Starbucks Challenge - Capstone Proposal

Illán Lois Bermejo

Index

I. DEFINITION

1) Project Overview

The end goal of our project was to improve the accuracy of the current promotional campaigns, by improving the rate of conversion of the offers, measured as those converted divided by the total, as a means of increasing the revenue, both by not exhausting our client base, making our offers more relevant, and not wasting offers on subjects that were prone to buying anyway.

In order to do so, we have evaluated the information provided in three files, related to offer campaigns, the customers such offers were sent to, and the characteristics of each offer, as to determine whether the offers sent to their customer base are being used, and which characteristics both in the offer and the customer are relevant for the determination of the success of the offer.

We have decided to divide our problem in two parts, having each their corresponding model:

- The first part is to determine which customers are offer-sensitive. We have created a model that predicts the user *"type"* in relation with our offers with an accuracy of 80% (when considering all users) or 95% (when we only adjust to those clients we are sure enough and have enough data about them).
- The second model determines, starting from the characteristics of an offer (including the recipient's *"type"*), the success of that offer. In this regard, we have obtained accuracies of 96%, which is a very good result, as it almost allows us to pinpoint our offers.

We decided to embrace this approach, and do two models, because:

- As we will see, the customer *"type"* is one of the most relevant features when determining the success of an offer.
- This means that, for other new users on whom we don't have still any offer-related information, we can use our customer model to generate their *"type"* and feed it to the offers model.
- Having two, independent models gives us the freedom of choosing offering campaigns based on the offer itself or our customer base

As we will see, our results suppose an incredible improvement both to the current success ratio (measured as an improved accuracy rate), and the result a less complex models, used as benchmark.

2) Problem Statement

Sending offers and promotions to customers and potential customers of a business in order to increase revenue, get market share or publicize brands or products is a commercial strategy used for a long time.

However, sending these offers indiscriminately to the entire customer base of a business is not the most efficient way to tackle this task.

This is what has traditionally been done by businesses, due to the lack of the necessary information about their clients to be able to apply optimization strategies.

This implies loss of effectiveness of these campaigns, due to various reasons:

- sending offers to users who have purchased the products or services of these companies without the existence of the offer
- spamming: sending a large number of emails to a customer can cause them to not pay attention to these offers, or apply filters to eliminate them without being read
- generalized shipping prevents us from sending offers selectively to certain customers and at certain times, which would allow us the possibility of a possible purchase

For all these reasons, for years all businesses have been dedicated to gathering information about their clients, through, for example, loyalty programs, with which they can focus their commercial and marketing campaigns on the right people.

However, it has been with the introduction of machine learning techniques that these strategies based on user and customer data have been able to unleash the full power of this information.

Therefore, in our case, the problem that we intend to solve is the optimization of the sending of offers to Starbucks customers, to avoid the loss of effectiveness associated with the problems described above.

3) Evaluation metrics

Accuracy has been the metric of our choice, for both the customers and offers models developed.

It seemed the obvious choice, as our goal was to create a model that could determine whether a customer was susceptible to offers, and a model that could predict whether an offer would be successful.

Nevertheless, we have not only paid attention to our accuracies, but we have also studied our confussion matrices, in order to check for any anomalies our models could produce.

II. Analysis

1) Data Exploration (I)

The data we have gathered to solve our problem is presented beneath. It consists of three json files:

- portfolio.json

This is a small dataset, containing only 10 rows, describing the portfolio of offers that are sent to customers. The '*channels'* column contains strings and needs treatment.

No null nor duplicated values.

We see that portfolio only has ten rows of data, corresponding to each different offers clasified in three kind of offers:

- bogo --> buy one, get one. Four offers, divided into two subkinds with different rewards (10 and 5, pressumably corresponding to 1 + 1 for free and 2 + 1 offers
- discount --> Four different kind of offers, with rewards of 5, 3, 2 and 2 again (50, 30 and 20%?)
- informational --> two offers, 0 reward (information emails, we pressume)

Beyond that, we observe that we have also information on different difficulties and durations for each offer, and also information on which channels was this offer offered.

We need a little manipulation: we had to divide the *channel* column into four (['web', 'email', 'mobile', 'social']), with ones when the corresponding channel was used and zeros when it did not.

This was the end result:

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

- profile.json

A bigger file, containing demographic information about 17.000 customers. Some missing values for all users declared with age of 118 years, in regards of gender and income. No duplicated values.

After some studies, we decided to drop the nulls. We also parsed the dates correctly, and extracted day, month and year.

End result:

	age	gender	id	income	date_bmo	year_bmo	month_bmo	day_bmo	seniority
1	55	F	0610b486422d4921ae7d2bf64640c50b	112000.0	2017-07-15	2017	7	15	376
3	75	F	78afa995795e4d85b5d9ceeca43f5fef	100000.0	2017-05-09	2017	5	9	443
5	68	M	e2127556f4f64592b11af22de27a7932	70000.0	2018-04-26	2018	4	26	91
8	65	M	389bc3fa690240e798340f5a15918d5c	53000.0	2018-02-09	2018	2	9	167
12	58	M	2eeac8d8feae4a8cad5a6af0499a211d	51000.0	2017-11-11	2017	11	11	257

- transcript.json

A massive file, 306534 rows containing information about all offers and transactions. The *'value'* column contains dictionaries that hold information on whether it is an offer or a purchase, and, in case it is an offer and has been completed, the reward associated.

	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'}

In these dictionaries, both *'offer id'* and *'offer_id'* exist, we will have to unify those values.

This table also seemed at first glance to have duplicated records.

Heavy work was required for this one.

We filtered this table by those transactions performed by users that remained in the previous table.

These tables are related with each other through the customer id and offer id values.

The transcript.json file will need heavy treatment to relate the transactional data and the offers with each other, and also the own offer development through time.

We also normalized the *value* field and extracted the information contained in those dictionaries.

Finally, we also removed all the informational offers.

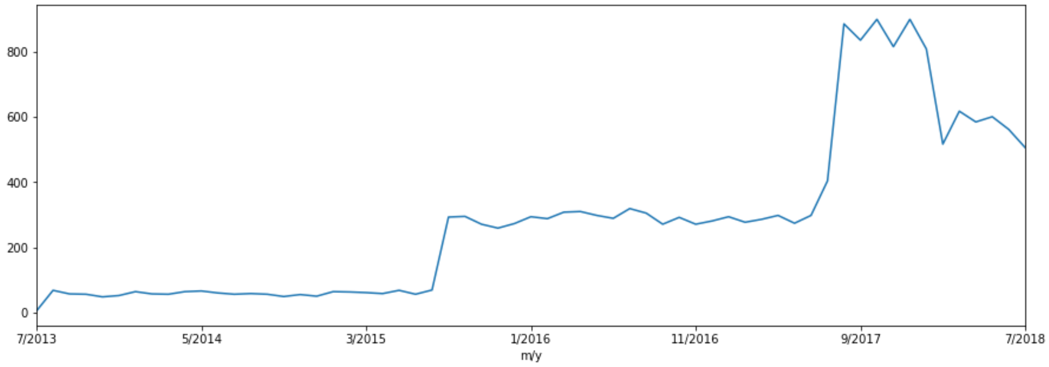
This was our end result:

	event	id	time	amount	offer_id	reward	event_num
66123	offer completed	3dde94fa581145cb9f206624f1a94d5a	168	NaN	2906b810c7d4411798c6938adc9daaa5	2.0	2
66783	offer completed	e9fb6ed2cecb4980ba98c86abc9c91e3	168	NaN	ae264e3637204a6fb9bb56bc8210ddfd	10.0	2
67614	offer completed	a7dc060f6fc94ca7bf71fbb188187dca	168	NaN	9b98b8c7a33c4b65b9aebfe6a799e6d9	5.0	2
68562	offer completed	30478a4c1e884a63a822aa87b833ed7a	168	NaN	2298d6c36e964ae4a3e7e9706d1fb8c2	3.0	2
69218	offer completed	84fb57a7fe8045a8bf6236738ee73a0f	168	NaN	ae264e3637204a6fb9bb56bc8210ddfd	10.0	2

2) Exploratory visualization

Inmediately afterwards, we performed an EDA:

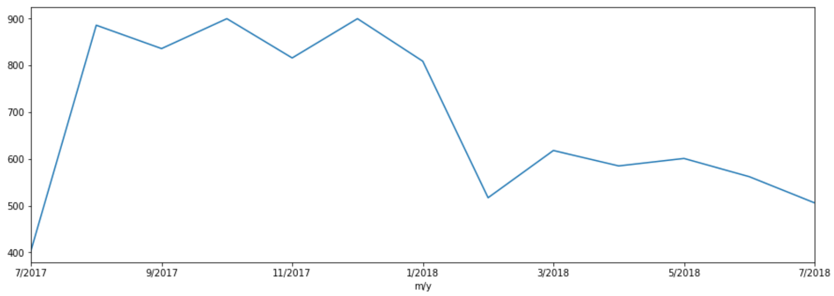
We looked at the distribution of our customers register data over time:



We can see that there have been four different phases in our customer base growth:

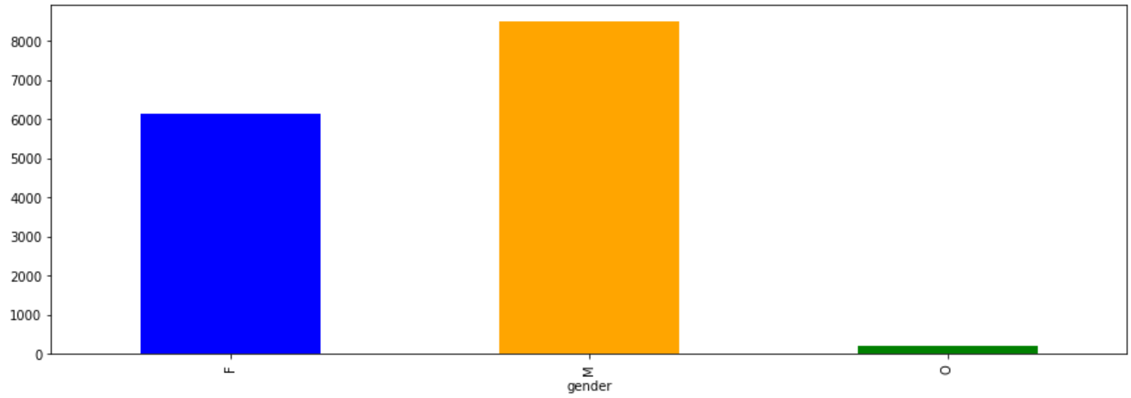
- one initial phase, long ago, in which few people registered
- one intermediate phase, from about year -3 to year -1, with bigger growth
- continued by a phase of explosion
- and a last phase of less great growth, but bigger than in phases 1 and 2

We can see a clear spike or explosion in the number of new clients during our third phase from August of 2017 to January of 2018.

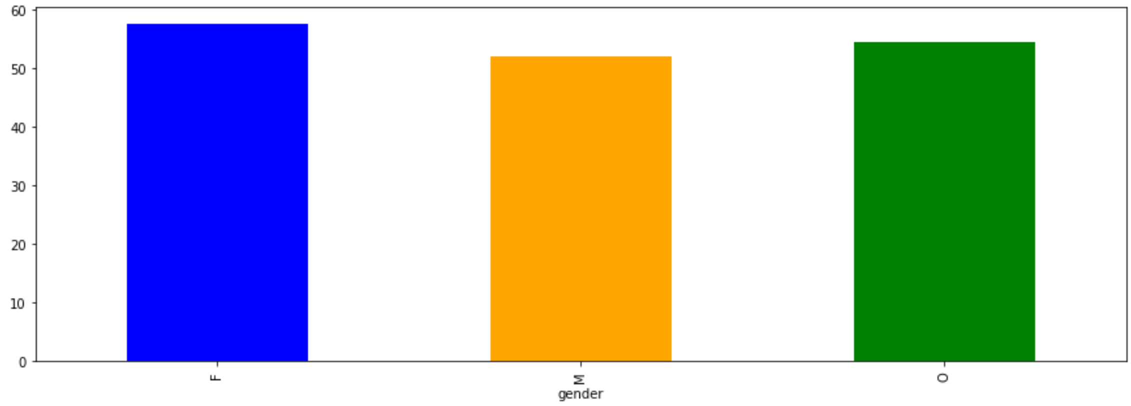


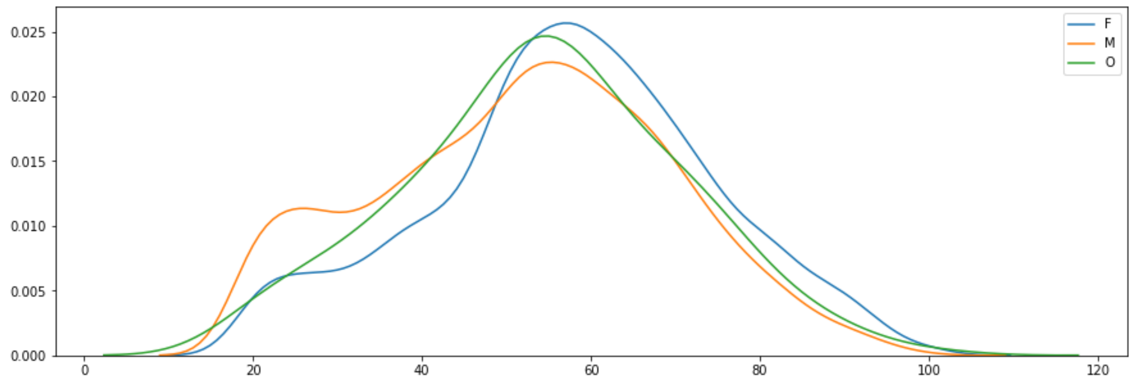
We studied also the demographic information contained:

First, we looked at the gender populations:

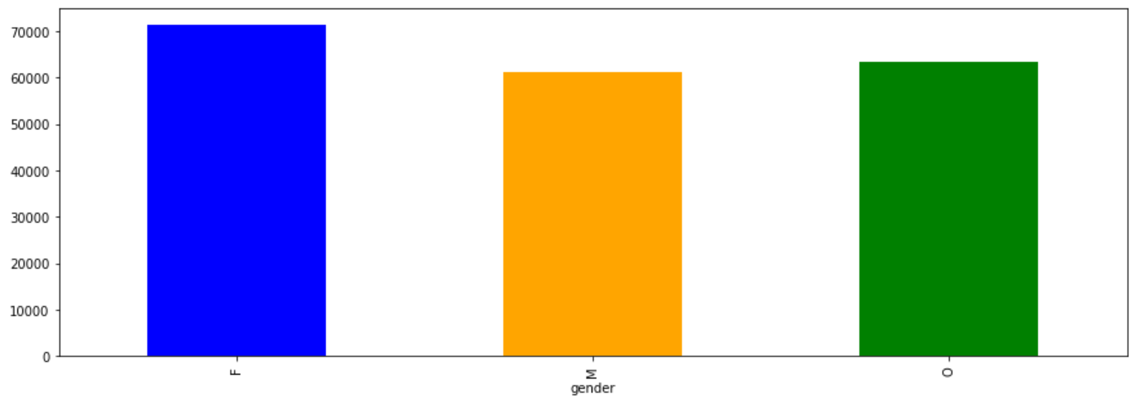


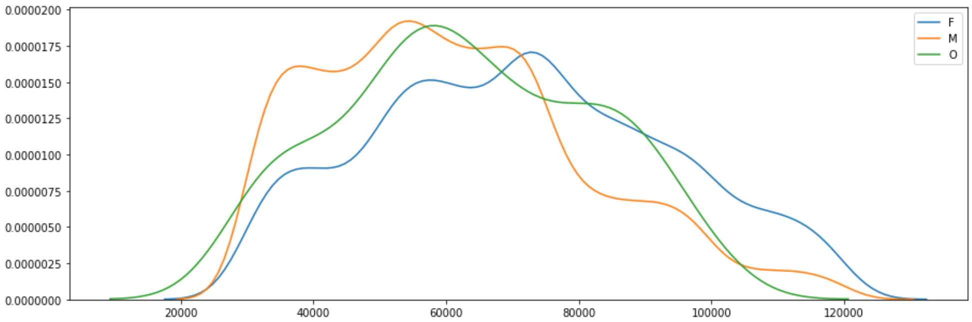
And their age distributions:



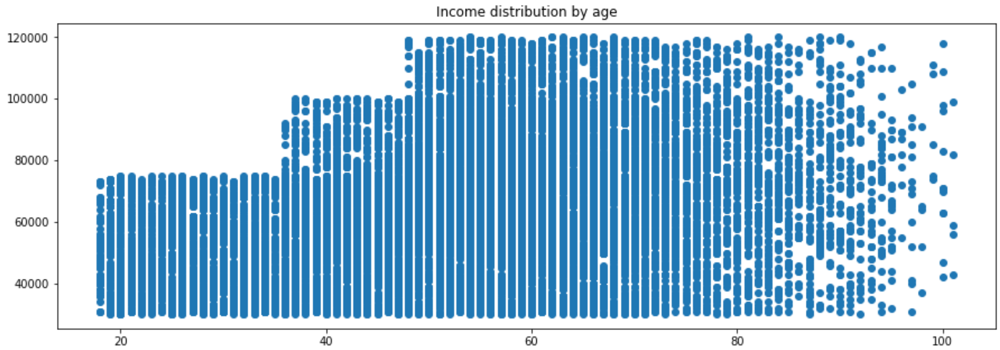


And their income distribution:

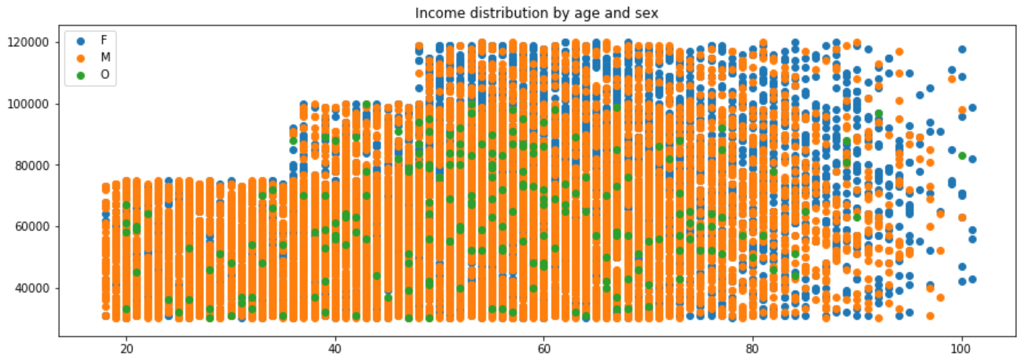




Finally, the income distribution per age:

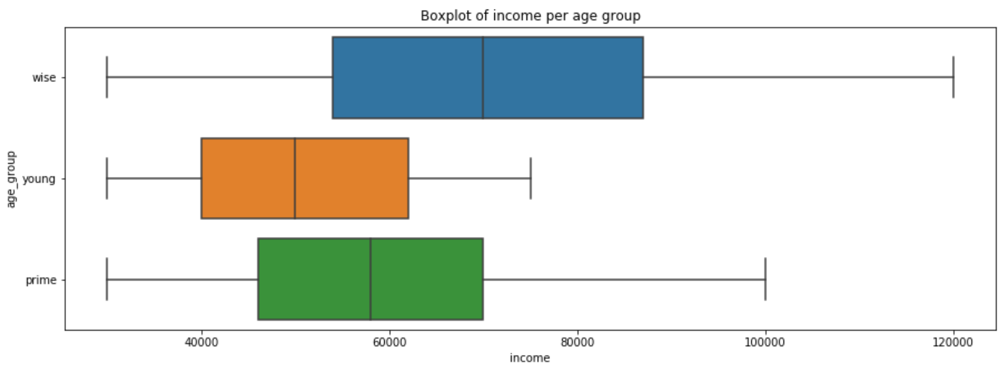


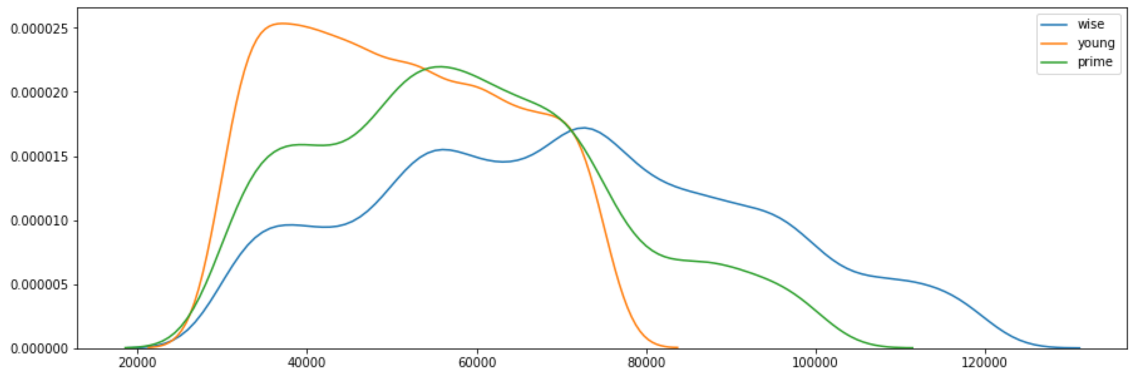
…and per age and gender:



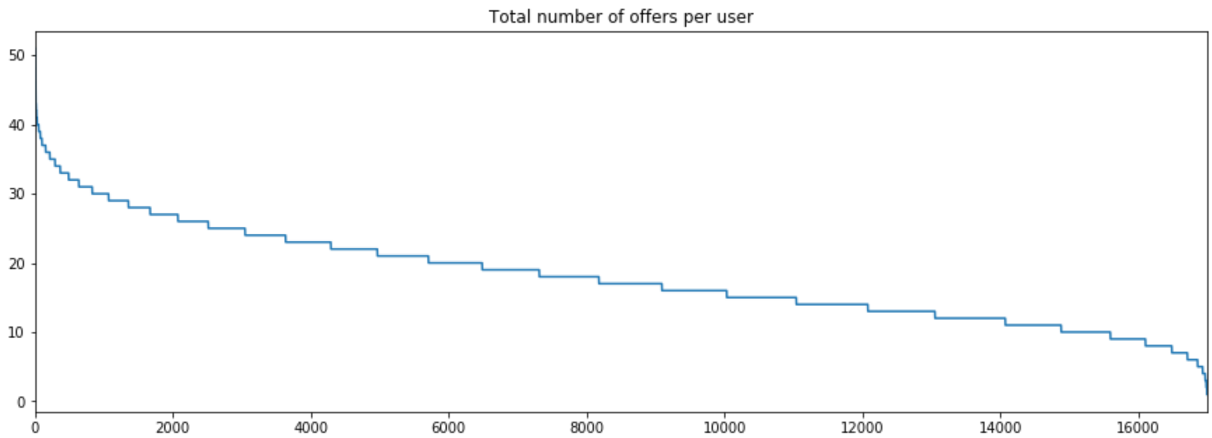
Aha! It seems very clear that some kind of random algorithm was used to randomly assign incomes in the dataset, and that for different age groups, a different upper limit was stablished.

At that point we decided to create three different populations groups based on those “steps”. Here it is the distribution of their incomes:





We checked transactions afterwards. We wanted to check how many offers we had per user:



We saw that not all users were receiving the same amount of offers. In particular, some users had at least 40 registers related to offers or their transaction, while others had just 2.

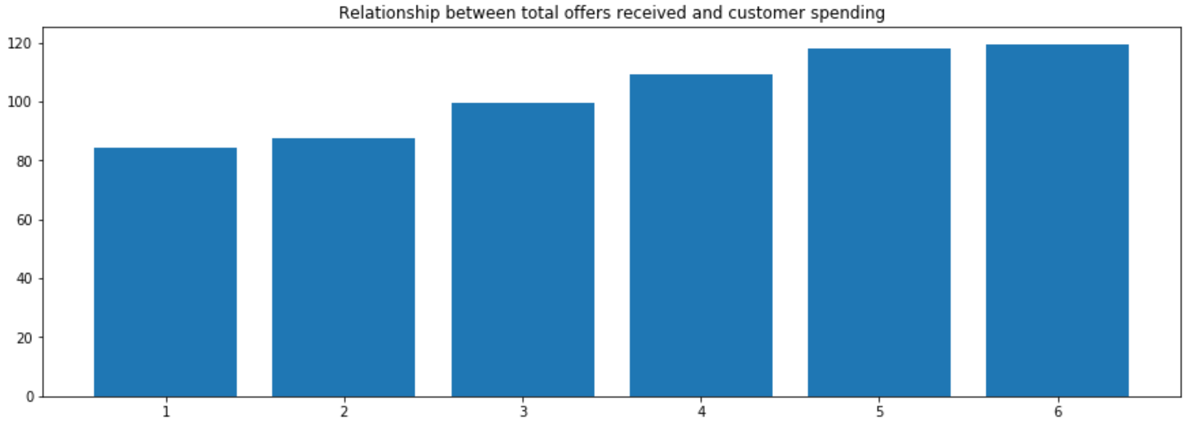
In this point we decided to segregate our dataframe in two:

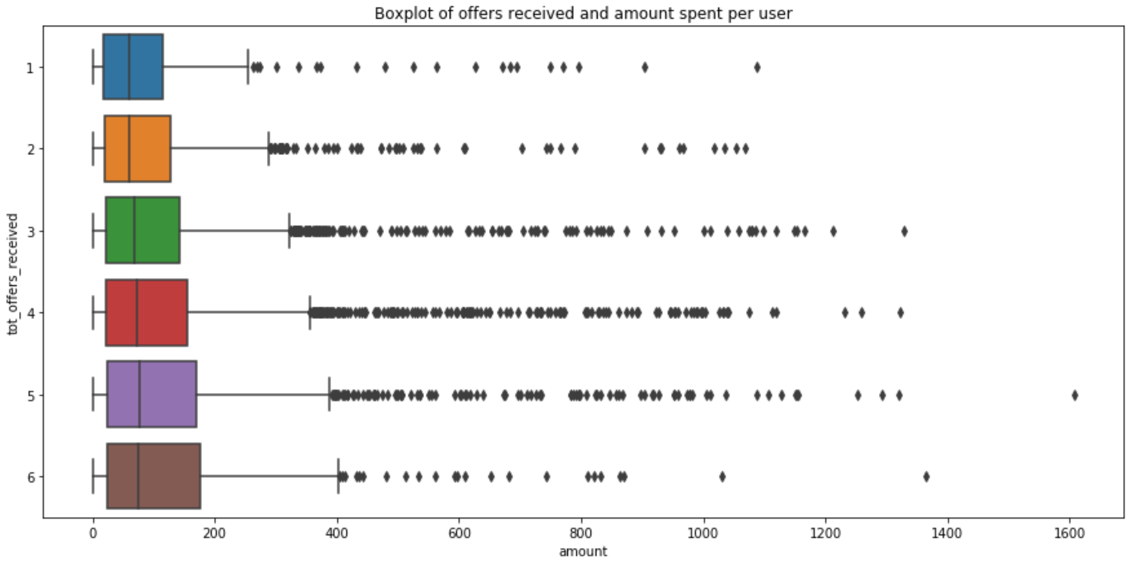
- Offers
- Transactions

	event	id	time	amount	event_num
12654	transaction	02c083884c7d45b39cc68e1314fec56c	0	0.83	-1
12657	transaction	9fa9ae8f57894cc9a3b8a9bbe0fc1b2f	0	34.56	-1
12659	transaction	54890f68699049c2a04d415abc25e717	0	13.23	-1
12670	transaction	b2f1cd155b864803ad8334cdf13c4bd2	0	19.51	-1
12671	transaction	fe97aa22dd3e48c8b143116a8403dd52	0	18.97	-1

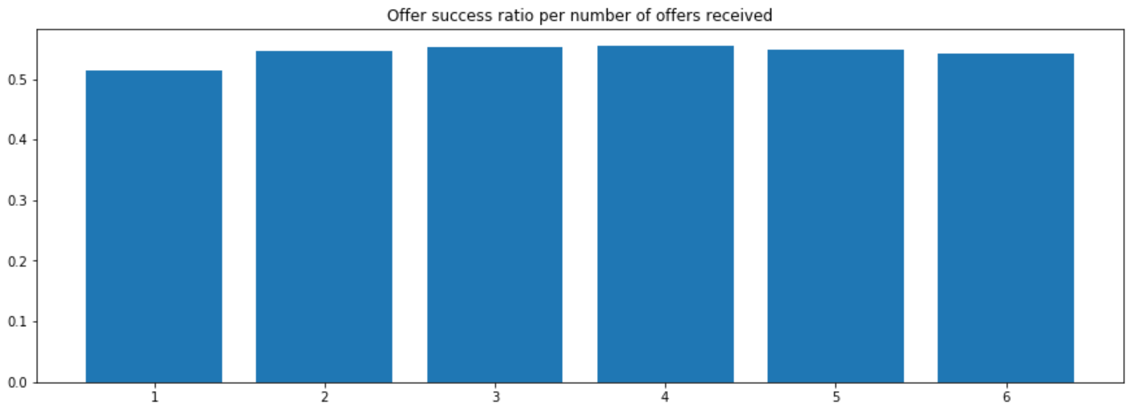
	event	id	time	offer_id	reward	event_num
0	offer received	78afa995795e4d85b5d9ceeca43f5fef	0	9b98b8c7a33c4b65b9aebfe6a799e6d9	0.0	0
1	offer received	a03223e636434f42ac4c3df47e8bac43	0	0b1e1539f2cc45b7b9fa7c272da2e1d7	0.0	0
2	offer received	e2127556f4f64592b11af22de27a7932	0	2906b810c7d4411798c6938adc9daaa5	0.0	0
3	offer received	8ec6ce2a7e7949b1bf142def7d0e0586	0	fafdcd668e3743c1bb461111dcafc2a4	0.0	0
4	offer received	68617ca6246f4fbc85e91a2a49552598	0	4d5c57ea9a6940dd891ad53e9dbe8da0	0.0	0

After that, we were ready to provide some more insight, as, for instance, the relationship between total offers received and customer spending:

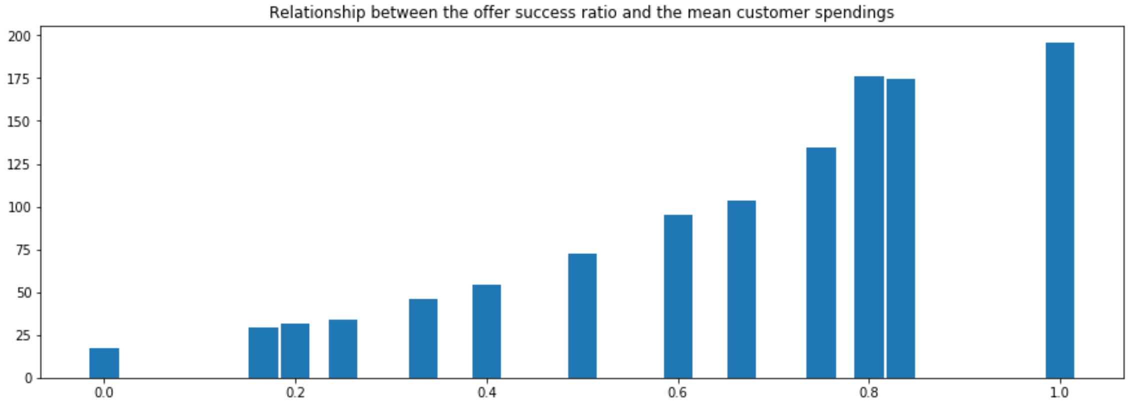


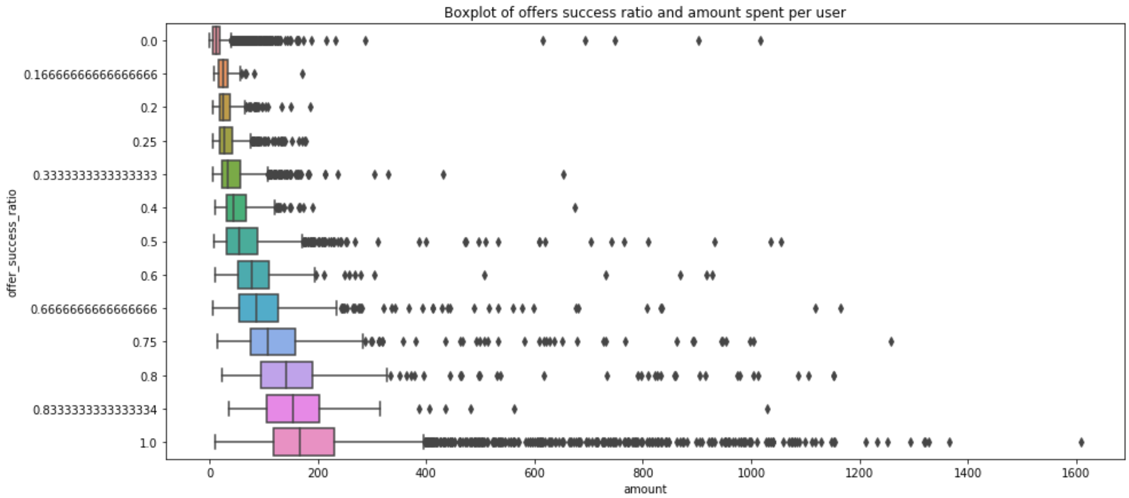


…or to check whether the number of offers sent had any impact on the success rate of those offers:

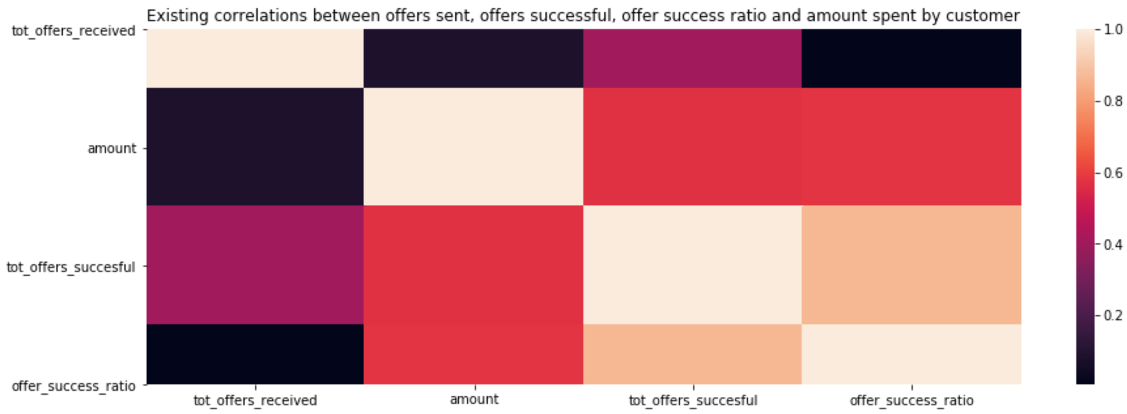


And, finally, one meaningful correlation: the one existing between the offer success ratio and the customer expenditures:





Generalizing, we studied some correlations:



3) Data Exploration (II) - Merging

Subsequently we moved on to do some mergers.

The first one was to divide the offers table in offers sent (those marked with 0) and completed (those marked with 2) and merge the table with itself, in a way that it would be more meaningful.

But first, there was a problem we needed to assess:

As we was, there is not a unique combination of id and offer_id in the tables (as we also saw above), as the same offer has been sent to the same customer several times, in different moments of time.

Also, the same exact offer has multiple values, as it is shown its evolution, from received to completed, going through viewed:

	event	id	time	offer_id	reward	event_num
4301	offer received	d3209835a40a423fbf2c967218d00bcd	0	ae264e3637204a6fb9bb56bc8210ddfd	0.0	0
26706	offer viewed	d3209835a40a423fbf2c967218d00bcd	36	ae264e3637204a6fb9bb56bc8210ddfd	0.0	1
35998	offer completed	d3209835a40a423fbf2c967218d00bcd	72	ae264e3637204a6fb9bb56bc8210ddfd	0.0	2
57537	offer received	d3209835a40a423fbf2c967218d00bcd	168	ae264e3637204a6fb9bb56bc8210ddfd	0.0	0
84070	offer viewed	d3209835a40a423fbf2c967218d00bcd	210	ae264e3637204a6fb9bb56bc8210ddfd	0.0	1
92637	offer completed	d3209835a40a423fbf2c967218d00bcd	240	ae264e3637204a6fb9bb56bc8210ddfd	0.0	2
205881	offer received	d3209835a40a423fbf2c967218d00bcd	504	ae264e3637204a6fb9bb56bc8210ddfd	0.0	0
219621	offer viewed	d3209835a40a423fbf2c967218d00bcd	510	ae264e3637204a6fb9bb56bc8210ddfd	0.0	1
249451	offer received	d3209835a40a423fbf2c967218d00bcd	576	ae264e3637204a6fb9bb56bc8210ddfd	0.0	0
275441	offer viewed	d3209835a40a423fbf2c967218d00bcd	606	ae264e3637204a6fb9bb56bc8210ddfd	0.0	1
277897	offer completed	d3209835a40a423fbf2c967218d00bcd	612	ae264e3637204a6fb9bb56bc8210ddfd	0.0	2
277898	offer completed	d3209835a40a423fbf2c967218d00bcd	612	ae264e3637204a6fb9bb56bc8210ddfd	0.0	2

Before merging both datasets, we need to uniquely identify each offer.

We do so by joining the customer and offer id fields, and adding a counter (starting in 0 for the earliest offer sent) for each time the same offer is presented to the same customer.

We do it so for both dataframes.

The idea in my head is that we are performing a "shuffle":

After that, we could merge the dataframe with itself, and later with the transactions dataframe, and extract some more features, as response times to offers, time from last purchase, mean amount a client expended previously receiving an offer, etc…

We added those features not only to our transactions table, but also, by merging, to our customer profiles table.

And, finally, we could determine the propensity of our users to redeem or let expire our offers:

We finally, did some final touches as, for instance, handling the missing values that appeared in our mergers and data manipulation.

4) Algorithms and techniques

As stated above, we decided that we were going to use two different models: one for determining if a customer was sensible to our offers, and another model, to predict the success or failure of a particular offer.

For the customer model we decided to use:

Two clustering algorithms first, to see whether some clusters naturally appeared from our data. The algorithms selected were:

- DBSCAN
- K-Means

Then, we used a multilineal regression as benchmark, and then we tried out:

- GaussianNB()
- LinearRegression()
- LogisticRegression()
- RandomForestClassifier()

Ending up selecting the last.

For the offers model we decided to use a multilineal regression as benchmark, and then we tried out:

- MultinomialNB()
- BernoulliNB()
- GaussianNB()
- LogisticRegression()
- SVC()
- SGDClassifier()
- RandomForestClassifier()

And we ended up selecting the last.

5) Benchmark

For each model we selected two benchmarks. One coming from the data itself, as primary reference, and the result in the same metric (accuracy), coming from applying a simple machine learning model to the same data (multilinear regression).

Customer model:

General accuracy:

- n_{sc} = number of susceptible clients
- n_{tc} = number of total clients

$$SR_{clients} = \frac{n_{sc}}{n_{tc}}$$

Value: 53%

Benchmark model accuracy:

The multilinear regression was capable of a 75% accuracy on that same data, and up to 90% on the sub selection (1,0) of our customers.

Offers model:

General accuracy:

- n_{or} = number of offers redeemed
- n_{os} = number of offers sent

$$SR = \frac{n_{or}}{n_{os}}$$

Value: 61%

Benchmark model accuracy:

The multilinear regression was capable of a 85% accuracy on that same data.

III. Methodology

1) Data Preprocessing

Before commencing with our models, we had to still do some further data manipulations.

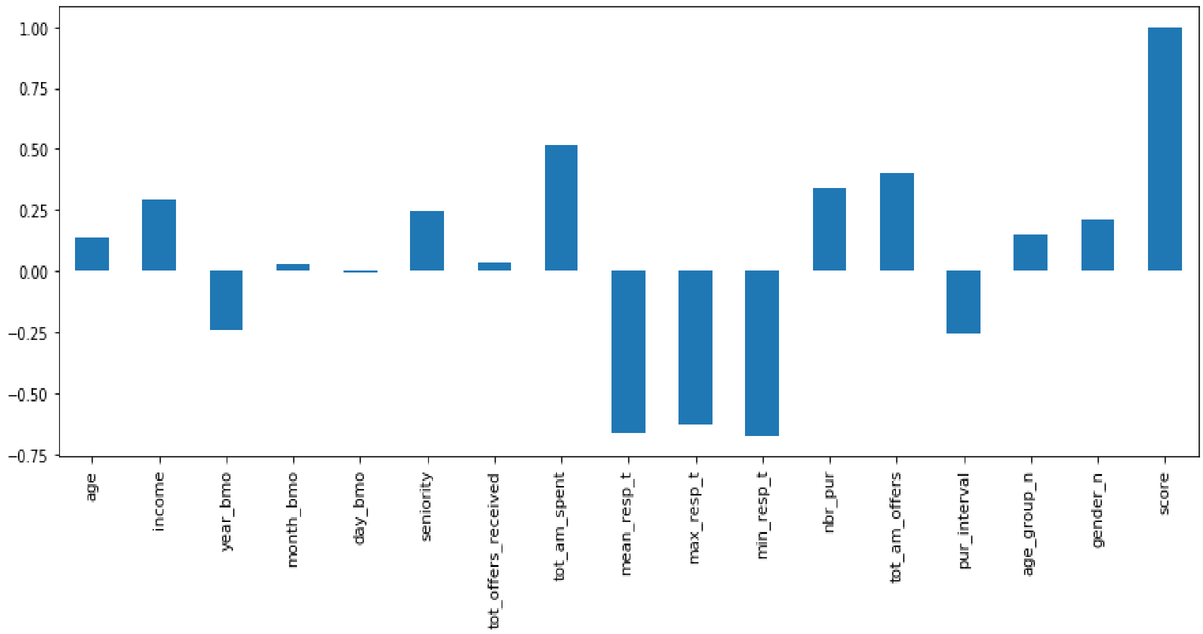
In both cases studied the existing correlations in our data, both with the target variable and the correlations among features, and normalized it previously to feed it to our models.

Also, for our customer data, we performed a PCA and also used two clustering algorithms that helped us to extract some characteristics and further explore the information contained in that data.

We will go on by case:

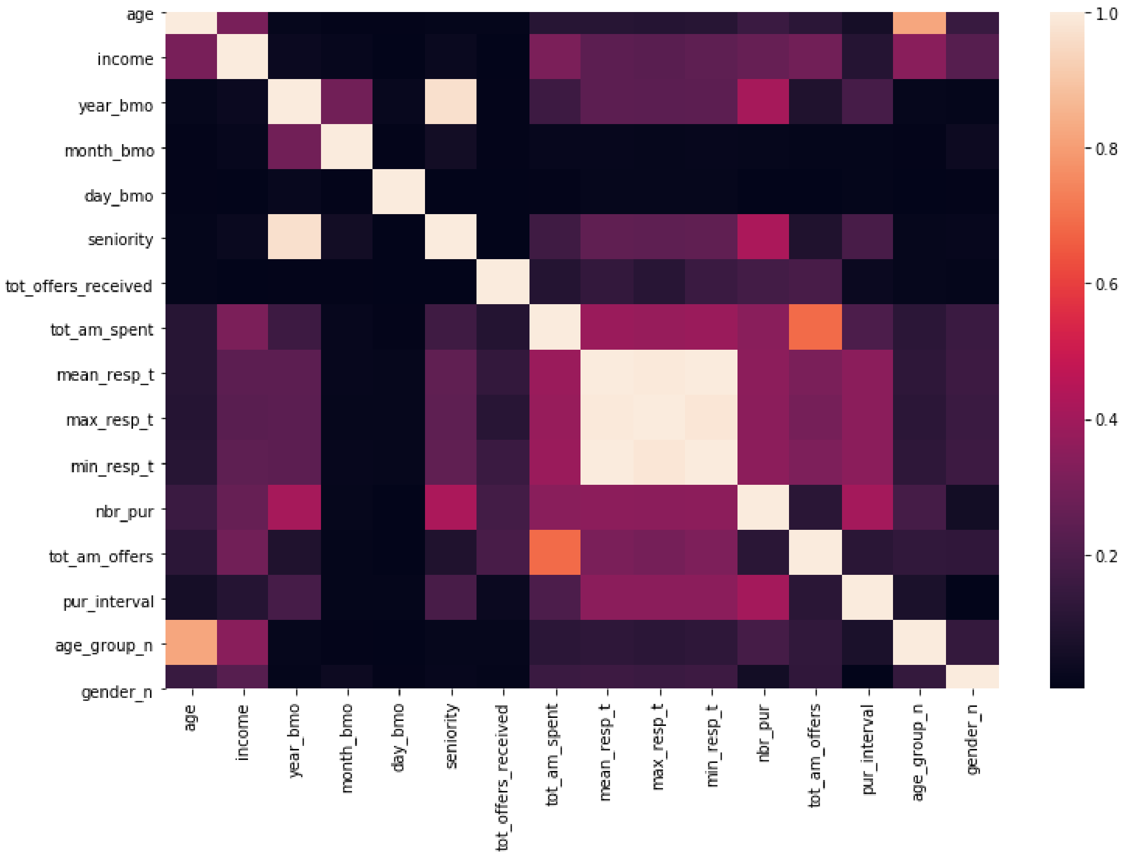
Customer model:

We examined our existing correlations:



We also plotted some groupings to see if we could “see” further hidden relationships, without much success.

We also studied the existing correlations among our variables:

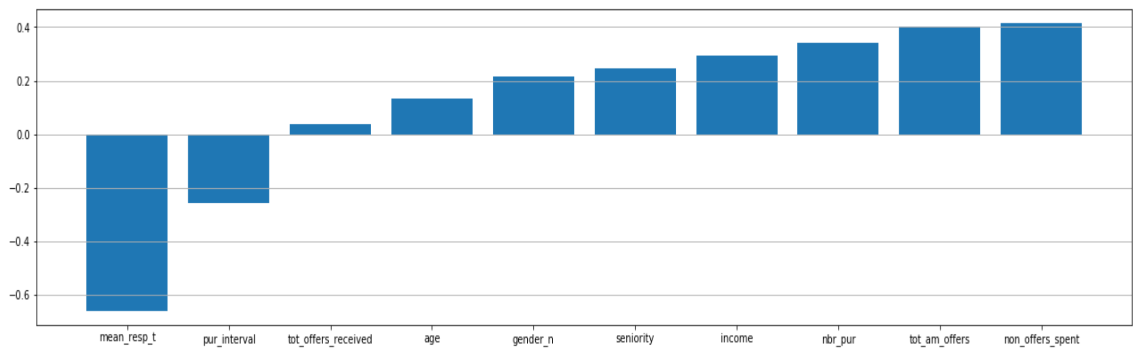


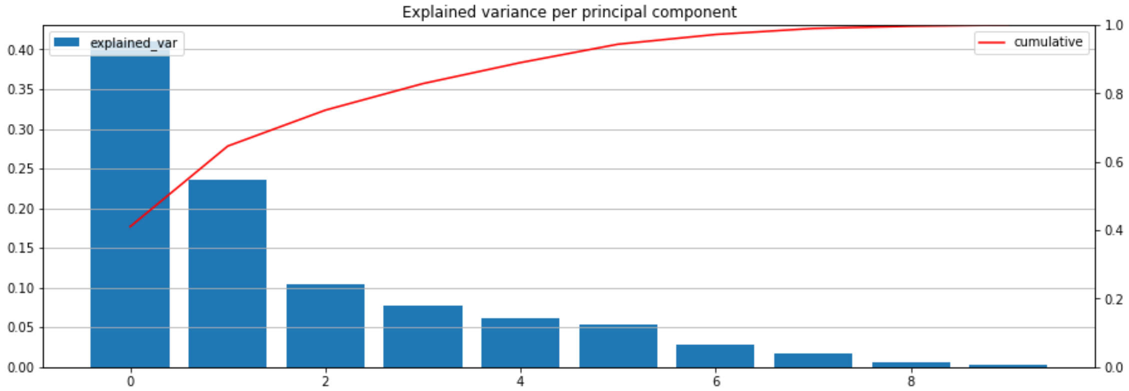
We decided to drop some heavily correlated features, and try to modify another 2 (*tot_am_spent* and *tot_am_offers*), so they would be less correlated:

Particularly, we created:

- non_offers_spent = tot_am_spent - tot_am_offers
- offers_spend_share = tot_am_offers / tot_am_spent

Then, we proceed to perform a normalization (using MinMaxScaler) and performed some analysis on the normalized data:





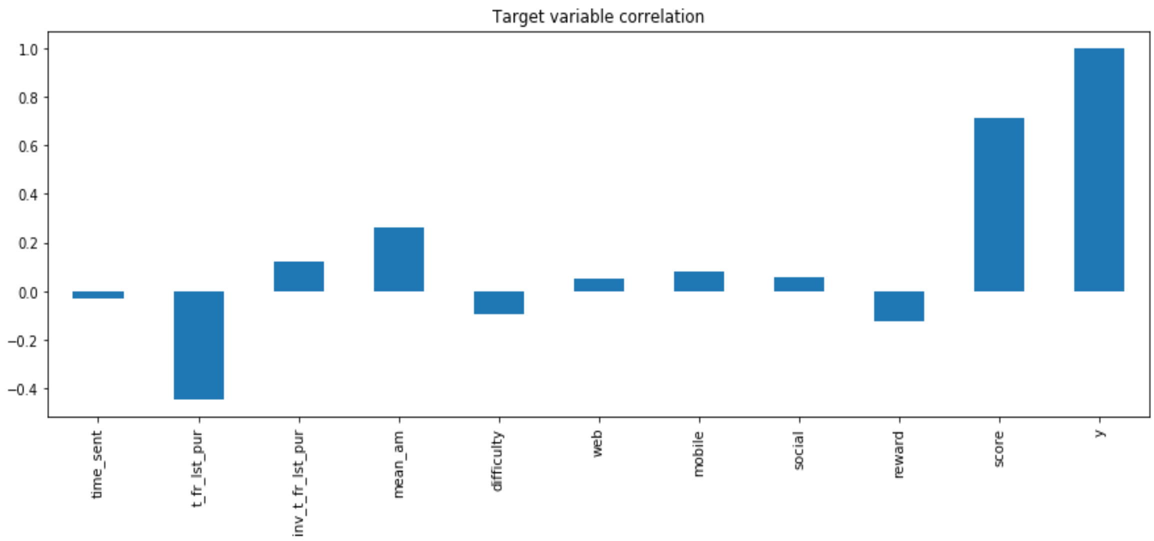
The PCA analysis revealed that no big gains could be expected from changing the base of our components.

Offers model:

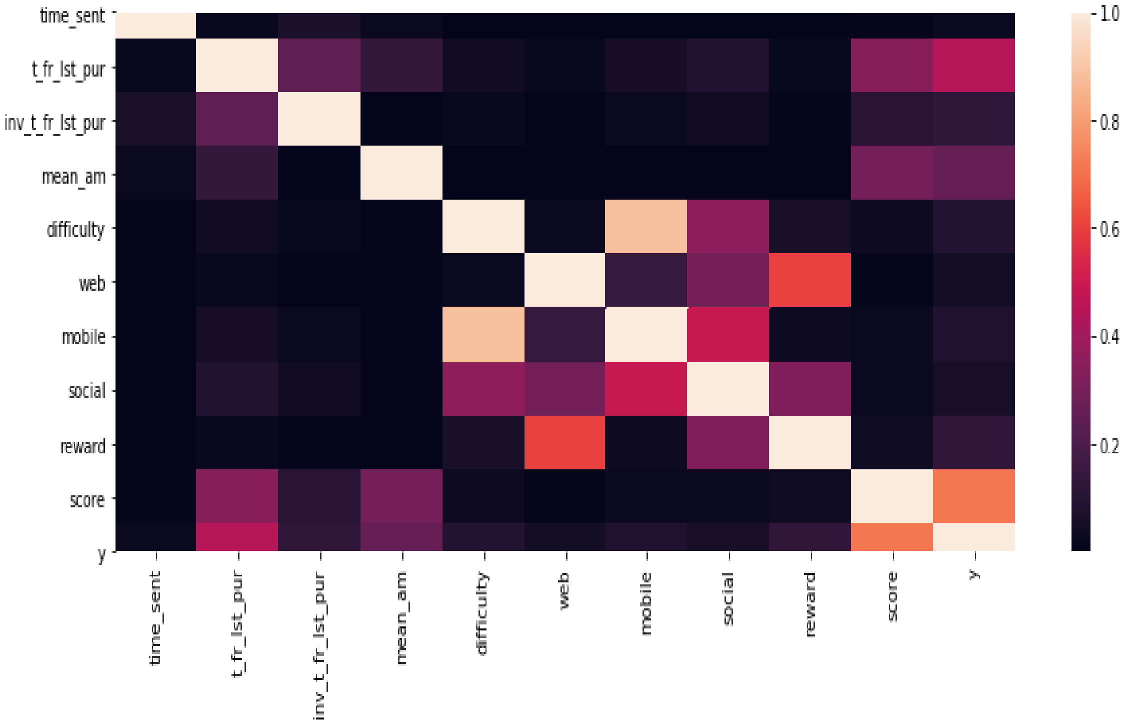
In this case, after dropping some columns due to different reasons (text values, containing information that was almost target-variable explanatory or simply, in the case of the email column, because there was no information to be extracted, as it was all filled with 1s)

	time_sent	t_fr_lst_pur	mean_am	difficulty	web	mobile	social	reward	score	y
0	0	54.0	24.607500	10	1	1	1	2	1.0	1
1	0	204.0	5.090000	10	0	1	1	10	0.0	0
2	0	132.0	28.828000	5	1	1	0	5	1.0	1
3	0	12.0	25.008000	10	0	1	1	10	1.0	1
4	0	132.0	3.701765	10	1	1	0	2	0.5	1

we analyzed the correlation of all features to our target variable:



And their existing inner correlations:



…and we went straight to normalize our data.

2) Implementation

We had decided to divide our problem in two parts, having each their corresponding model:

- The first part is to determine which customers are offer-sensitive. We have created a model that predicts the user *"type"* in relation with our offers with an accuracy of 80% (when considering all users) or 95% (when we only adjust to those clients we are sure enough and have enough data about them).

- The second model determines, starting from the characteristics of an offer (including the recipient's *"type"*), the success of that offer. In this regard, we have obtained accuracies of 96%, which is a very good result, as it almost allows us to pinpoint our offers

So, our first model to be developed and trained should be the **Customers model**. Then, we would proceed with the **Offers model**.

Up to this point, for each, after the data preprocessing we had a treated DataFrame for each case.

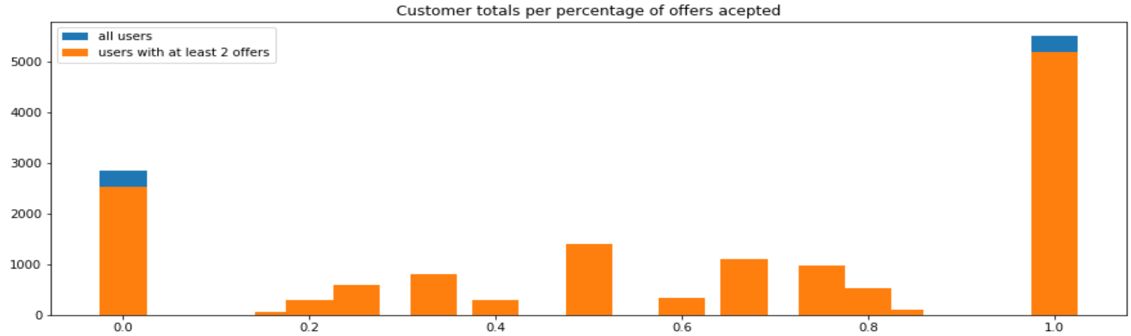
We needed to separate our features, from our target variable, as they are fed independently.

Also, it is convenient to reserve part of your data and use it for testing. This would help us avoid overfitting.

So the final stage was to divide the dataframes in 4 parts: x_train, y_train, x_test and y_test.

To do so, we used the *train_test_split* function already provided by scikit-learn. We decided to use a particular random state in order to make our results reproducible, and decided to use a 70% of our data for training, and 30% for testing, close to the 75%-25% the tool has by default.

For the **Customers model** we had a particular situation. In the moment that we defined our target variable (the *score* variable), we had a continuous variable, and we discretized it:



The original variable was the offer success rate. Now, in the discretization, we assigned the value 0.5 to those users that:

1) Were in the mid range
2) Had received one offer or less

Now, at this point, we had that our original benchmark accuracy:

- n_{sc} = number of susceptible clients
- n_{tc} = number of total clients

$$SR_{clients} = \frac{n_{sc}}{n_{tc}}$$

Of 53% was calculated with all our customer base.

Also, we had the curiosity of seeing whether the models could learn more from all the data, keeping the 0.5 customers.

So, four our **Customers models** we had 2 original dataframes: one with all values (ternary case, labels: 0, 0.5 and 1) and one dropping the dubious cases (binary case, labels: 0 and 1). We wanted to inspect the results of the models we ran on them on both scenarios, and check the differences.

So, for this scenario, 2 training sets and 2 test sets were constructed.

Now, for all these models and situations, the metric of our choice was the accuracy (keeping always an eye on the confussion matrix).

Appart from these complications, the coding was simple, and all functions created, were done just for the purpose of not duplicating the code.

3) Refinement

For both models, the final selected model was a Random Forest Classifier.

As, the comparison of the resulting accuracies between the train and test revealed some overfitting, we decided to fine tune these models.

We decided to fine tune the following hyperparameters:

- max_depth = [2,3,4,5]
- min_samples_split =[2,3,4,5]
- min_samples_leaf = [1,2,3,4,5]

And we used the *GridSearchCV* tool provided within scikit-learn.

As a result, our model got more robust, and we lost some accuracy in the train, and gained it in the test, disappearing the difference that signaled the overfitting.

IV. Results

1) Model Evaluation and Validation

Customers model

The general benchmark for this model was about 53% (that meant, considering all clients were susceptible to our offers).

In the case of our customer characterization, we have also run some clustering algorithms that allowed us to get some insight in which of our customers are more prone to redeem our offers, and create a persona for those clients.

As we have seen, we have been able also to use those clusters to create a minimodel over them, that yielded about 70% (considering all clients) or 85% (when dropping the dubious) accuracy, both when using DBSCAN and KMeans algorithms for such clustering.

This already meant a significant improvement from our current situation.

But we could do better: a simple linear regression yielded results of 75% and 90%.

And, finally, a tuned random forest classifier took us to results of 80% and 95% in terms of accuracy.

This is a big improvement from our current situation.

The model also performs well when compared to the linear regression set as benchmark.

Finally, the fine tuned model has shown that it has corrected the overfitting present in the first Random Forest Classificator, so we can be confident that it will performed as estimated.

Offers model

For our general model, we started from a general benchmark of 61% (considering the previous success rate).

A simple linear regression took that accuracy up to the 85% range, and also showed us that the most important features when determining the success of an offer were:

- the time past from that client's last purchase
- the mean ammount the client has spent up to the moment of the offer
- the offer reward

- the client's score (0, 0.5 or 1)

Again, after running a battery of models, we selected a random forest classifier as the model of our choice. That model, after fine tunning, got the accuracy up to 96%.

This is a big improvement from our current situation.

The model also performs well when compared to the linear regression set as benchmark.

Finally, the fine tuned model has shown that it has corrected the overfitting present in the first Random Forest Classificator, so we can be confident that it will performed as estimated.

2) Justification

As we have just seen, our model performance exceeds both our original benchmark, and the results obtained with our benchmark model.

This happens for both models performed:

Customers model:

Model \ Data	All	Without uncertain values
Benchmark value	53%	---
DBSCAN/K-Means	70%	85%
Linear Regression	75%	90%
Random Forest Classifier	80%	95%

Offers model:

Model \ Data	All
Benchmark value	61%
Linear Regression	90%
Random Forest Classifier	96%

Also, for both models (especially, the second one) the results are good enough to justify their implementation.

In particular, an accuracy of 96% means that, for this use case, we could practically pinpoint our offers, after assessing their success ratio.

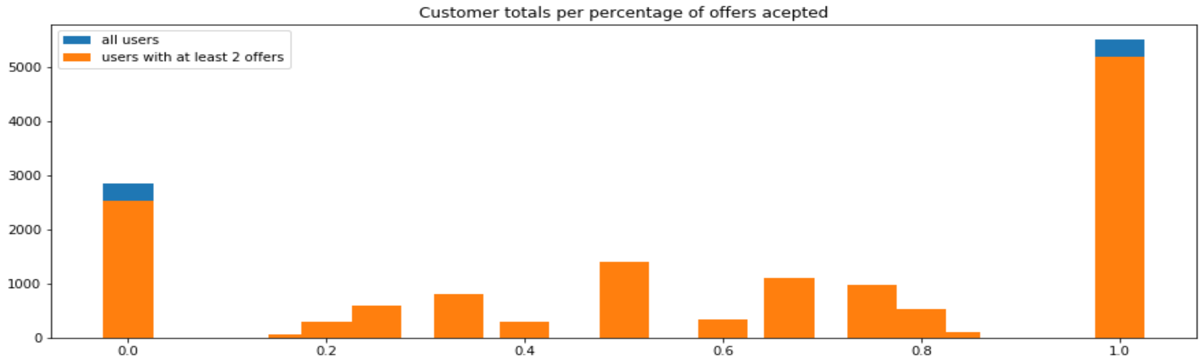
But also, an 80% accuracy on all our clients, or 95% when cherrypicked are really nice results, that would also allow us to send our offers directly to those interested, with little harm.

V. Conclusion

1) Free-form visualization

Along this project, not only we have constructed two useful models, but we have also got a lot of useful insight from our data.

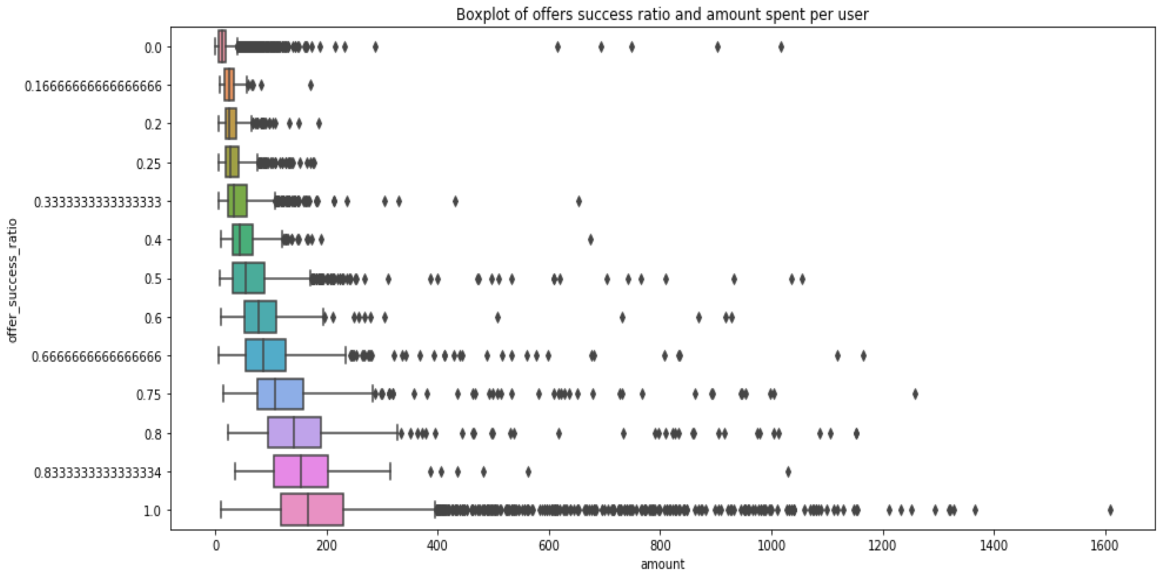
If I had to choose a visualization that represented the project, I would pick this one:



That represents the consumer behavior in relation to the offers, and the big sectorization (could we say, radicalization?) existing.

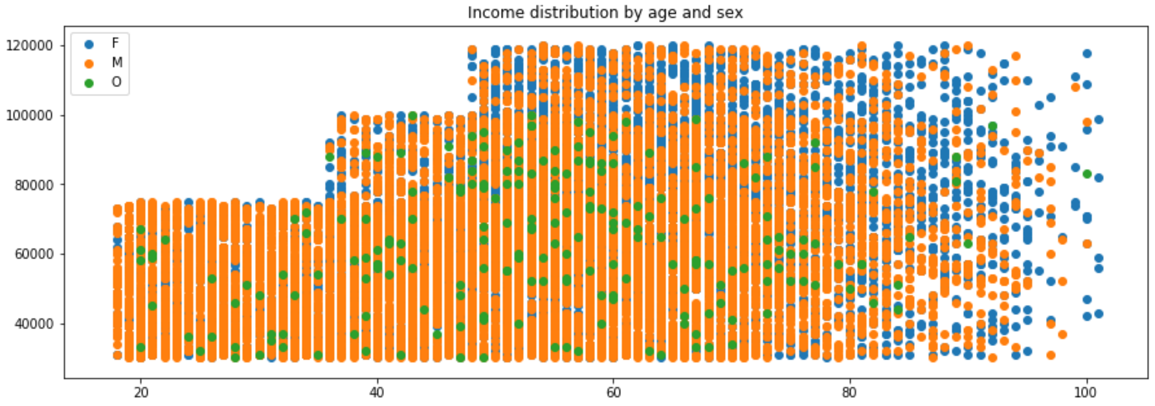
This lead me to think that the clustering of the customer base was possible.

Also, for their beauty, boxplots are always interesting:



Also, in this particular case, this boxplot was particularly informative on the relationship between the success of an offer, and the amount spent by the customer. This relationship led me to introduce expenditure features.

Although, for its beauty, and also, because I found it fun, and because it showed clearly that the data had been generated with some form of algorithms, I would choose this other one:



2) Reflection

Along this project we have tried to perfectionate the offer campaigns of a coffee retailer.

Starting from the information on 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

we have cleaned and treat that information along an excruciating, never-ending process, and used it to feed two models:

a first model that tells us whether a client is interested in our offers or not

a second model that tells us whether a particular offer will be successful

Our models have obtained accuracies in the 95-96% range, when using a random forest classifier.

For this particular problem, as the failures should not be very harmful, this means almost pinpointing our offers and our interested customers.

What is best, it improves our current situation dramatically.

Finally, I wanted to state that this project reveals the true power of machine learning algorithms for dealing with certain kind of problems, and the reason for their present date popularity.

3) Improvement

There are certain features we could create both for the customers and offers datasets, and feed our models.

Also, dimensionality reduction via PCA, or just by keeping the most prominent features, was not performed, as it was revealed that was not necessary. Nevertheless, it is always a path worth testing.

Finally, the author has the concern of how would impact the results, if, instead of considering the client behaviour as discrete (0 and 1, with 0.5 as mark for 'dubious'), a continuous variable was used instead.

This could take the prediction on the offers model beyond.

Nevertheless, it would mean that the metrics for assessing the customers models employed would become less intuitive (which was the reason for the discrete approach in the first place).

Indeed, those issues will be addressed in a second iteration on the project.