Project Definition

Project Overview

It was given a mock transcript of the costumer behavior on the Starbucks mobile app, profile demographics and data on the offers they made available. With that information we will know the outcome of each interaction (offer received, offer viewed and finally offer completed) and all other purchases that did not involve an offer.

This project consists in a machine learning model as well as a web app to decide which offer is most suitable to each costumer or if any offer is suitable at all. This decision will be based on consumers purchase history and offers success history as well.

The web app will inform for the last day that we have costumer data what offer should be presented to him/her in a form of a list containing the customer id and the id of the offer to be served.

Problem Statement

Given the history of each costumer's actions on the app and their demographics we will show that is possible to offer much more assertive offers that will have a higher rate of success than selecting them at random.

Metrics

To access the quality of our prediction we will focus on the accuracy of the classification. The reason to use this metric is that is not a problem where is critical to have the lowest false positives or false negatives. We want simply to outperform randomness in selecting offers to show customers.

Analysis

Data Exploration

1) Portfolio

The portfolio data set shows the offers that are considered in this project along with information on channels of distribution, duration, type, etc.:

	reward	channels	difficulty	duration	offer_type	id	id_short
0	10	[email, mobile, social]	10	7	bogo	ae264e3637204a6fb9bb56bc8210ddfd	0
1	10	[web, email, mobile, social]	10	5	bogo	4d5c57ea9a6940dd891ad53e9dbe8da0	1
2	0	[web, email, mobile]	0	4	informational	3f207df678b143eea3cee63160fa8bed	2
3	5	[web, email, mobile]	5	7	bogo	9b98b8c7a33c4b65b9aebfe6a799e6d9	3
4	5	[web, email]	20	10	discount	0b1e1539f2cc45b7b9fa7c272da2e1d7	4

The portfolio data set is composed of:

- id (string) - offer id
- offer_type (string) - type of offer ie BOGO, discount, informational
- difficulty (int) - minimum required spend to complete an offer
- reward (int) - reward given for completing an offer
- duration (int) - time for offer to be open, in days
- channels (list of strings)
- id_short – we've atrributed a shorted id for better visualization.

2) Profile

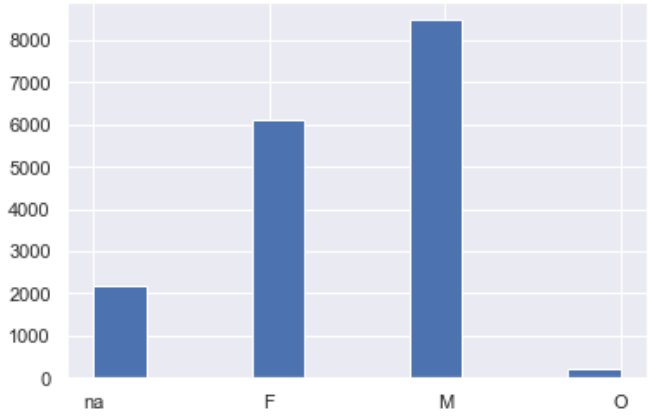
The profile data set shows information about the customer:

	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

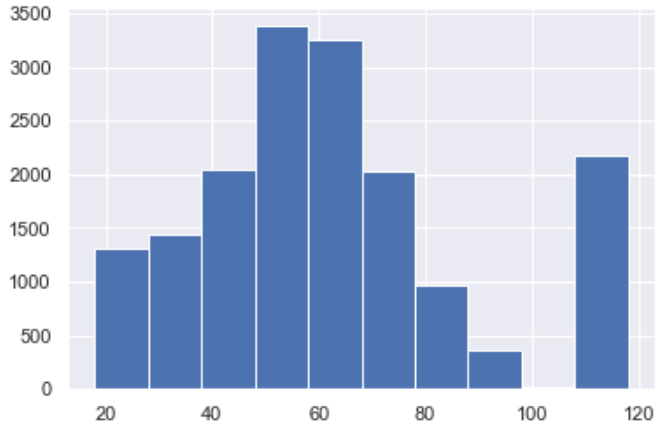
This data set is composed of the following columns:

- age (int) - age of the customer
- became_member_on (int) - date when customer created an app account
- gender (str) - gender of the customer (note some entries contain 'O' for other rather than M or F)
- id (str) - customer id
- income (float) - customer's income

We can observe that we do not have all information for every customer and we'll deal with this latter on the writeup. For now we can state the Gender is either M, F or O but it can also be a null value:



As for the age we will consider that 118 years old is a generic age probably due to considering a null value equals to 1900 for year of birth:

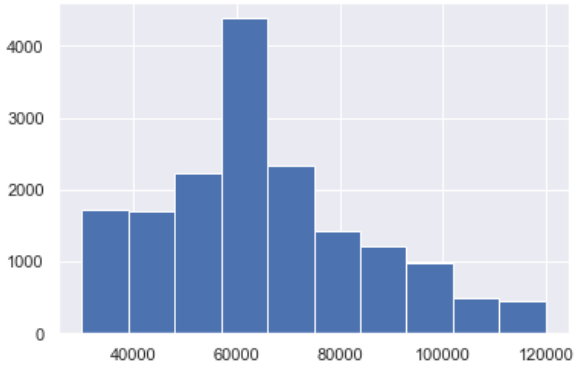


Here is the quantity of those null values on this dataframe:

```
profile.isna().sum()
```

```
gender              2175
age                    0
id                     0
became_member_on       0
income              2175
```

The income is distributed as:



3) Transcript

The transcript data set records all the transactions, offers received, viewed and completed:

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

The data set is composed of the following columns:

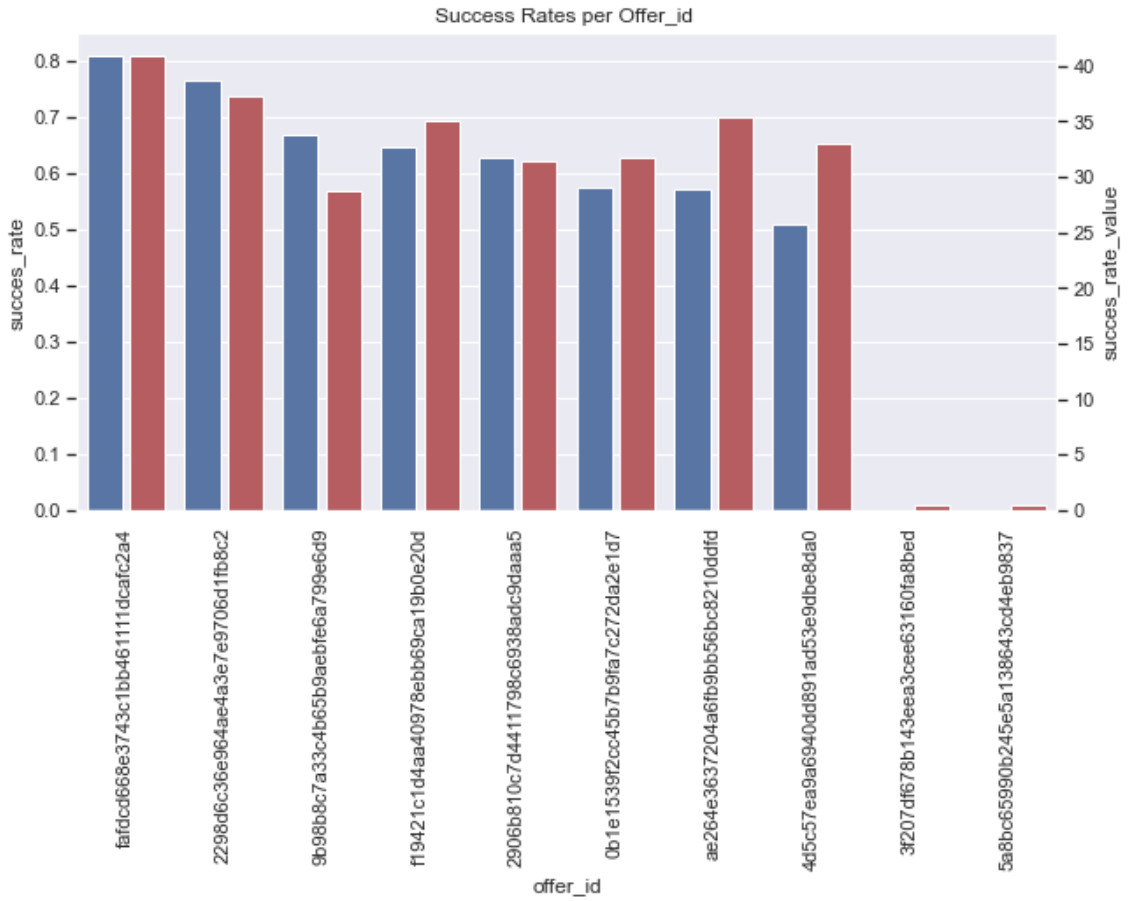
- event (str) - record description (ie transaction, offer received, offer viewed, etc.)
- person (str) - customer id
- time (int) - time in hours since start of test. The data begins at time t=0
- value - (dict of strings) - either an offer id or transaction amount depending on the record

For each time of the event is given on the value columns a dictionary of the characteristics of the event. We will later explode this dictionary into columns to be more easily treated on pandas.

Data Visualization

The first thing we'll analyze is the offer performance.

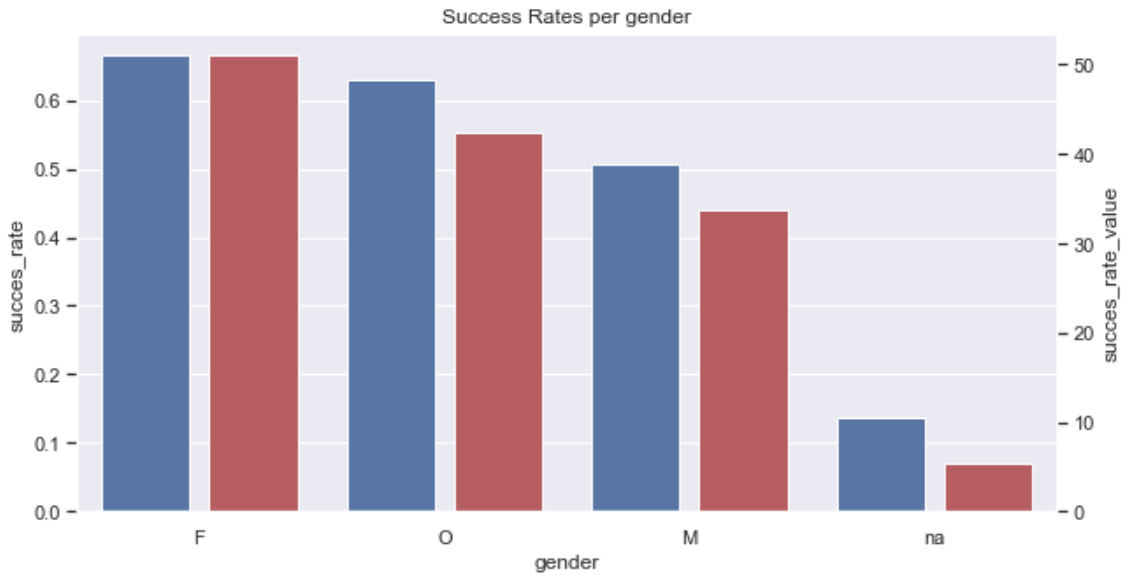
Here we've plotted in blue the success rate defined by the count of offer_success in relation to offer_viewed instances. In red we'll find the mean amount of transactions for each offer:



We can see that the first offer has an 80% success rate and algo bring more transaction value per offer. The last two offers are only informational, so they have only residual to null values attributed to them.

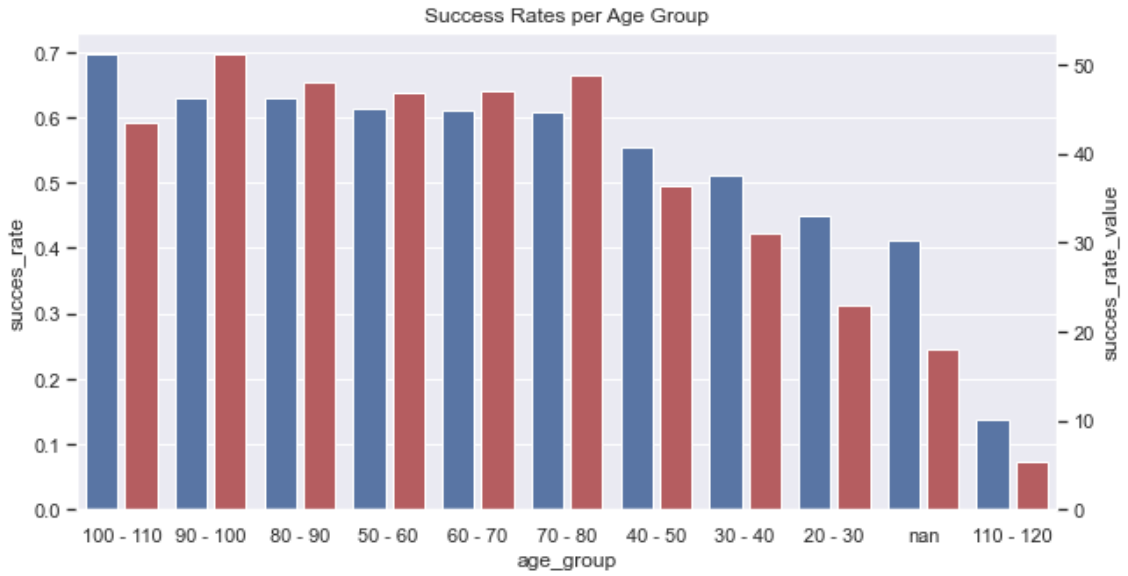
Next we'll look at success rate per demographics.

Gender:



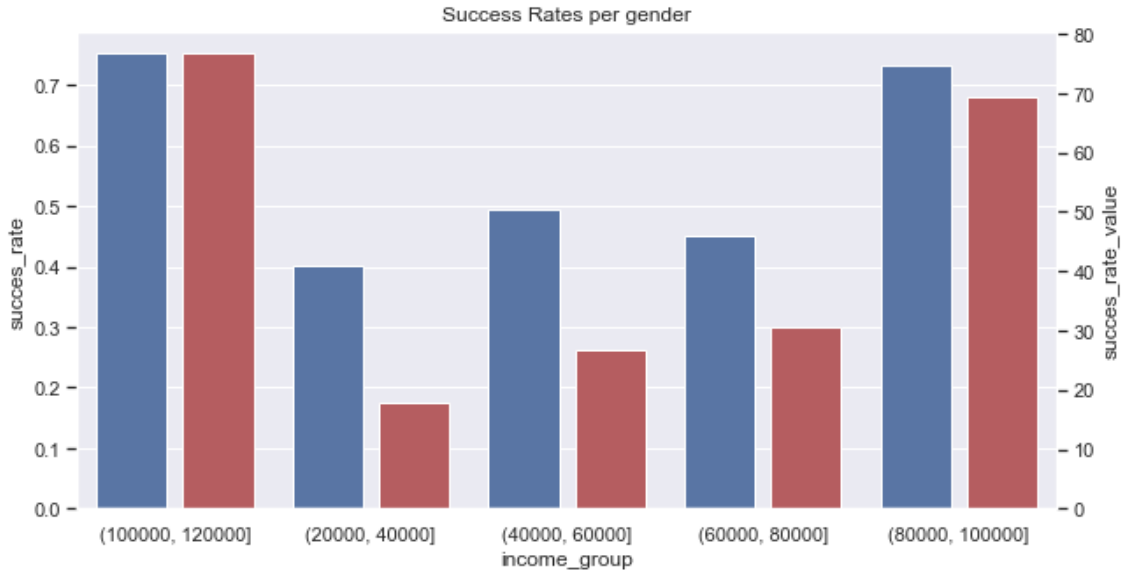
Female users responds better than the other groups in rates and values.

Age:



We can see that the younger age group (20 to 40) are the least responsive to offers.

Income:



Here we notice that the higher the income, more likely the customer is to respond to an offer.

Methodology

Data Preprocessing

The Data Frame we'll be using in this project will be formed by merging the transcript, portfolio and profile data set.

Next we'll explain the features generated within this dataframe.

- offer_amount:

To train the model we are going to need to know at the time the offer was sent if it turn out to be successful or not. To do so we'll group by person, offer_id and event and rank it by the time column.

Having observed that the offer completed has the same timestamp as the transaction that occurred we'll also merge the filtered dataframes so we can assign the offer completed with the transaction generated.

Next we'll merge this grouped dataframe to the original and fill the values corresponding to the amount that that offer generated with the mean. This will guarantee that the offer received and viewed has the same value as the offer completed.

- days_since_offer:

Using a lambda function to determine the time of the last offer received and then making if the value of the actual time minus that last offer received time.

- Dummy Variables:

Making dummies for the columns ['gender','offer_id']

Instead of 0/1 to the offer dummy variable we are going to assign the offer_amount value to them.

- 'amount_cum','offer_success_cum'

Cumulative sum of the total amount of offers values with success and cumulative sum of a flag of successful offers respectively.

- selected_offer:

This will be the columns we are going to predict in the algorithm. It states the offer_id if the offer was successful or 'offer_fail' if that offer had not generated any revenue.

Implementation

We used a XGBoost classifier algorithm to predict the offer most likely to be a success or 'offer_fail' if no offer would be selected.

We've trained the classifier with a GridSearch technique with the following parameters:

'classifier__max_depth':[2,4,6,8]

'classifier__n_estimators' : [100 , 200 ,300]

The project is run in two steps:

1. process_data.py

Here the dataframe from the project is created and stored in a data base. We will predict the offer we should offer for the last day available for each customer.

2. train_classifier.py

Here we will train and save the model to be used in the web app.

Refinement

The challenge in this project was to find features that indicate what offer should we offer. Initially we were using only dummy variables but we've found that we had a better result considering the cumulative effect of each offer success and the total offer success value.

Results

Model Evaluation and Validation

Best parameter from Grid Search:

max_depth=4, n_estimators=200

Classification report:

	precision	recall	f1-score	support
0b1e1539f2cc45b7b9fa7c272da2e1d7	0,11	0,05	0,07	680,00
2298d6c36e964ae4a3e7e9706d1fb8c2	0,14	0,15	0,14	989,00
2906b810c7d4411798c6938adc9daaa5	0,11	0,05	0,07	786,00
3f207df678b143eea3cee63160fa8bed	0,14	0,05	0,07	21,00
4d5c57ea9a6940dd891ad53e9dbe8da0	0,13	0,06	0,08	691,00
5a8bc65990b245e5a138643cd4eb9837	0,20	0,08	0,12	24,00
9b98b8c7a33c4b65b9aebfe6a799e6d9	0,17	0,10	0,12	868,00
ae264e3637204a6fb9bb56bc8210ddfd	0,14	0,09	0,11	743,00
f19421c1d4aa40978ebb69ca19b0e20d	0,14	0,07	0,09	867,00
fafdcd668e3743c1bb461111dcafc2a4	0,16	0,24	0,19	1039,00

offer_fail	0,81	0,96	0,88	8626,00
accuracy	0,59	0,59	0,59	0,59
macro avg	0,20	0,17	0,18	15334,00
weighted avg	0,52	0,59	0,54	15334,00

Here we notice that although the model has a small precision for selecting the offer it has a very good accuracy for predicting when the offer will fail.

Justification

The final results are discussed in detail. Explain the exploration as to why some techniques worked better than others, or how improvements were made are documented.

Conclusion

Reflection

We have successfully built an application designed to automate offer suggestions to customers in the platform.

We have loaded, analyzed the data, trained a model and built a web app for its usage.

Even though the accuracy for each offer is not the best it is better the random (10%) and most importantly we know when the customer is not susceptible to an offer.

Improvement

The implementation could be improved with feature engineering combining more columns, relationship between them, cumulative effects and so on. This could improve offer selection.