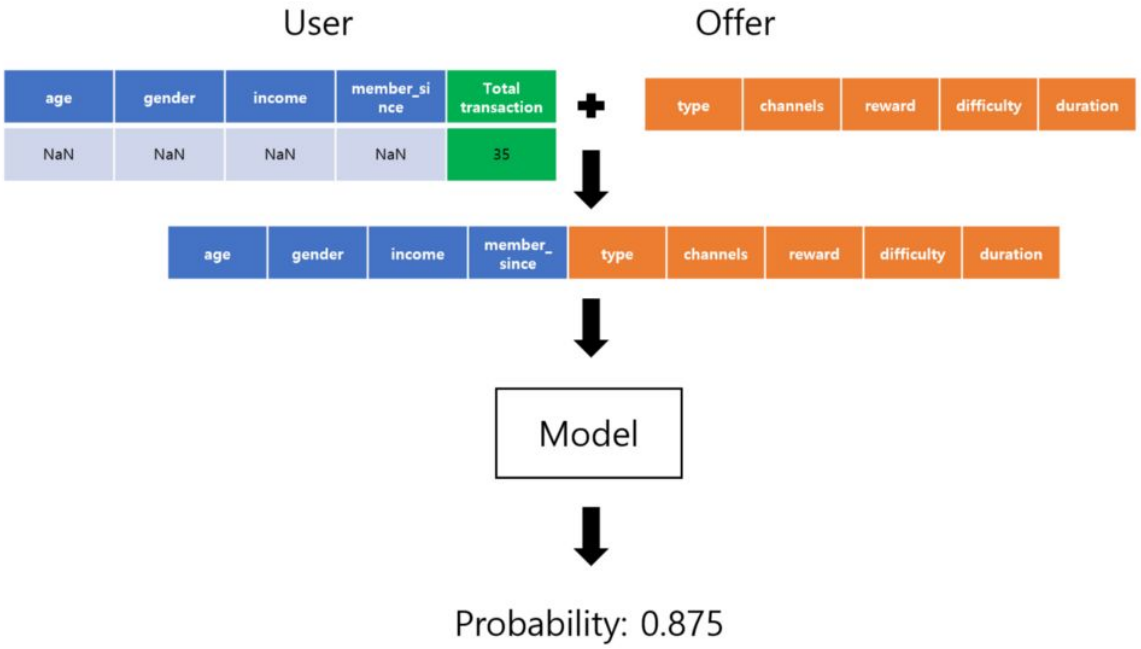


XGBoost Regularization from https://xgboost.readthedocs.io/en/latest/tutorials/model.html

Regularization is also used in XGBoost. It controls the complexity of the model and prevent the model to depend on specific features. **This method greatly reduces overfitting and enables parallel learning, which speeds up learning.**

Refinement

- The optimal hyperparameter was searched through GridSearchCV. In addition, 10-Fold Cross Validation was used to verify whether untrained data was well predicted.
- The log value was taken in the amount column and changed to a normal distribution.
- I inserted the user's total transaction price to the user's information. Because the model couldn't decide the probability for 12% of the users who have NaN value in age, income, gender. Their input was just average values and program's information!



- Removed unnecessary columns. Because all offers are delivered by email, the email column is always 1. member_since column is also removed because it had very little influence as a result of correlation analysis and feature importance analysis. Reducing the number of features is important to avoid overfitting.

Results

Model Evaluation and Validation

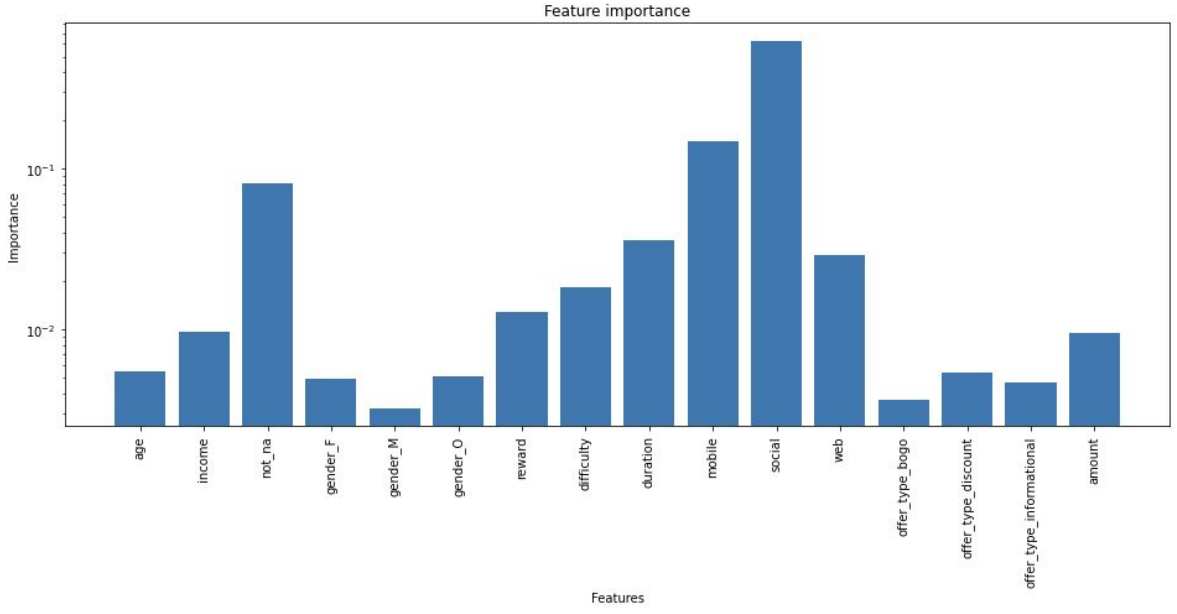
Dataset	Metric	Score

Train	Accuracy	0.85
	F1	0.90
Test	Accuracy	0.83
	F1	0.89

- The difference from the test set is not large and there is a little overfitting.
- Compared to the baseline model, both Accuracy and F1 score showed slight performance improvement.

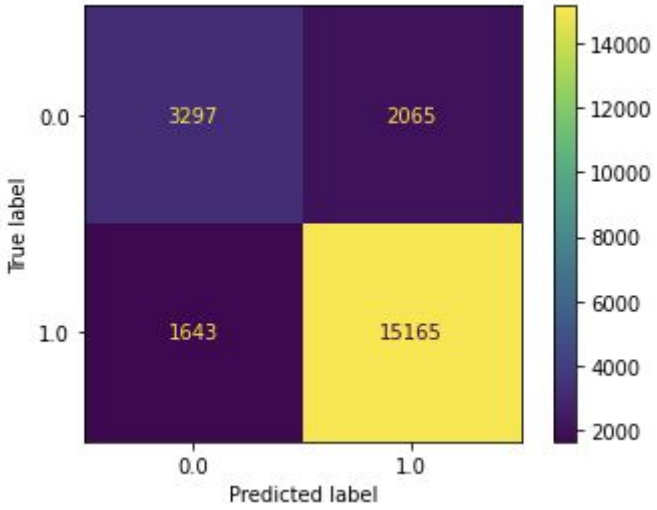
Justification

- I compared the distribution of the data that the model corrects and the data that are not correct. However, no significant difference was found in distribution.

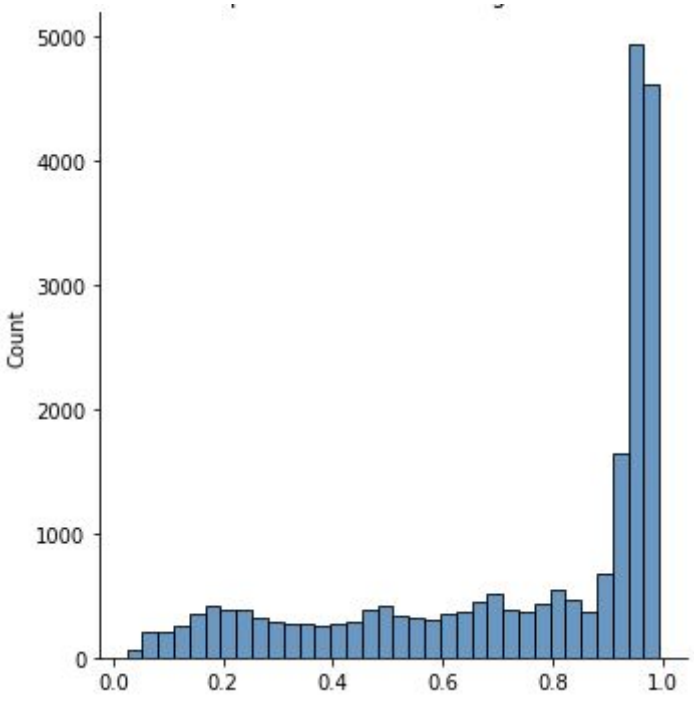


- Total transaction amounts have good feature importance. Sending through social media had an overwhelmingly large influence.

Confusion Matrix



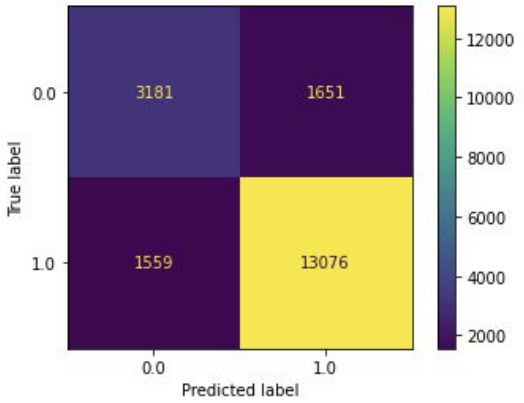
My model is good at predicting who to view, but is relatively poor at predicting who to not view. Because of the unbalanced data, predicting “not viewed” is more difficult.



You can see that most of the predicted values are near 1.0 and few near 0.0. This means that my model is difficult to predict “not viewed”.

Not NA User vs NA User

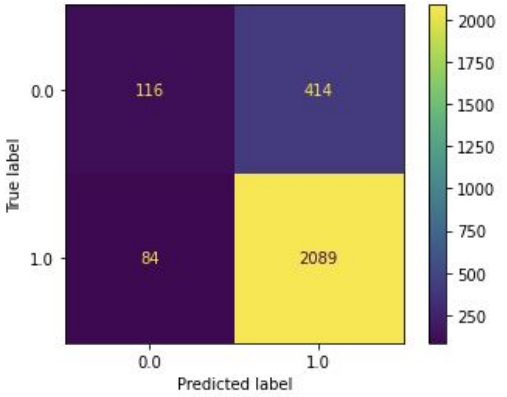
Confusion matrix for not_na



Accuracy: 0.8906750221374565

	precision	recall	f1-score	support
0.0	0.67	0.66	0.66	4832
1.0	0.89	0.89	0.89	14635
accuracy			0.84	19467
macro avg	0.78	0.78	0.78	19467
weighted avg	0.83	0.84	0.83	19467

Confusion matrix for na



Accuracy: 0.8934987168520103

	precision	recall	f1-score	support
0.0	0.58	0.22	0.32	530
1.0	0.83	0.96	0.89	2173
accuracy			0.82	2703
macro avg	0.71	0.59	0.61	2703
weighted avg	0.78	0.82	0.78	2703

Although there was no significant difference in the accuracy of users without information and users with no information, the accuracy of predicting that users without information will not see an offer was significantly low (F1 score 0.32).

Model incorrectly predicted about 500(18%) people who don’t have age, income, gender value, which was far less than my prediction. But, this is a high percentage given that they account for 12%.

In order to avoid sending wrong offers, it would be helpful to handle users who do not have information separately(ex: special another model) when building a recommendation system that sends an actual offer.

how improvements were made

Feature Engineering

- Inserting the user's total transaction price improves 1.2%p of test accuracy. It can help to predict the users without information
- When I took a log of the total transaction amount and converted it to a normal distribution, I received a very small increase in test accuracy (F1 score 0.3%p).
- Removing useless/a little influential columns didn't affect prediction.

Changing prediction threshold

Changing prediction threshold value to 0.4(predictIf the probability is greater than 0.4, it is 1, and if it is less, it is 0) shows best accuracy and f1 score both. Do not show a very big difference

	accuracy	f1 score
0.2	0.805323	0.884685
0.3	0.828101	0.894738
0.4	0.833153	0.895120
0.5	0.832747	0.891063
0.6	0.823230	0.880929
0.7	0.796888	0.856952
0.8	0.763013	0.824972

Hyperparameters Optimization

```
{
'xgb__colsample_bylevel': 0.9,
'xgb__colsample_bytree': 0.8,
'xgb__gamma': 1,
'xgb__max_depth': 6,
'xgb__min_child_weight': 3,
'xgb__n_estimators': 50,
}
```

XGBoostClassifier with these hyperparameters showed best F1 score 0.8977924674474604. To explain this:

- colsample_bylevel: Denotes the subsample ratio of columns for each split, in each level.
- colsample_bytree: Denotes the fraction of columns to be randomly samples for each tree.

- gamma: A node is split only when the resulting split gives a positive reduction in the loss function. Gamma specifies the minimum loss reduction required to make a split.
- max_depth: maximum depth of decision tree
- min_child_weight: minimum sum of weights of all observations required in a child
- n_estimators: number of decision tree

Conclusion

Reflection

- By combining the user's information and the user's information on the offers received, a model was created to predict whether or not the received offer was viewed.
- It was difficult to improve accuracy due to unbalanced data. However, when the F1 score was used as an index, an 0.89 was achieved.
- While preprocess the data, I handled missing values, converted categorical column to numeric type, normalize the columns, split the data into train set and test set by time.
- XGBoost classifier is used for model. It is a state-of-the-art model that is strong in overfit, fast in learning, and accurate.
- It showed a little better than the benchmark. There was little overfit. Learning accuracy and F1 scores were high. But the accuracy of predicting not to see the offer was low. It is lower if there is no user information
- I improved the model performance by hyperparmeter optimization using GridSearchCV, Cross Validation, removing unnecessary columns, inserting total transaction amount of users.
- Even without knowing the user information other than the total transaction amount, it was able to predict quite accurately than I expected. That's interesting.
- By error analysis, I didn't find good ideas to improve model performance. Instead, I'm gonna need other approaches or create special model for NA handling.

Another approaches for improvement?

- If there are more features and data about the user, the learning accuracy will be higher.
- If I have more data, I can apply the YouTube recommendation system algorithm. When this method is applied to the current dataset, the size of the dataset is greatly reduced, so the test accuracy is only 85%.

References

- XGBoost Documentation, https://xgboost.readthedocs.io/

- Scikit Learn, https://scikit-learn.org/
- Plot by Seaborn, https://seaborn.pydata.org/