Udacity
Machine Learning Engineer Nanodegree
Starbuck's– Capstone Project Report

Mustafa Ahmad Aboelhamd

July 23, 2021

Definition

Project Overview

Offers and discounts have always been known as effective tools at increasing sales and attracting customers. But not all offers are created equal and different customers have different tastes. In this project we will be using machine learning to make better choices about which offers to send.

Problem statement

Given a dataset of customer info, offers info and a log of transactions and interactions with offers of the past month, we will train a model that – given a customer and an offer- tries to predict whether that customer will respond to that offer. Responding to an offer means viewing the offer and then making the required transactions to complete it while it is active.

We will first preprocess, clean and encode the data to the required form for the model. We will then train the model with said data and evaluate our predictions on test data.

Metrics

The metric used in this project will be accuracy. Accuracy is defined as the total of correctly predicted labels divided by the total of labels in the test dataset. The reason we are not using more advanced metrics such as precision, recall or AOC is because the amount of positive and negative labels is relatively evenly divided, therefore a useless model like one that predicts the same thing every time won't be able to perform well with this metric.

Analysis

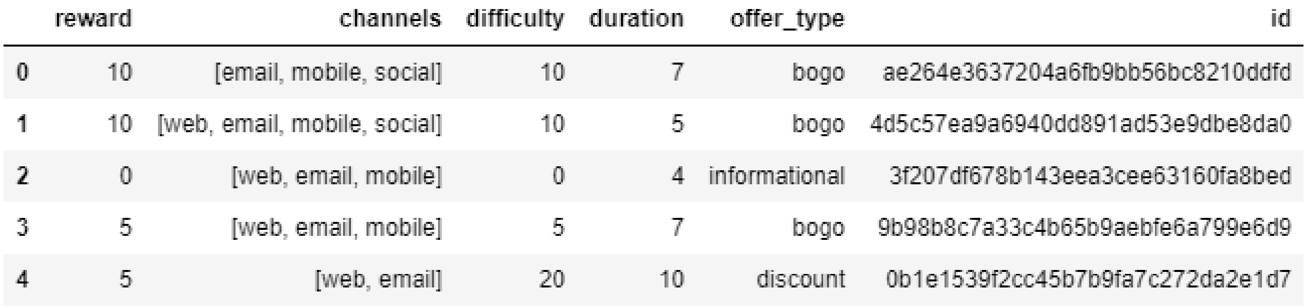
Data exploration

The dataset was obtained from Udacity as part of the Machine Learning Engineer Nanodegree and the explanations given for each feature are below. I also included a sample of the 5 top rows from each dataframe to help you visualize its structure.

The data is contained in three files:

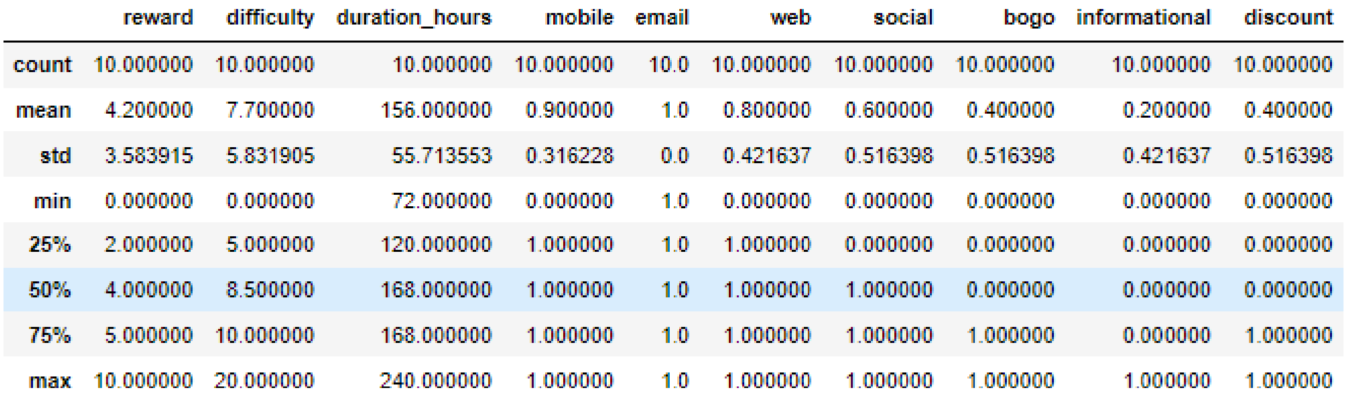
- portfolio.json - containing offer ids and meta data about each offer (duration, type, etc.)
- profile.json - demographic data for each customer
- transcript.json - records for transactions, offers received, offers viewed, and offers completed

portfolio.json (10 x 6)



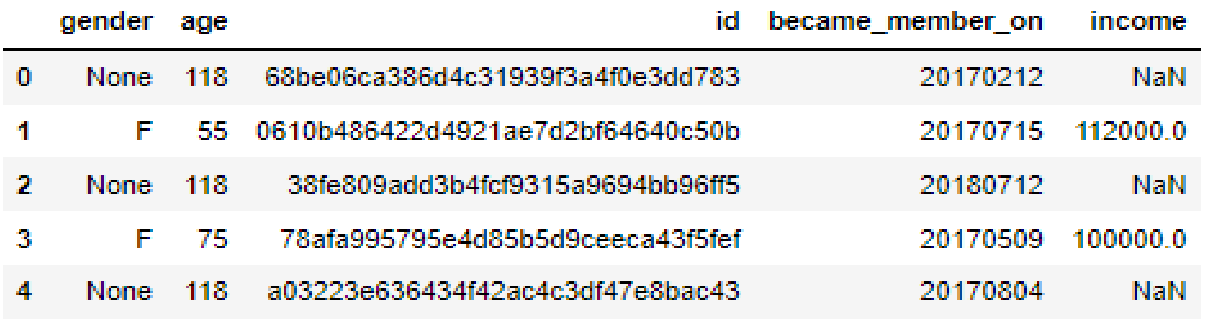
	reward	channels	difficulty	duration	offer_type	id
0	10	[email, mobile, social]	10	7	bogo	ae264e3637204a6fb9bb56bc8210ddfd
1	10	[web, email, mobile, social]	10	5	bogo	4d5c57ea9a6940dd891ad53e9dbe8da0
2	0	[web, email, mobile]	0	4	informational	3f207df678b143eea3cee63160fa8bed
3	5	[web, email, mobile]	5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

portfolio statistical description :



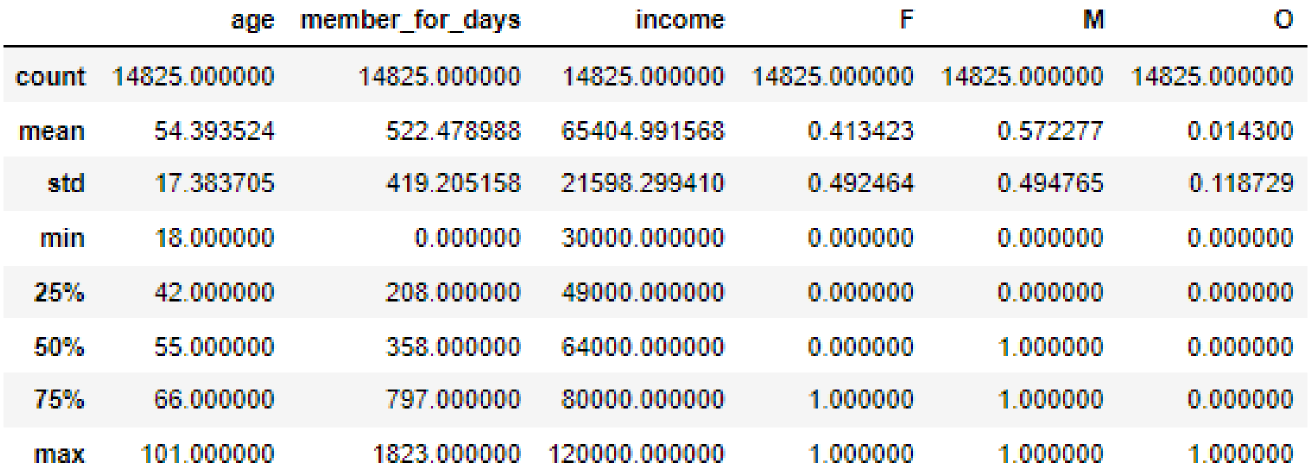
	reward	difficulty	duration_hours	mobile	email	web	social	bogo	informational	discount
count	10.000000	10.000000	10.000000	10.000000	10.0	10.000000	10.000000	10.000000	10.000000	10.000000
mean	4.200000	7.700000	156.000000	0.900000	1.0	0.800000	0.600000	0.400000	0.200000	0.400000
std	3.583915	5.831905	55.713553	0.316228	0.0	0.421637	0.516398	0.516398	0.421637	0.516398
min	0.000000	0.000000	72.000000	0.000000	1.0	0.000000	0.000000	0.000000	0.000000	0.000000
25%	2.000000	5.000000	120.000000	1.000000	1.0	1.000000	0.000000	0.000000	0.000000	0.000000
50%	4.000000	8.500000	168.000000	1.000000	1.0	1.000000	1.000000	0.000000	0.000000	0.000000
75%	5.000000	10.000000	168.000000	1.000000	1.0	1.000000	1.000000	1.000000	0.000000	1.000000
max	10.000000	20.000000	240.000000	1.000000	1.0	1.000000	1.000000	1.000000	1.000000	1.000000

profile.json (17000 x 5)



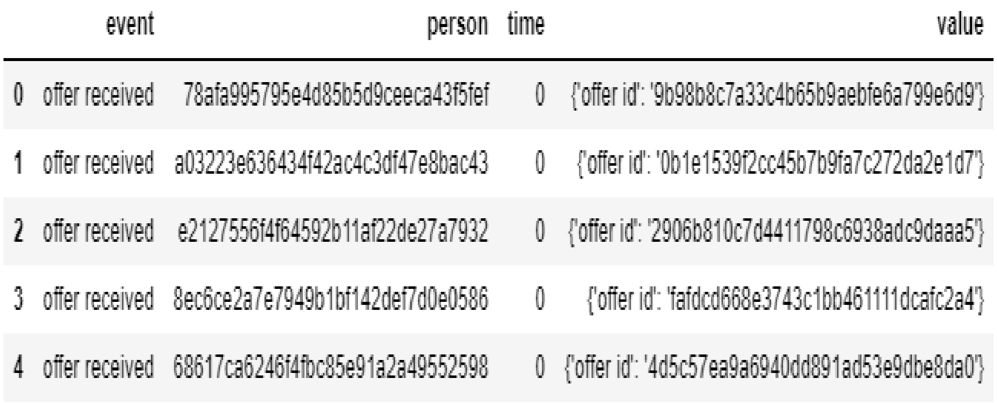
	gender	age	id	became_member_on	income
0	None	118	68be06ca386d4c31939f3a4f0e3dd783	20170212	NaN
1	F	55	0610b486422d4921ae7d2bf64640c50b	20170715	112000.0
2	None	118	38fe809add3b4fcf9315a9694bb96ff5	20180712	NaN
3	F	75	78afa995795e4d85b5d9ceeca43f5fef	20170509	100000.0
4	None	118	a03223e636434f42ac4c3df47e8bac43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

profile statistical description:

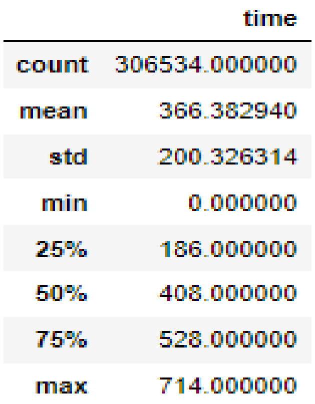


	age	member_for_days	income	F	M	O
count	14825.000000	14825.000000	14825.000000	14825.000000	14825.000000	14825.000000
mean	54.393524	522.478988	65404.991568	0.413423	0.572277	0.014300
std	17.383705	419.205158	21598.299410	0.492464	0.494765	0.118729
min	18.000000	0.000000	30000.000000	0.000000	0.000000	0.000000
25%	42.000000	208.000000	49000.000000	0.000000	0.000000	0.000000
50%	55.000000	358.000000	64000.000000	0.000000	1.000000	0.000000
75%	66.000000	797.000000	80000.000000	1.000000	1.000000	0.000000
max	101.000000	1823.000000	120000.000000	1.000000	1.000000	1.000000

transcript.json (306534 x 4) and its statistical description:



	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'}

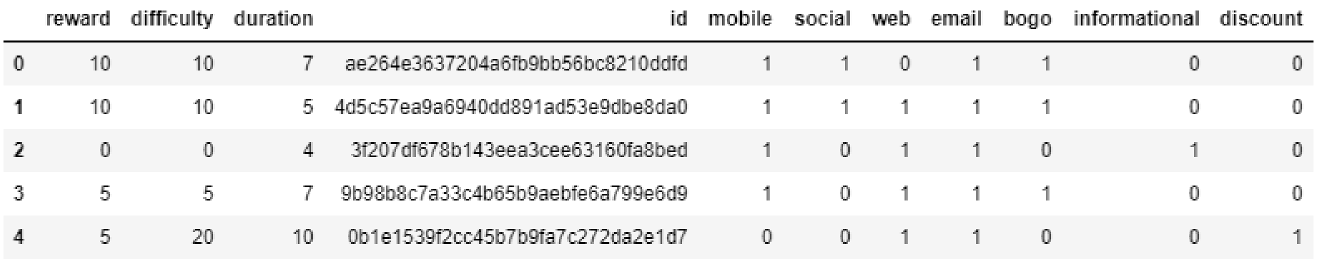


	time
count	306534.000000
mean	366.382940
std	200.326314
min	0.000000
25%	186.000000
50%	408.000000
75%	528.000000
max	714.000000

First thing that sticks out about the data is that in the profile dataframe, there are some NaN values in the gender and income columns. With the df.isna.sum() command we check all the dataframes and we see that profile is the only dataframe with NaN values.

Something else that is weird is how all of the rows in the above picture with NaN values have the same very unusual age, 118. Upon checking the count of rows with that age we see that it is 2175, the same number of NaN values in the dataframe. Therefore we assume that some customers didn’t want to give their information when registering as members, so the income and gender got NaN values, but the age defaulted to 118.

Another problem with the current data is that there are categorical values like gender, channels and offer type. We can’t feed our model text data, so we will have to convert it into numbers. Also when there are more that two categories we will convert that to one-hot-encoding. Below we see an example after one-hot-encoding was applied to channels and offer type.

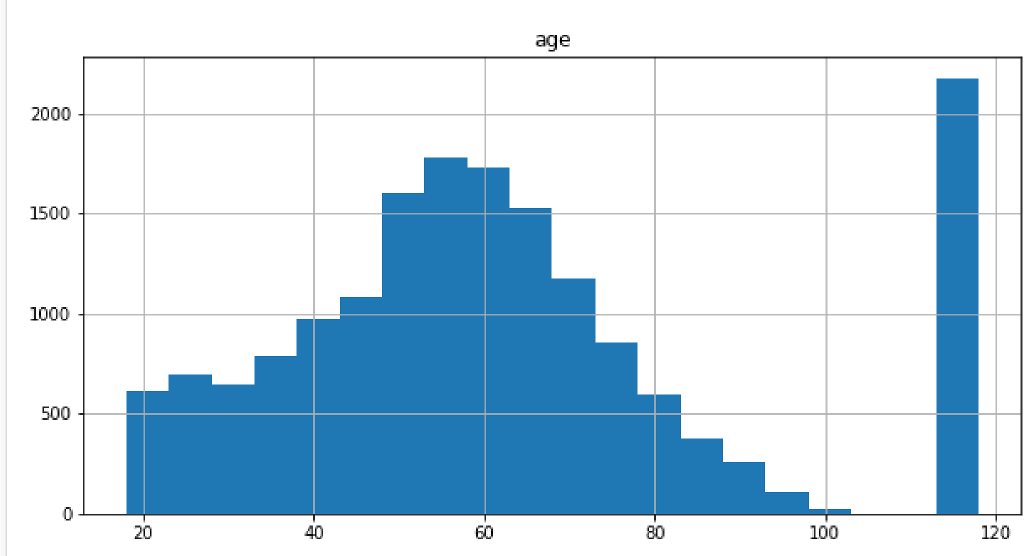


	reward	difficulty	duration	id	mobile	social	web	email	bogo	informational	discount
0	10	10	7	ae264e3637204a6fb9bb56bc8210ddfd	1	1	0	1	1	0	0
1	10	10	5	4d5c57ea9a6940dd891ad53e9dbe8da0	1	1	1	1	1	0	0
2	0	0	4	3f207df678b143eea3cee63160fa8bed	1	0	1	1	0	1	0
3	5	5	7	9b98b8c7a33c4b65b9aebfe6a799e6d9	1	0	1	1	1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

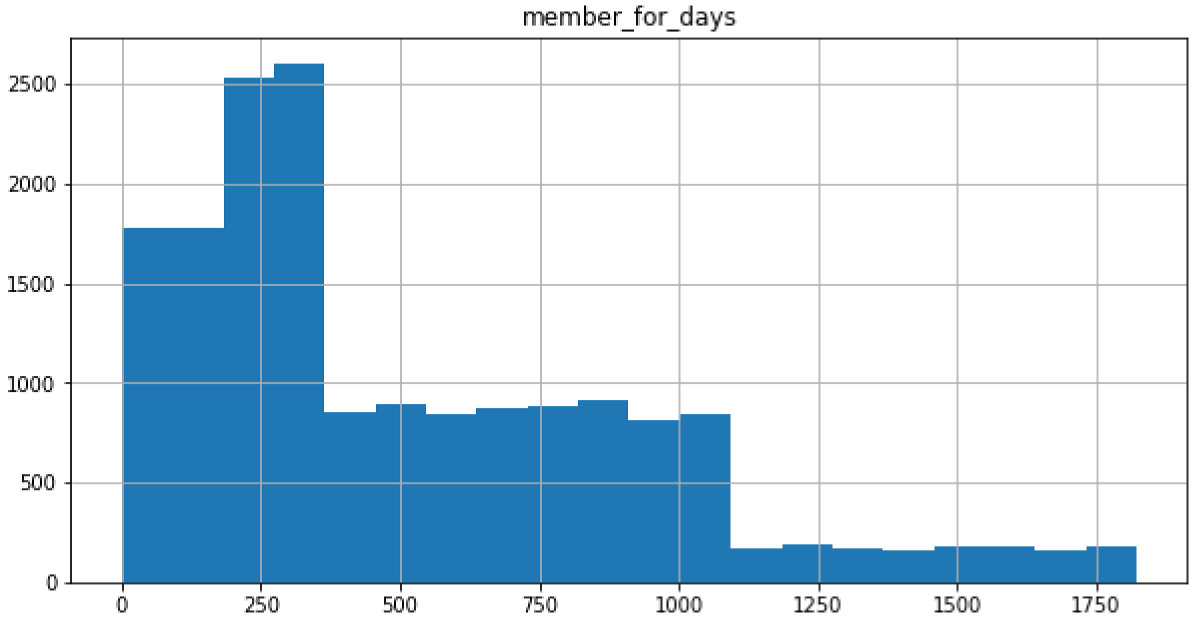
Lastly, the date the members became member on is an integer instead of a date.

Exploratory Visualization

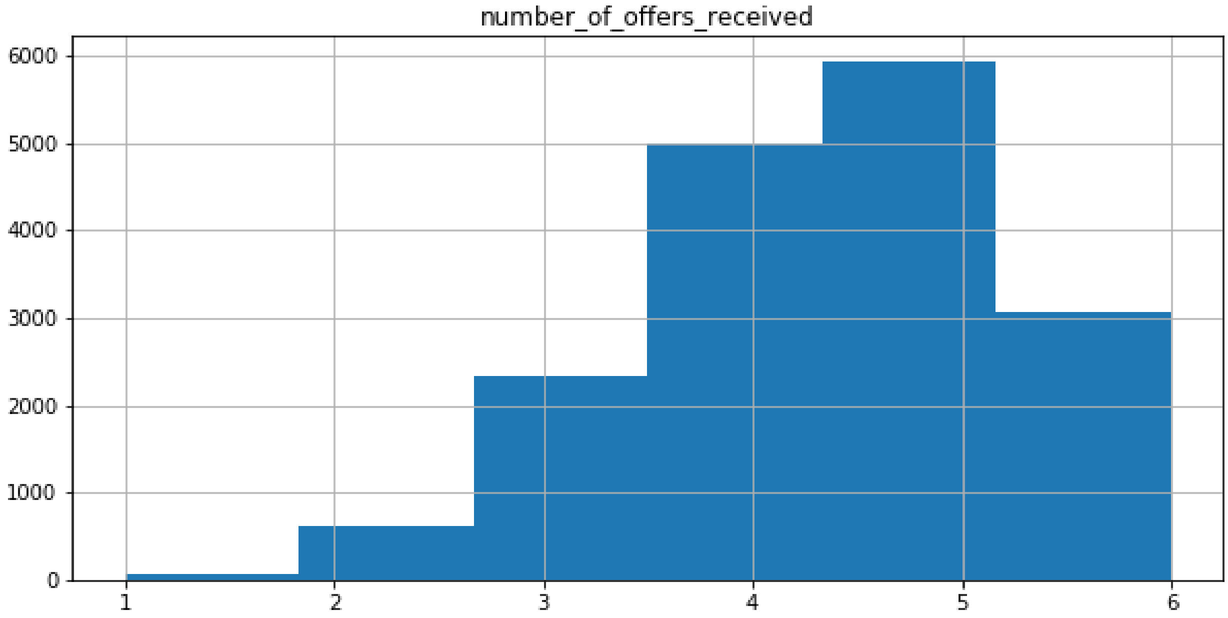
The distribution of age before dropping the NaN rows, the outliers are obvious



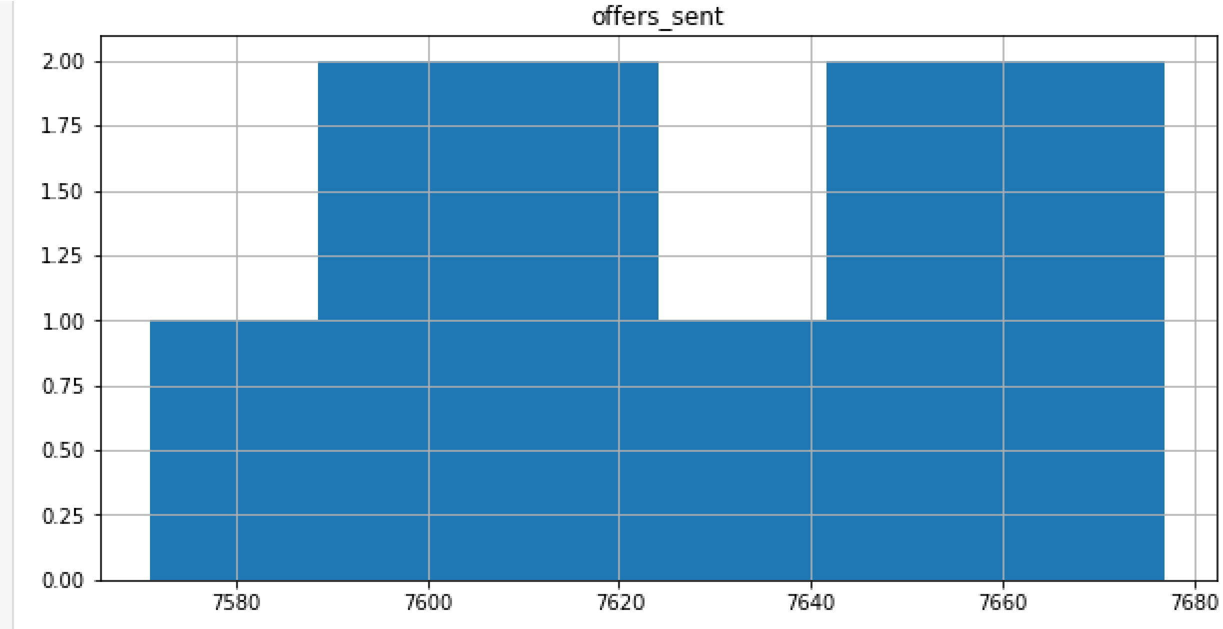
Distribution of the number of days that customers have been members. We can see that most customers are relatively new. This might mean that they might not be as aware of offers as older users.



Distribution of number of offers sent to each customer. It seems like there are not many outliers that will dominate.



Distribution of offers sent by offer_id. The axes are reversed but if you look at the x values you see that all of them have been sent a similar amount of times.



Benchmark:

A logistic regression algorithm will be used in this project as a benchmark model, because it is a simple model and I expect CatBoost to be able to beat it. After applying preprocessing (discussed below) and training the model I achieved an accuracy score of about 0.825, which will be our target to beat with the CatBoost technique.

Algorithms and Techniques:

This is a binary classification problems since there are only two results we can predict, either the customer will respond to the offer (1) , or he will not (0).

The main model we will be using for this problem is the **CatBoost** Classifier class.

CatBoost is a boosting algorithm that can handle categorical variables in the data. Most machine learning algorithms cannot work with strings or categories in the data. Thus, converting categorical variables into numerical values is an essential preprocessing step.

You can learn more about the CatBoost here.

CatBoost can internally handle categorical variables in the data. These variables are transformed to numerical ones using various statistics on combinations of features.

CatBoost often does better than Logistic Regression. I will use CatBoost because I have a categorical features. Another reason why CatBoost is being widely used is that it works well with the default set of hyperparameters. Hence, as a user, we do not have to spend a lot of time tuning the hyperparameters.

Methodology

Data Preprocessing

We identified some problems with the data in the data exploration section and here we will see which steps I took to solve them and other issues before feeding the data to our model.

The rows with NaN values and age 118 were dropped because almost all the relevant info about the consumer (everything besides the date of membership) was missing, so giving that to the model would just confuse it more than help it.

The categorical features ‘gender’, ‘channels’ and ‘offer type’ were converted into numerical values and then into one-hot-encoded ones.

‘Became member on’ feature was converted from an integer into date and then converted into a feature ‘member for days’ so it is numerical and able to be fed to the model.

Scaling was applied to all features that needed it.

Then the transcript dataframe was looped through and processed to create a new dataframe with the following features for every offer-person combination present in the transcript : viewed offer, completed offer, responded to offer(if someone did both), total money spent (per person) and viewing, completion and response rate (per person again). That resulting dataframe was merged with portfolio and profile, based on offer and customer id respectively.

The features that were not needed for the model (ids and labels) were dropped and the result was ready for the model.

Implementation

The CatBoostClassifier class model is pretty straightforward and has a default values for all parameters and the project is simple and standard enough that the default values take care of the problem. That being said there is an important parameter, which is iterations. The default value of it is 1000, but I decided to change it and see how the accuracy would react. In the end I decided on values below (explanation in the next section, refinement):

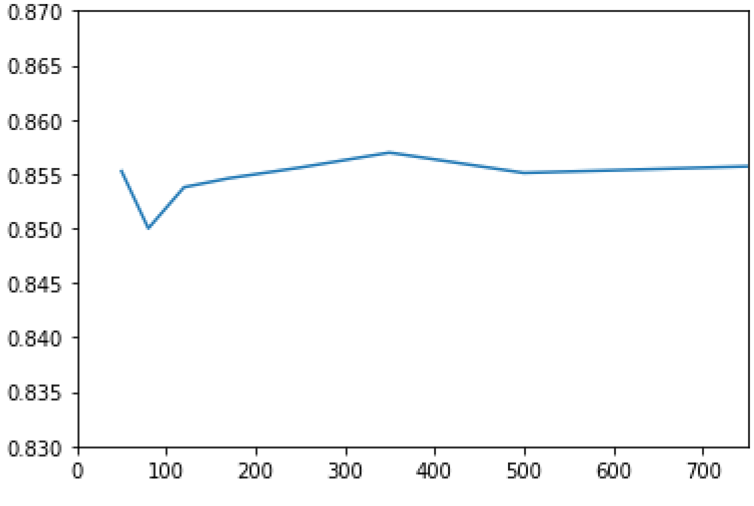
Iteration = 350

Learning rate = 0.126 (decided by the model based on iteration value)

Refinement

As I said above, I decided to test if the default value of 1000 for iterations was optimal for the problem. At first I decided with bigger values to see if maybe the model was not converging yet. The accuracy did not seem to improve though, so I decided to lower the value to see maybe there was some overfitting going on. That did not seem to be the case either, in fact the accuracy stayed pretty stable with some fluctuations up and down that I assume are due to randomness. I decided to cut the by about to a third to reduce the training time.

The graph below shows some samples I took. X axis is the number of iterations and y axis is the accuracy score. Mind the values in the Y axis, the fluctuations are smaller than they appear when on 0-1 axis. I decided on 350 because it peaks here. (Note: I took sample on the following values: 50, 80, 120, 170, 250, 350, 500, 750)



Results

Model Evaluation and Validation

I tested the model with 33% of data that was not in the training set and got and accuracy of 0.857. Changing the data test size and content did not change this percentage significantly

Justification

The model got an accuracy of 0.857, which means it is more accurate than the benchmark. While there is still some improvement, given a customer and an offer, the model will in most cases predict correctly whether or not the customer will respond to that offer or not, which can be useful for a company.