CAPSTONE PROJECT STARBUCKS

Julia Gilly

Data Science Nanodegree

June 24th, 2022

CONTENT

DEFINITION

PROJECT OVERVIEW

This project contains three simulated data sets (profile.json, portfolio.json and transcript.json). This data mimics customer behavior. It shows us, what will happen, when Starbucks sends out an offer to the customers, how they react, if they open the offer, if they do a transaction afterwards and if they complete the offer. There are three kinds of offers here: BOGO (buy-on-get-one), discount and informational with the last one being offer (it cannot be completed as it is only informational and therefore not a "real" offer). Every offer has an expiry date.

For simplification there are no explicit products used here.

The task is to find out which people (regarding age, gender…) will be most responsive to the offer given (and therefore make a transaction).

THE DATA

As mentioned above, we have three data sets which were provided by Udacity.

1. profile.json
2. portfolio.json

3. transcript.json

Profile.json holds all information regarding the customer:

- gender (categorical): M=Male, F=Female, O=Other or null
- age (numeric), missing value encoded as 118
- id (customer id)
- became_member_on (date) format YYYYMMDD
- income (numeric)

Portfolio.json holds all the information regarding the offer:

- reward: (numeric) money awarded for the amount spent
- channels: (list) web, email, mobile, social
- difficulty: (numeric) money required to be spent to receive reward
- duration: (numeric) time for offer to be open, in days
- offer_type: (string) bogo, discount, informational
- id: (string/hash)

With Transcript.json we find out when the offer was sent, received, looked at and completed. It even gives us the transactions done from each customer (including the money spend):

- person: (string/hash)
- event: (string) offer received, offer viewed, transaction, offer completed
- value: (dictionary) different values depending on event type
    - offer id: (string/hash) not associated with any "transaction"
    - amount: (numeric) money spent in "transaction"
    - reward: (numeric) money gained from "offer completed"
- time: (numeric) hours after start of test

EVALUATION METRICS

For this project I used the Accuracy Score:

Accuracy = $\frac{(\mathbf{TP}+\mathbf{TN})}{(\mathbf{TP}+\mathbf{TN}+\mathbf{FP}+\mathbf{FN})}$

With TP=True Positive, TN=True Negative, FP=False Positive and FN=False Negative.

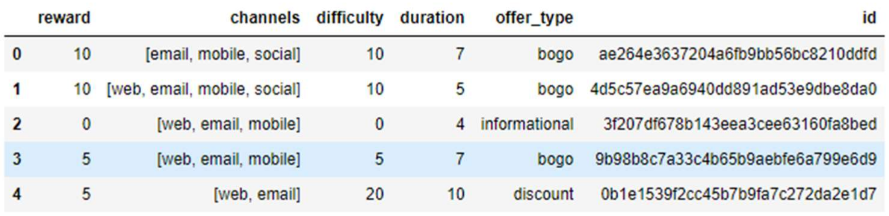
ANALYSIS

WRANGLE DATA (GATHER AND ASSESS)

For gathering the data I used the standard function from pandas (pd): pd.read_json. Then I had a closer look at each data set. To make it easier and not do so much of coding I created two functions that help me show the data in a pie chart (pie_chart) and in a histogram (hist_chart). For those two functions you only need the parameters of which table you want to analyze and which column in the table.

PORTFOLIO.JSON

With portfolio.head() we get the following overview:



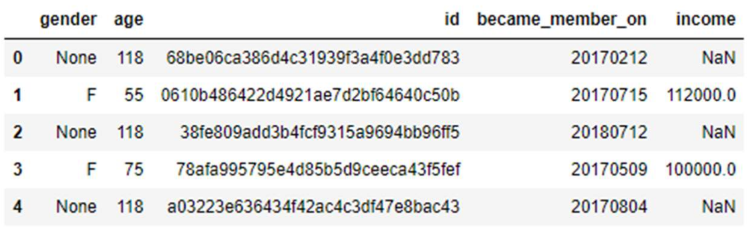
	reward	channels	difficulty	duration	offer_type	id
0	10	[email, mobile, social]	10	7	bogo	ae264e3637204a6fb9bb56bc8210ddfd
1	10	[web, email, mobile, social]	10	5	bogo	4d5c57ea9a6940dd891ad53e9dbe8da0
2	0	[web, email, mobile]	0	4	informational	3f207df678b143eea3cee63160fa8bed
3	5	[web, email, mobile]	5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

1 portfolio.json

You can see that the column "channels" holds more than one information. Each channel should be shown in a different column. It might even be beneficial to create a dummy variable out of the column offer_type for further investigation.

PROFILE.JSON

Profile.head() shows us the following overview:

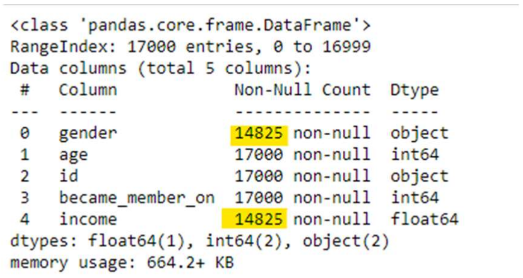


	gender	age	id	became_member_on	income
0	None	118	68be06ca386d4c31939f3a4f0e3dd783	20170212	NaN
1	F	55	0610b486422d4921ae7d2bf64640c50b	20170715	112000.0
2	None	118	38fe809add3b4fcf9315a9694bb96ff5	20180712	NaN
3	F	75	78afa995795e4d85b5d9ceeca43f5fef	20170509	100000.0
4	None	118	a03223e636434f42ac4c3df47e8bac43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

2 profile.json

The income and gender columns have a lot of NaN.

By looking at profile.info() we can detect the same amount of NaN for gender and income:



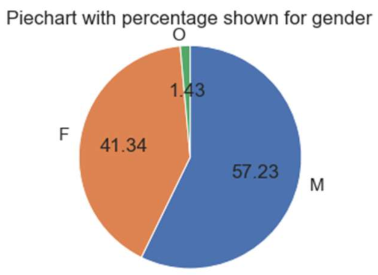
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 17000 entries, 0 to 16999
Data columns (total 5 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   gender            14825 non-null  object
 1   age               17000 non-null  int64
 2   id                17000 non-null  object
 3   became_member_on  17000 non-null  int64
 4   income            14825 non-null  float64
dtypes: float64(1), int64(2), object(2)
memory usage: 664.2+ KB
```

3 Info about profile.json

This does not seem coincidencial. It might be that those customers without an income did not specify their gender.

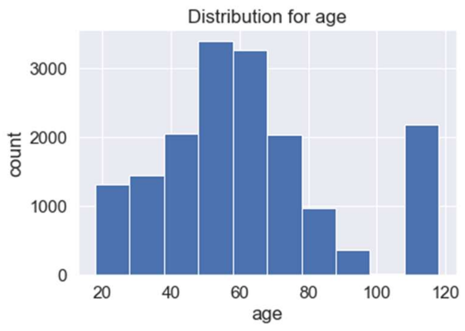
So, by looking a little closer at those with age 118 (which is a dummy) we find out that those with age 118 didn't give us an income or gender. It might be beneficial to delete those customers as they do not give us the required information for the algorithm used later.

By looking a little bit closer at gender, that we have slightly more male customers:



4 Gender distribution for all customers

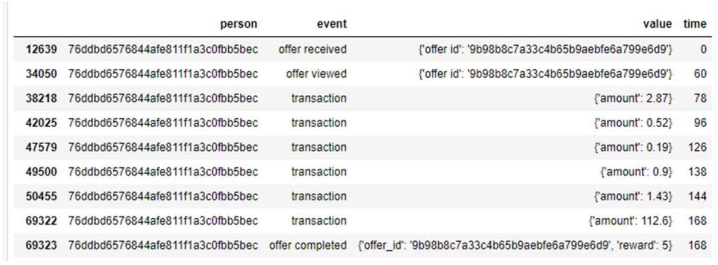
If we take a closer look at age, we see that the customers are slightly normally distributed (except for those with age 118 which is a false age):



5 Age Distribution for all customers

TRANSACTION.JSON

The last data set holds all the information about the transactions and offers (received, viewed, completed):



	person	event	value	time
12639	76ddbd6576844afe811f1a3c0fbb5bec	offer received	{'offer id': '9b98b8c7a33c4b65b9aebfe6a799e6d9'}	0
34050	76ddbd6576844afe811f1a3c0fbb5bec	offer viewed	{'offer id': '9b98b8c7a33c4b65b9aebfe6a799e6d9'}	60
38218	76ddbd6576844afe811f1a3c0fbb5bec	transaction	{'amount': 2.87}	78
42025	76ddbd6576844afe811f1a3c0fbb5bec	transaction	{'amount': 0.52}	96
47579	76ddbd6576844afe811f1a3c0fbb5bec	transaction	{'amount': 0.19}	126
49500	76ddbd6576844afe811f1a3c0fbb5bec	transaction	{'amount': 0.9}	138
50455	76ddbd6576844afe811f1a3c0fbb5bec	transaction	{'amount': 1.43}	144
69322	76ddbd6576844afe811f1a3c0fbb5bec	transaction	{'amount': 112.6}	168
69323	76ddbd6576844afe811f1a3c0fbb5bec	offer completed	{'offer_id': '9b98b8c7a33c4b65b9aebfe6a799e6d9', 'reward': 5}	168

6 transaction.json

The value column holds information about the offer, the amount spent and the reward. This should be in different columns. Also, we should create a dummy variable out of event.

CLEAN DATA

<u>What must be cleaned</u>:

1. portfolio.json:
    - channels hold more than one info
    - offer_type column holds different things
2. profile.json:

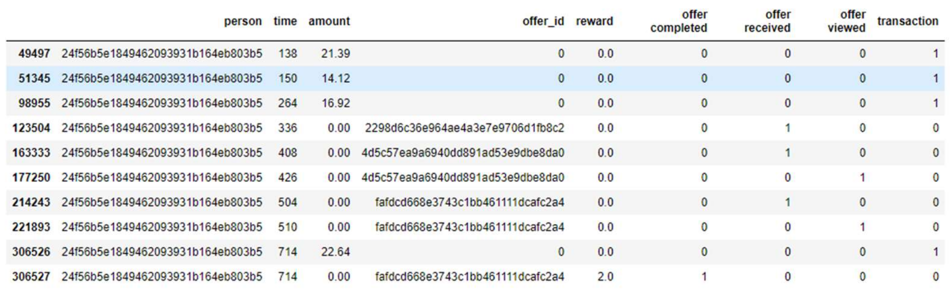
- missing data in gender and income
- age 118 is not the real age of the customer --> placeholder

3. transcript.json:
    - value holds more than one info
    - offer id is written differently in value
    - event column hold different things

So mostly I had to create the dummy variables for channels, offer_type, event. This was done by using the python function pd.get_dummies.

Because those with age 118 hold no more information about the customer (only became_member_on) I deleted those customers.

I used a lambda-function for unpacking the tuples in value to a dataframe. By doing this I got two offer ids: offer id and offer_id. So, I filled those in offer_id with the other column offer id. The last step of cleaning was filling the missing values in transcript with 0:



	person	time	amount	offer_id	reward	offer completed	offer received	offer viewed	transaction
49497	24f56b5e1849462093931b164eb803b5	138	21.39	0	0.0	0	0	0	1
51345	24f56b5e1849462093931b164eb803b5	150	14.12	0	0.0	0	0	0	1
98955	24f56b5e1849462093931b164eb803b5	264	16.92	0	0.0	0	0	0	1
123504	24f56b5e1849462093931b164eb803b5	336	0.00	2298d6c36e964ae4a3e7e9706d1fb8c2	0.0	0	1	0	0
163333	24f56b5e1849462093931b164eb803b5	408	0.00	4d5c57ea9a6940dd891ad53e9dbe8da0	0.0	0	1	0	0
177250	24f56b5e1849462093931b164eb803b5	426	0.00	4d5c57ea9a6940dd891ad53e9dbe8da0	0.0	0	0	1	0
214243	24f56b5e1849462093931b164eb803b5	504	0.00	fafdcd668e3743c1bb461111dcafc2a4	0.0	0	1	0	0
221893	24f56b5e1849462093931b164eb803b5	510	0.00	fafdcd668e3743c1bb461111dcafc2a4	0.0	0	0	1	0
306526	24f56b5e1849462093931b164eb803b5	714	22.64	0	0.0	0	0	0	1
306527	24f56b5e1849462093931b164eb803b5	714	0.00	fafdcd668e3743c1bb461111dcafc2a4	2.0	1	0	0	0

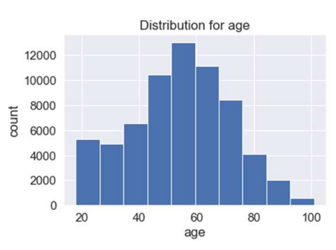
7 transaction.json after cleaning

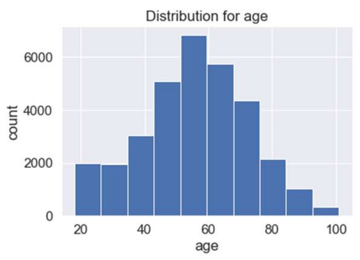
EXPLORE DATA

For exploring the data, I merged the three data sets together in one dataset called df. Afterwards I looked at different attributes:

- age
- gender
- amount of money spent
- duration (offer)
- income
- channels
- which kind of offer

Age:

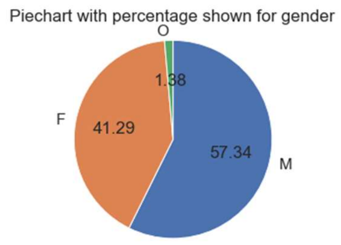


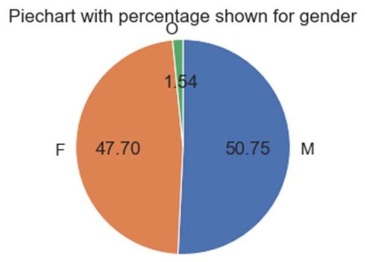


8 Distribution fo age by Offer received Distribution of age by offer completed

It seems a little bit more likely that customers around 50 are completing an offer

Gender:

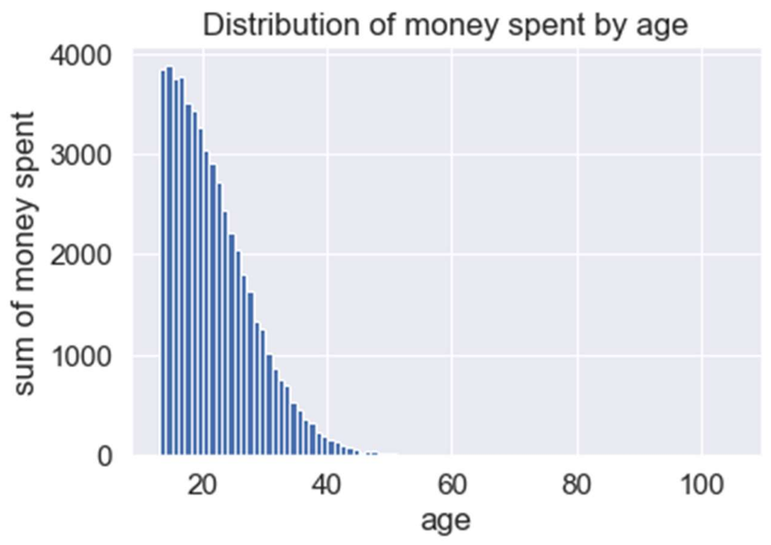




9 Distribution of gender by Offer received Distribution of gender by Offer completed

There are more women completing an offer than men.

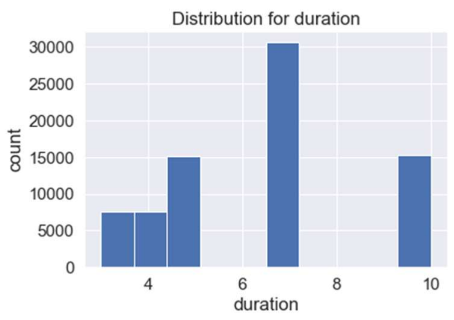
Amount of money spent:

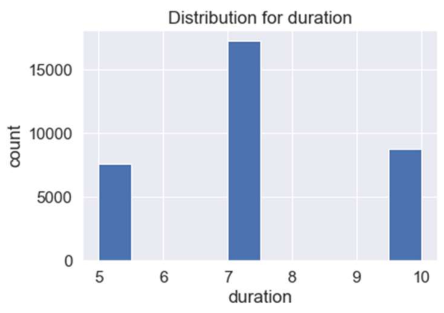


10 Distribution of money spent by age

Even so people around the age of 50 are more likely to complete an offer, we see that younger people spent more money.

Duration (offer):

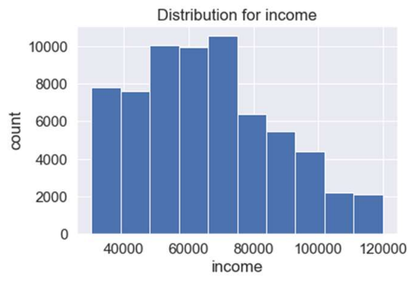


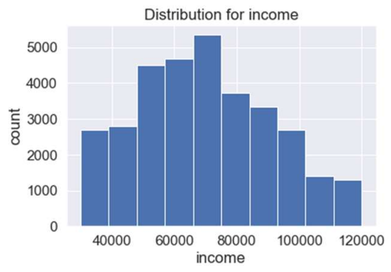


11 Distribution of duration for offer received **Distribution of duration for offer completed**

What becomes obvious by looking at the distribution for duration for offers received and completed is that the offers with 3/4 days are never completed.

Income:





12 Distribution of income for offer received **Distribution of income for offer completed**

This might not be as obvious as the other parameters, but it seems that customers with an income under 5.000 are not completing offers that often.

Channels (completion rate for each channel):

Email: 58,17%

Mobile: 54,77%

Social: 51,11%

Web: 67,44%

We see that web is the best channel to promote something, because it has the highest completion rate out of the four channels.

Which kind of offer (completion rate for each offer):

Bogo: 41,57%

Discount: 83,52%

Informational: 0%

Offering discount is the best way to promote our offers as it has the highest completion rate.

CONCLUSTION OF ANALYSIS

After cleaning the data and looking at it from different angles, it became apparent that the age and gender are import for completing an offer. On the other hand, younger people tend to spend much more money on Starbucks without actually completing the offers. So, it seems we should take the age and gender into consideration of sending out offers. For younger people we might have to think of different kind of offers (something that is more attractive to them).

Also interesting is the duration for an offer. It seems we should get rid of the 3/4 days offers, as they don't seem to work at all. Best is to use the 7/10 days offers.

Regarding the income it has a small influence on whether someone completes an offer or not, but it might not be that influential for our later model.

The Channels are influential on whether a person completes an offer or not. Web and email seem to have the best chances on completing an offer if it was viewed.

The kind of offer is important as well. It seems that a discount is working far better than bogo. Not really surprising was the fact that informational offers have a completion rate of 0% (which was expected because there is no completion here).

ALGORITHM

Before I can start with the algorithm, I had to prepare the data a little further. I created a new column called event, which has 0 for offer received, 1 for offer viewed, 2 for transaction and 3 for offer completed. For the gender I changed M to 0, F to 1 and O to 2. All missing values were filled with 0. For further investigation I splitted the became_member_on column into year and month.

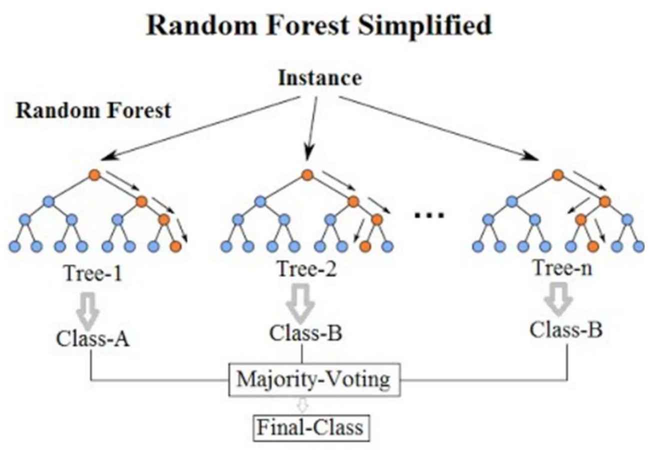
The parameters (X) I used for predicting if an offer was received, viewed, completed or a transaction was made where gender, time, age, income, year, month, duration, difficulty, bogo, discount and informational.

After splitting the data into training and test data I tried different classifiers:

- RandomForestClassifier
- KNeighborsClassifier
- AdaBoostClassifier

RandomForest:

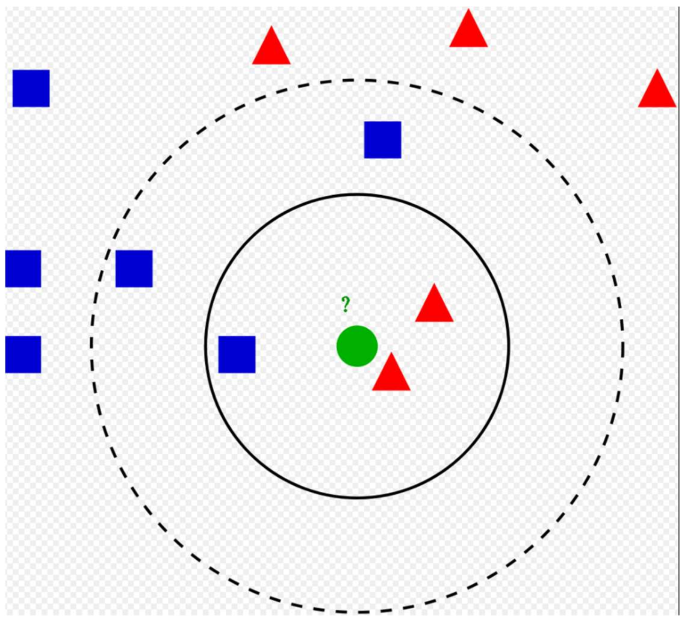
The random forest is a supervised learning algorithm. It creates and merges multiple decision trees into one forest. The output is the class selected by most trees.



11 Random Forest (Source Wikipedia: https://en.wikipedia.org/wiki/Random_forest)

KNeighbors:

KNeighbors is also a supervised learning algorithm. The output (in our case the event (offer received…)) is decided by the k-nearest neighbors. If k=1 than the output is simply identified by the nearest neighbor.



12 K-Nearest Neighbor: solid line for k=3 and dotted line for k=5 (Source : https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm)

In the picture above we would decide for k=3 that the green dot is red, for k=5 the green dot would be blue.

AdaBoost:

AdaBoost is short for Adaptive Boosting. It trains and deploys tress in series. AdaBoost implements boosting where it puts more weight on difficult to classify outputs and not so much weight on those which are already classified correctly.

When I used the RandomForestClassifier with the initial setting I got an accuracy rate of 0.748.

I changed the following parameter in the RandomForestClassifier:

- max_depth
- min_samples_split

Max_depth is the maximum depth of the tree. Min_sample_split changes the minimum number of samples required to split an internal node.

After adjusting the model a bit (max_depth=18, min_samples_split=7) the accuracy rate was a little bit better: 0.807

The KNeighborsClassifer did not give a really good accuracy rate (0.76).

I got the best result with the AdaBoostClassifer.

I changed the following parameters for AdaBoost:

- n_estimators
- algorithm
- learning_rate

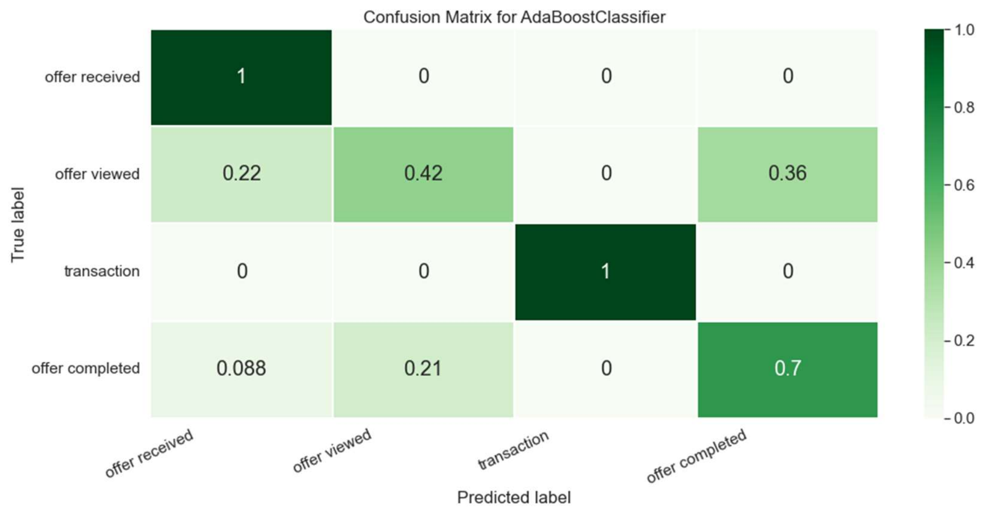
The algorithm I used was 'SAMME' which means I used the discrete boosting algorithm.

The learning_rate tells us how much we are going to contribute with the new model in comparison to the existing one.

N_estimators give us the maximum number of estimators at which boosting is terminated.

After adjusting the model with n_estimators=700, algorithm='SAMME' (discrete boosting algorithm), and learning_rate=0.449 I got an accuracy rate of 0.858.

But this is just the overall accuracy rate. I wanted to know if I would predict correctly if an offer would be completed, so I looked at the confusion matrix:



13 Confusion Matrix of AdaBoostClassifier after modification

The model will predict with a 70% accuracy if an offer will be completed or not.

CONCLUSION

With the model we will be able to predict (with 70%) if a customer will actually complete an offer and therefore spent more money.

For a better model we might need more information about the customers and the offers. But this could be a good starting point.

We might have to think about the offers itself. There are offers which were never completed (duration 3/4 days). In a next step I would try to catch more younger customers in the offers, because those tend to spend more money than older ones. But as shown in the analysis they do not complete the offers that often. Maybe we should try other offers for them? Find out what would make them complete the offers?