Machine Learning Engineer Nanodegree

Udacity

Capstone Project Report

Sairam Iyengar

November 7th , 2020

Table of Contents

I. Definition

a. Project Overview

This project derives from the direct marketing system which Starbucks uses to keep in touch with its customers.

Aiming to incentivize and reward the customers registered in its platform, Starbucks periodically sends individual messages containing offers related to its products.

There are three types of offers that can be sent: buy-one-get-one (BOGO), discount, and informational. In a BOGO offer, a user needs to spend a certain amount to get a reward equal to that threshold amount. In a discount, a user gains a reward equal to a fraction of the amount spent. In an informational offer, there is no reward, but neither is there a requisite amount that the user is expected to spend.

Offers can be delivered via multiple channels: e-mail, social media, on the web, or via the Starbucks's app.

Nonetheless, marketing campaigns have associated costs. Hence, to be considered a successful campaign, it must generate profit higher than that initial cost. That means, companies expect to have a return on investment (ROI) as high as possible.

Thus, companies do not want to spend money sending offers to customers that are not likely to buy their products. On the other hand, new customers need to be attracted, so it is necessary to identify who are the people with a higher probability to respond to a marketing campaign.

Sometimes, recurrent consumers deserve some reward so they can feel appreciated and not forgotten. However, some customers keep coming back even if they do not receive offers. To give an example, from a business perspective, if a customer is going to make a 10 dollar purchase without an offer anyway, you would not want to send a "buy 10 dollars, get 2 dollars off" offer, unless this relative short-term loss means a more satisfied customer who will consume more in the future.

Another case is those customers who only buy products when receiving some reward, while other ones are opposed to marketing campaigns and do not want to be contacted at all.

Those are a few examples that illustrate how complex is the marketing decision process that has been faced by companies for years.

Considering the recent advances of artificial intelligence and the massive amount of data gathered over the years, this is a topic that could be widely improved by intelligent systems owing to the fact that they can analyze a large

amount of data and understand patterns sometimes hidden for the human perception.

Personally, this is a very appealing subject since proposing better marketing campaigns may benefit not only the companies by raising their profit, but also the customers who receive more relevant offers in accordance with their consumer behavior. Perhaps, this win-win situation is the key to maintain the economy growing while people can afford better services and products.

The data set used in this project is provided by Udacity and Starbucks as part of the Machine Learning Engineer Nanodegree program. It contains simulated data that mimics customer behavior on the Starbucks rewards mobile app.

The program used to create the data simulates how people make purchasing decisions and how those decisions are influenced by promotional offers.

Each person in the simulation has some hidden traits that influence their purchasing patterns and are associated with their observable traits. People produce various events, including receiving offers, opening offers, and making purchases.

As a simplification, there are no explicit products to track. Only the amounts of each transaction or offer are recorded.

b. Problem Statement

Starbucks, as well as any other company, invests money in marketing campaigns expecting to have a profit higher than the assumed before. So, identifying the most relevant offer to the correct customers is crucial for a successful campaign.

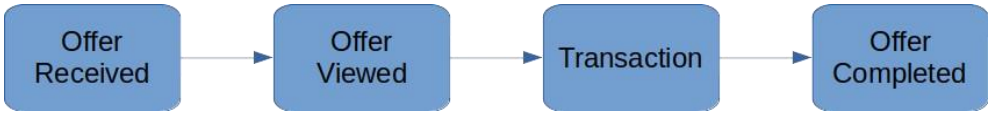
However, some targeted customers do not even see the offer sent to them, which may be a problem with the channel chosen. Other ones do not buy anything, despite seeing the offer, what might be a problem with the offer type sent, or maybe that is not a customer to be considered as a target.

There are other cases where the customers identify themselves with the offer, which leads them to try new products or spend more money than usual. Those are the situations to be identified and pursued.

The problem this project proposes to solve is finding the most appropriate offer for each one of the customers, which means finding the offer that is more likely to lead the customer to buy Starbucks products.

In the context of this project, an appropriate offer is that one where the customer sees the offer received and buys products under its influence, completing the offer lifecycle.

Appropriate offer



If a customer does not see an offer, it is not an appropriate one. If he or she sees the offer but does not complete it, it is not appropriate as well, since it did not lead the consumer to buy products. Similarly, if the customer buys some products, completes an offer, and receives a reward before visualizing that offer, it is not considered effective because the customer was not under the influence of that offer when decided to make a purchase.

In order to face the problem stated above, this project proposes to apply machine learning techniques to study customers' behavior by analyzing the transcriptions of their relationship with Starbucks.

More specifically, a neural network will be trained to predict how customers may react when receiving each one of the available offers: if they will complete the offer cycle or not. So, it will be possible to identify which one is more suitable for each customer.

Since consumers' behavior is not a feature isolated in the time, the next actions are affected by past experiences. Regarding this time-dependency, this project proposes to build and train a Recurrent Neural Network (RNN) as the central piece of the solution, aiming to analyze the customer behavior through the time.

The final result of this project is a direct marketing system that, given a customer, is able to predict the likelihood of each offer be completed.

The theoretical workflow for approaching the solution stated includes several machine learning techniques, following the guideline sections below.

i. Data loading and exploration

Load files and present some data visualization in order to understand the distribution and characteristics of the data, and possibly identify inconsistencies.

ii. Data cleaning and pre-processing

Having analyzed the data, handle data to fix possible issues found.

iii. Feature engineering and data transformation

Prepare the data to correspond to the problem stated and feed the neural networks. The transcription records must be structured and labeled as appropriate offer or not.

iv. Splitting the data into training, validation, and testing sets

Prepare three datasets containing distinct registers within each one.

The largest dataset is employed to train the networks, while the validation set, to evaluate the models during the training phase.

The testing set contains data never seen before by the networks, so it will be possible to consider this dataset as being new interactions between Starbucks and customers. By using this dataset, it will be possible to measure the final performance and compare the results of the trained models.

v. Defining and training a Feed-Forward Neural Network

Training of the benchmark model.

vi. Defining and training a Recurrent Neural Network

Training of the proposed model.

vii. Evaluating and comparing model performances

Comparison between the accuracy of both network models to verify each one is more suitable to solve the problem stated.

viii. Presenting predictions for offer sending

Present the resulting predictions, along with discussions on how this system should be employed.

c. Metrics

The accuracy of the models will be measured to evaluate the performance of the networks. By using the same metric, we are able to quantify and compare both the benchmark and the final models.

Considering that customers might have variations in their standard behavior, having an accuracy very close to 100% might indicate that the network just memorized the customers' behavior instead of understanding their consumption patterns.

On the other hand, having too low accuracy also might indicate that the network was not able to learn general patterns.

Hence, the target accuracy for the RNN in this project is about 80%.

$$\texttt{accuracy}(y, \hat{y}) = \frac{1}{n_{\text{samples}}} \sum_{i=0}^{n_{\text{samples}}-1} 1(\hat{y}_i = y_i)$$

II. Analysis

a. Data Exploration and Exploratory Visualization

This section presents some data visualization in order to understand the distribution and characteristics of the data, and possibly identify inconsistencies.

Data exploration is one of the most important parts of the machine learning workflow because it allows you to notice any initial patterns in data distribution and features that may inform how to proceed with modeling and clustering the data.

Data exploration uses visual exploration to understand what is in a dataset and the characteristics of the data. These characteristics can include size or amount of data, completeness and correctness of the data, and possible relationships amongst data elements.

The data for this project is contained in three files:

i. portfolio

Offer ids and meta data about each offer sent during 30-day test period. Size: 10 offers x 6 fields

- 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)

	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

There are three types of offers that can be sent: buy-one-get-one (BOGO), discount, and informational:

- In a BOGO offer, a user needs to spend a certain amount to get a reward equal to that threshold amount.

- In a discount, a user gains a reward equal to a fraction of the amount spent.
- In an informational offer, there is no reward, but neither is there a requisite amount that the user is expected to spend.

Offers can be delivered via multiple channels: email, social media, on the web, via the Starbucks's app.

Every offer has a validity period (duration) before the offer expires. We see that informational offers have a validity period even though these ads are merely providing information about a product. Here, the duration is the assumed period in which the customer is feeling the influence of the offer after receiving the advertisement.

As we can see, *offer_type* and *channels* are presented as a categorical values, which must be converted to columns (one hot encoding) before passed into a neural network.

In this portfolio, any offer is sent by email, so this is an informative feature and might be filtered out when feeding the neural networks.

Features *reward*, *difficulty*, and *duration* present values in different ranges. It is a good practice to scale then to the same range than other features.

Apart from that, no issue is noticeable in this dataset.

ii. profile.json

Demographic data for each rewards program users Size:

17000 users x 5 fields

- gender: (categorical) M, F, O, or null
- age: (numeric) missing value encoded as 118
- id: (string/hash)
- became_member_on: (date) format YYYYMMDD
- income: (numeric)

	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 dataset presents *gender* as a categorical feature which should be mapped to columns (one hot encoding).

Analyzing the missing values

	count	mean	std	min	25%	50%	75%	max
age	17000.00	62.53	26.74	18.00	45.00	58.00	73.00	118.00
became_member_on	17000.00	20167034.23	11677.50	20130729.00	20160526.00	20170802.00	20171230.00	20180726.00
income	14825.00	65404.99	21598.30	30000.00	49000.00	64000.00	80000.00	120000.00

At a glance, it is possible to notice empty values in columns income and gender, as well as missing value encoded as 118 in the column age.

By analyzing dataset, it is possible to conclude that all the missing values occur in the same rows. This may have several possible causes, for instance:

- the interface where those profiles were created might not have requested those information to be input, or
- those users may not have agreeded with the privacy policy.

In addition to that, those users have been signing up all over the analyzed period in the same fashion than the other users.

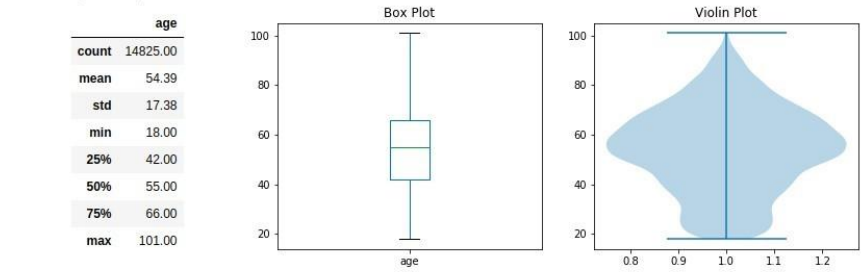
Altough those users could be filtered out of the data set, they might have some peculiar behaviour which would be interesting to be analyzed. Thus, my approach with those record is:

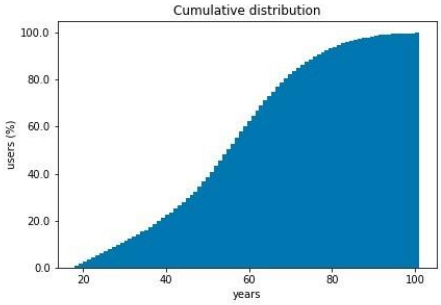
- maintain those records in the data set
- mark those users as belonging to a particular class, by creating a new feature for that
- since gender is a discrete feature, create another category: *unknown 'U'*
- as age and income are continuous features, fill them with the respective mean values

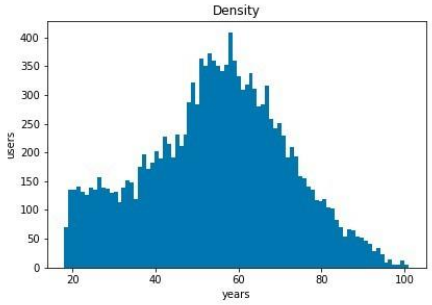
Explore ages

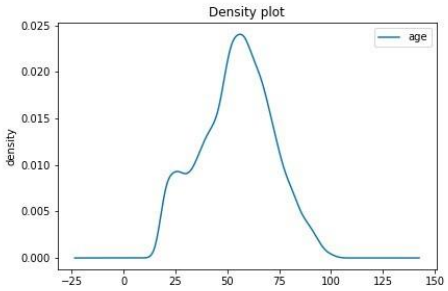
Ages description

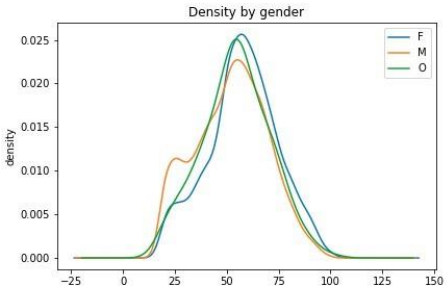
	age
count	14825.00
mean	54.39
std	17.38
min	18.00
25%	42.00
50%	55.00
75%	66.00
max	101.00









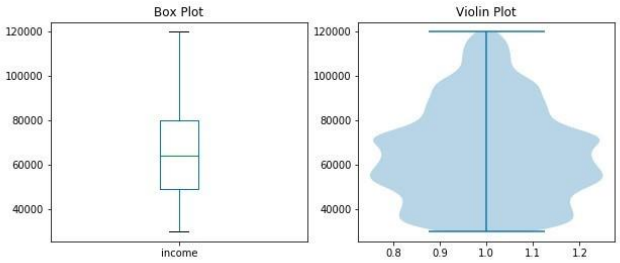


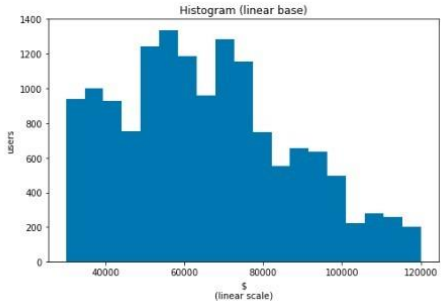
Analyzing the graphs above, we can see that although the age is normally distributed among the population, there is a local peak around the interval 20 ~ 25 for both male and female genders. Apparently, this deviation is not a problem to be handled beforehand. However, this is a point to be taken into consideration if networks have difficulty to converge. Hence, simply standardization for this feature seems to be good enough.

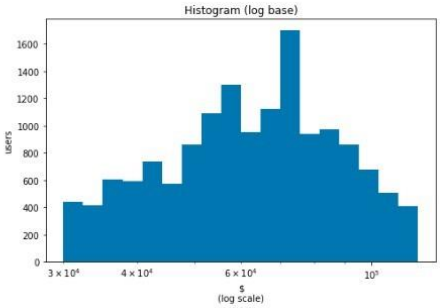
Explore income

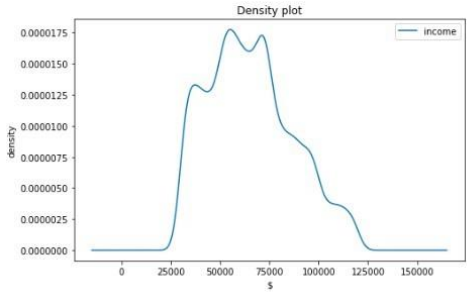
Income description

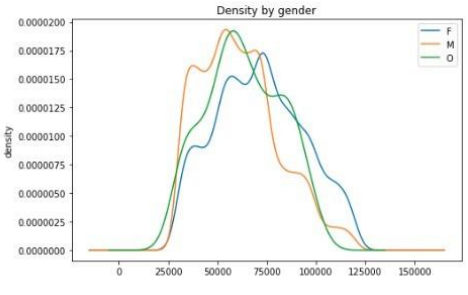
	income
count	14825.00
mean	65404.99
std	21598.30
min	30000.00
25%	49000.00
50%	64000.00
75%	80000.00
max	120000.00

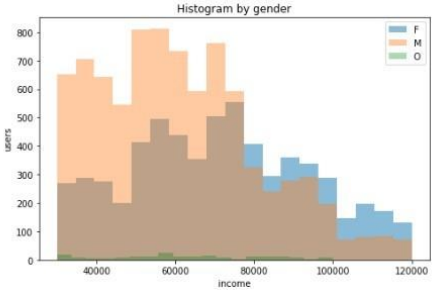


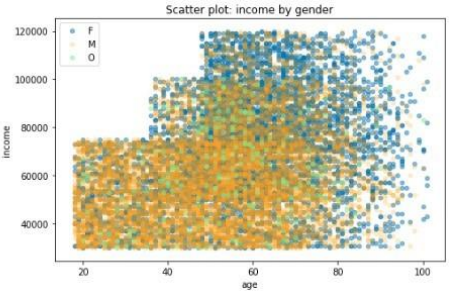












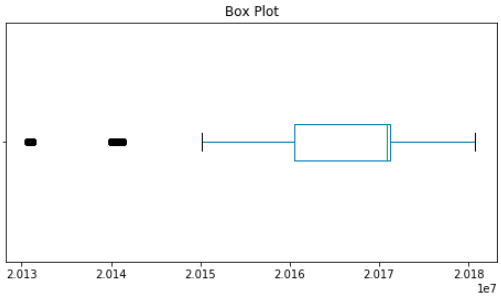
Explore membership

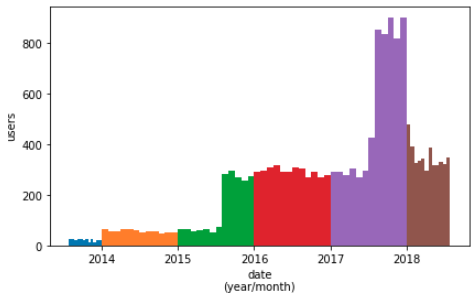
Membership description

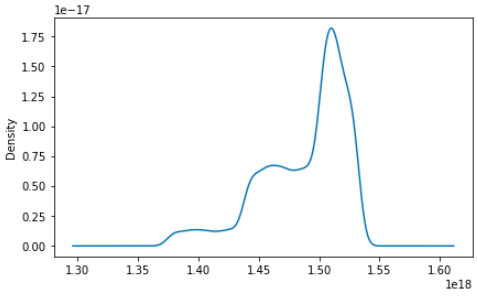
	became_member_on
count	17000.00
mean	20167034.23
std	11677.50
min	20130729.00
25%	20160526.00
50%	20170802.00
75%	20171230.00
max	20180726.00

Membership description

	became_member_on
count	17000
unique	1716
top	2017-12-07 00:00:00
freq	43
first	2013-07-29 00:00:00
last	2018-07-26 00:00:00







Here we see that the feature *became_member_on* presents a left-skewed distribution, and should be scaled to a normal distribution, in addition to having its values converted to a continuous representation (as type long).

iii. transcript.json

Event log containing records for transactions, offers received, offers viewed, and offers completed.

Size: 306648 events x 4 fields

- 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

	person	event	time	amount	offer_id	reward
0	78afa995795e4d85b5d9ceeca43f5fef	offer received	0	NaN	9b98b8c7a33c4b65b9aebfe6a799e6d9	NaN
1	a03223e636434f42ac4c3df47e8bac43	offer received	0	NaN	0b1e1539f2cc45b7b9fa7c272da2e1d7	NaN
2	e2127556f4f64592b11af22de27a7932	offer received	0	NaN	2906b810c7d4411798c6938adc9daaa5	NaN
3	8ec6ce2a7e7949b1bf142def7d0e0586	offer received	0	NaN	fafdcd668e3743c1bb461111dcafc2a4	NaN
4	68617ca6246f4fbc85e91a2a49552598	offer received	0	NaN	4d5c57ea9a6940dd891ad53e9dbe8da0	NaN
12650	389bc3fa690240e798340f5a15918d5c	offer viewed	0	NaN	f19421c1d4aa40978ebb69ca19b0e20d	NaN
12651	d1ede868e29245ea91818a903fec04c6	offer viewed	0	NaN	5a8bc65990b245e5a138643cd4eb9837	NaN
12652	102e9454054946fda62242d2e176fdce	offer viewed	0	NaN	4d5c57ea9a6940dd891ad53e9dbe8da0	NaN
12653	02c083884c7d45b39cc68e1314fec56c	offer viewed	0	NaN	ae264e3637204a6fb9bb56bc8210ddfd	NaN
12655	be8a5d1981a2458d90b255ddc7e0d174	offer viewed	0	NaN	5a8bc65990b245e5a138643cd4eb9837	NaN
12654	02c083884c7d45b39cc68e1314fec56c	transaction	0	0.83	NaN	NaN
12657	9fa9ae8f57894cc9a3b8a9bbe0fc1b2f	transaction	0	34.56	NaN	NaN
12659	54890f68699049c2a04d415abc25e717	transaction	0	13.23	NaN	NaN
12670	b2f1cd155b864803ad8334cdf13c4bd2	transaction	0	19.51	NaN	NaN
12671	fe97aa22dd3e48c8b143116a8403dd52	transaction	0	18.97	NaN	NaN
12658	9fa9ae8f57894cc9a3b8a9bbe0fc1b2f	offer completed	0	NaN	2906b810c7d4411798c6938adc9daaa5	2.0
12672	fe97aa22dd3e48c8b143116a8403dd52	offer completed	0	NaN	fafdcd668e3743c1bb461111dcafc2a4	2.0
12679	629fc02d56414d91bca360decdfa9288	offer completed	0	NaN	9b98b8c7a33c4b65b9aebfe6a799e6d9	5.0
12692	676506bad68e4161b9bbaffeb039626b	offer completed	0	NaN	ae264e3637204a6fb9bb56bc8210ddfd	10.0
12697	8f7dd3b2afe14c078eb4f6e6fe4ba97d	offer completed	0	NaN	4d5c57ea9a6940dd891ad53e9dbe8da0	10.0
0	78afa995795e4d85b5d9ceeca43f5fef	offer received	0	NaN	9b98b8c7a33c4b65b9aebfe6a799e6d9	NaN
15561	78afa995795e4d85b5d9ceeca43f5fef	offer viewed	6	NaN	9b98b8c7a33c4b65b9aebfe6a799e6d9	NaN
47582	78afa995795e4d85b5d9ceeca43f5fef	transaction	132	19.89	NaN	NaN
47583	78afa995795e4d85b5d9ceeca43f5fef	offer completed	132	NaN	9b98b8c7a33c4b65b9aebfe6a799e6d9	5.0
49502	78afa995795e4d85b5d9ceeca43f5fef	transaction	144	17.78	NaN	NaN

In the representative dataset above, we notice that:

- offer received and offer viewed have an associated offer id
- transaction has an amount value that indicates how much the customer spent
- offer completed is associated to a reward value
- every event is informed with person and time
- distinct events can occur to the same person at the same time (see registers 47582 and 47583)

So that, we need to check if these statements are always true.

Is there missing information?

	person	event	time	amount	offer_id	reward
offer received	True	True	True	False	True	False
offer viewed	True	True	True	False	True	False
transaction	True	True	True	True	False	False
offer completed	True	True	True	False	True	True

Is there event with extraneous information provided?

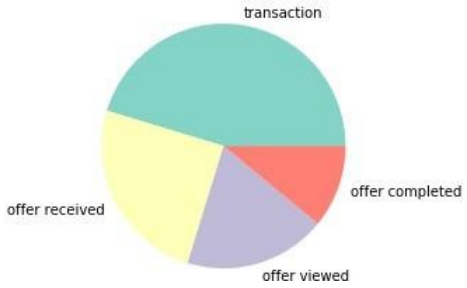
	person	event	time	amount	offer_id	reward
offer received	False	False	False	True	False	True
offer viewed	False	False	False	True	False	True
transaction	False	False	False	False	True	True
offer completed	False	False	False	True	False	False

According to these verifications, there is no missing nor extraneous information in this dataset.

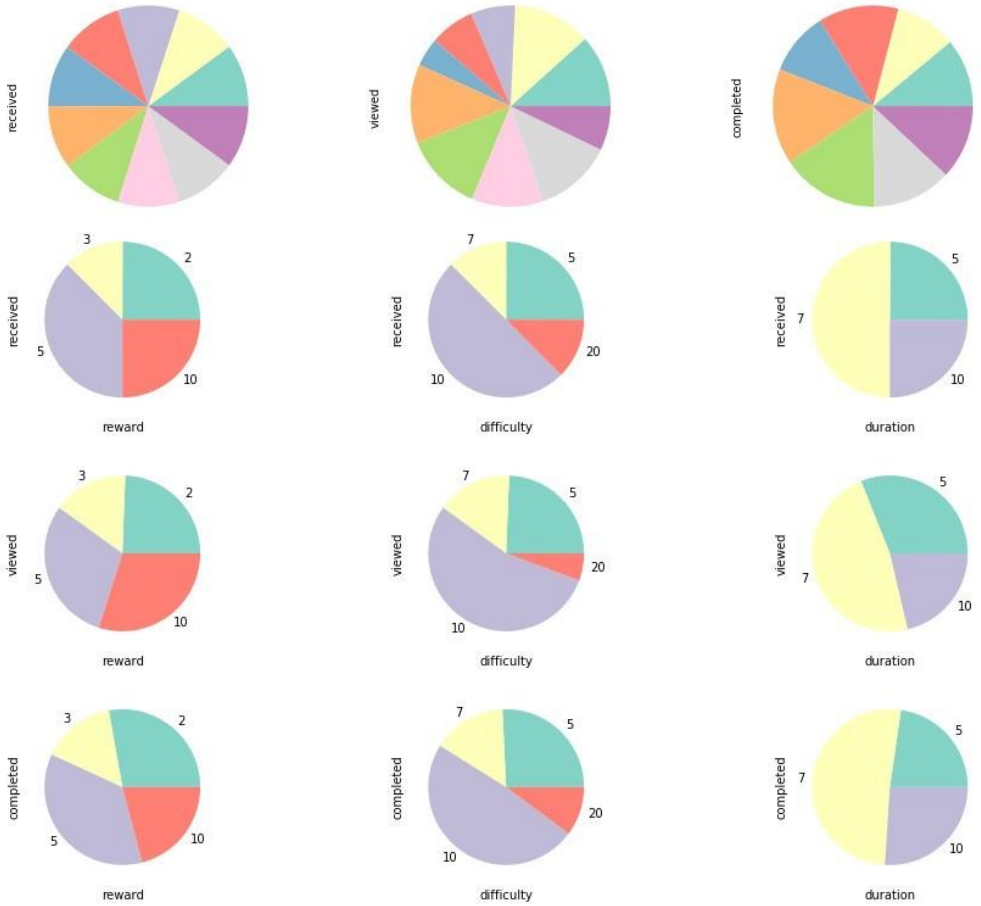
Now, we neeed to verify if there is some inconsistent values in it.

Explore events

	transaction	offer received	offer viewed	offer completed
event	138953	76277	57725	33579



received	viewed	completed
7658	6716	3688.0
7593	7298	3331.0
7617	4144	NaN
7677	4171	4354.0
7668	2663	3420.0
7646	7337	5156.0
7597	7327	5317.0
7618	6687	NaN
7571	7264	4296.0
7632	4118	4017.0



There is no anomaly detected in the data related to events.

Besides that, we note a correlation between views and channels. Not surprisingly, the more channels used to deliver the offer, the better. Additionally, using social media seems to be the most effective channel.

Furthermore, there is some correlation between offer completion and features reward, difficulty, and duration. However, it seems more related to the number of offers received.

Explore Transaction

Total

	amount
count	138953.00
mean	12.78
std	30.25
min	0.05
25%	2.78
50%	8.89
75%	18.07
max	1062.28

Count by person

	amount
count	17000.00
mean	8.17
std	5.12
min	0.00
25%	4.00
50%	7.00
75%	11.00
max	36.00

Sum by person

	amount
count	17000.00
mean	104.44
std	125.92
min	0.00
25%	21.82
50%	69.41
75%	148.78
max	1608.69

These description tables show us some interesting points.

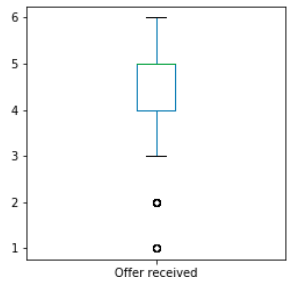
- There are customers who did not make one single purchase in the period
- There are anomalous values in the amount feature

No purchase

```
Customers who did not make a transaction in this period: 422
```

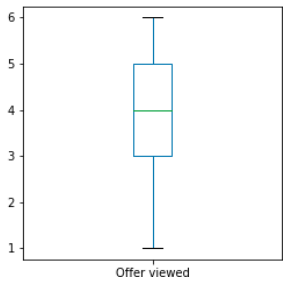
Offers received

	event
count	422.00
mean	4.46
std	1.12
min	1.00
25%	4.00
50%	5.00
75%	5.00
max	6.00



Offers viewed

	event
count	412.00
mean	3.62
std	1.42
min	1.00
25%	3.00
50%	4.00
75%	5.00
max	6.00



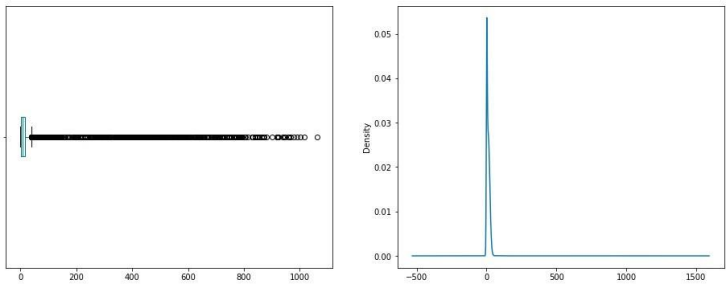
There are 422 customers who did not make transactions in this period, even though they received and viewed offers.

Admittedly, these customers may be analyzed in order to verify why they did not buy anything in the studied period. It is possible that they are opposed to marketing campaigns.

Perhaps, they should be filtered out of the dataset. Anyways, this is a point to be aware and analyze while training and evaluating the networks.

Anomalous values

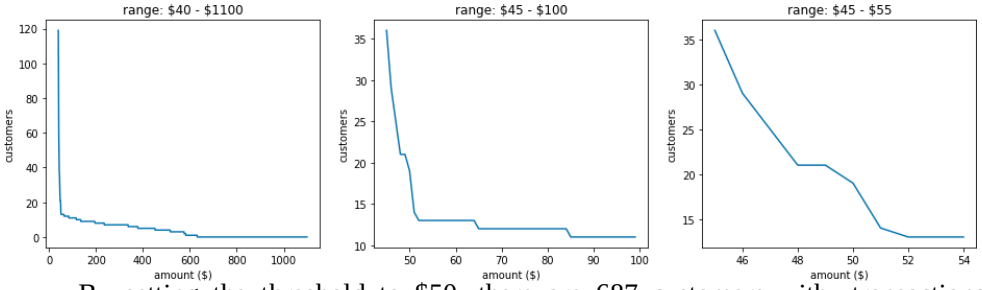
	count	mean	std	min	25%	50%	75%	max
amount	138953.0	12.78	30.25	0.05	2.78	8.89	18.07	1062.28



Anomalous values in transactions are so high that they deform completely the plots above.

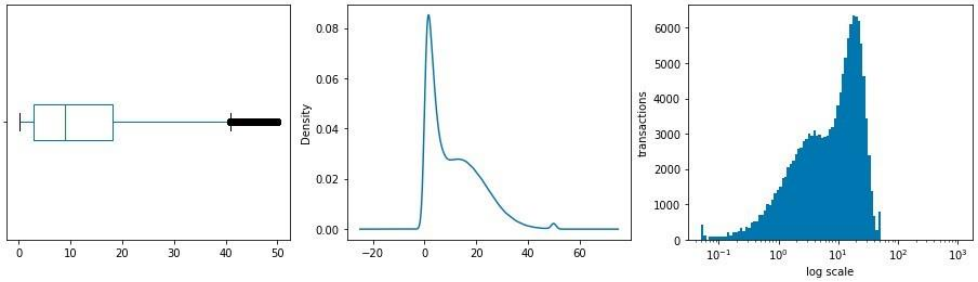
Altough it is quite clear that one purchase with a value of $1,062.28 in this dataset is an anomaly, it is especially tricky finding the value that correctly classifies outliers.

I tried to define this value by finding the number of times one customer repeats transactions with a similar amount. The three graphs below show the number of customers who repeat a transaction with a value higher than a certain amount. Around the value 50, that number tends to stabilize.



By setting the threshold to $50, there are 687 customers with transactions higher than this amount, and only 19 of them repeat a similar transaction.

The following graphs present transactions with value clipped to $50.



b. Algorithms and Techniques

i. Data Preprocessing

As we see in the section above, *Data Exploration and Exploratory Visualization*, it is necessary to handle some features before passing them into a neural network.

Considering the customer features *age* and *income*, their distribution is an approximately normal distribution, so these features will be standardized by using the Robust Scaler algorithm, from the scikit-learn library.

This algorithm scale features using statistics that are robust to outliers.

It removes the median and scales the data according to the quantile range.

Another case is the feature *became_member_on*, which presents a left- skewed distribution. It will be handled with the scikit-learn Quantile Transformer algorithm.

This method transforms the features to follow a uniform or a normal distribution using quantiles information. Therefore, for a given feature, this transformation tends to spread out the most frequent values. It also reduces the impact of outliers: this is therefore a robust preprocessing scheme.

The most problematic feature in the datasets is the amount spent by the customers in their purchases. However, this feature is not considered while defining whether an offer is appropriate.

In this project, the amount spent impacts only on the data-cleaning. Since there are abnormal values in this feature, those respective customers will be removed from the datasets.

ii. Neural Network

This project proposes implementing a Recurrent Neural Network to analyze the offers sent to customers along the time.

Once a customer received an offer and bought the related product, this offer might not be suitable anymore for this customer in the subsequent moment. Perhaps, he or she wants to try a different product, thus sending that offer twice in a row might not be as good as sending a different one. This temporal correlation is what suggests the use of a Recurrent Neural Network (RNN), which is specialized for the task of processing sequential data.

RNNs maintain a hidden state shared across several time steps, which would be interpreted as the internal customer state along the time.

The neural network models will be implemented using the PyTorch framework.

c. Benchmark

A more traditional model will be trained in the same dataset used by the intended Recurrent Neural Network, so that the results are comparable. In this case, it means training a Feedforward Neural Network (FNN).

Basically, an FNN analyzes a static input and makes predictions not considering the customer history. Differently, RNN is able to make predictions based on past events, instead of analyzing the current moment as an isolated situation.

A linear network model might understand an offer as adequate because it produced a good result in the past, and tends to repeat that action every time. However, it may not be able to detect whether the same offer becomes inadequate when sent a second time to the same customer, perhaps owing to the fact that the customer does not want to repeat the same purchase forever.

In another case, the customer is conditioned to buy products, so no offer sending is necessary anymore. However, that linear network keeps suggesting the same offer over again.

This relationship between past experiences and future behavior is what the Recurrent Neural Network is supposed to recognize.

Building and training both models allows us to compare the predictions made considering only the static user state (FNN) and those made based on the customer history (RNN). Then, we will be able to evaluate whether the problem stated is better addressed with a Recurrent Neural Network model.

III. Methodology

a. Data Preprocessing

In the section *Data Exploration and Exploratory Visualization*, some issues were identified in the datasets. In *Algorithms and Techniques*, it was explained how to tackle those problems. This section presents the results obtained for those features.

i. Portfolio Channels

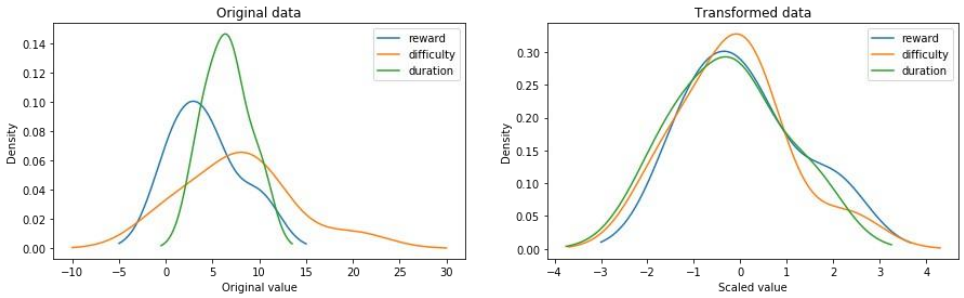
and Offer Type

One-hot encoding.

id	reward	difficulty	duration	bogo	discount	informational	email	mobile	social	web
ae264e3637204a6fb9bb56bc8210ddfd	10	10	7	1	0	0	1.0	1.0	1.0	0.0
4d5c57ea9a6940dd891ad53e9dbe8da0	10	10	5	1	0	0	1.0	1.0	1.0	1.0
3f207df678b143eea3cee63160fa8bed	0	0	4	0	0	1	1.0	1.0	0.0	1.0
9b98b8c7a33c4b65b9aebfe6a799e6d9	5	5	7	1	0	0	1.0	1.0	0.0	1.0
0b1e1539f2cc45b7b9fa7c272da2e1d7	5	20	10	0	1	0	1.0	0.0	0.0	1.0
2298d6c36e964ae4a3e7e9706d1fb8c2	3	7	7	0	1	0	1.0	1.0	1.0	1.0
fafdcd668e3743c1bb461111dcafc2a4	2	10	10	0	1	0	1.0	1.0	1.0	1.0
5a8bc65990b245e5a138643cd4eb9837	0	0	3	0	0	1	1.0	1.0	1.0	0.0
f19421c1d4aa40978ebb69ca19b0e20d	5	5	5	1	0	0	1.0	1.0	1.0	1.0
2906b810c7d4411798c6938adc9daaa5	2	10	7	0	1	0	1.0	1.0	0.0	1.0

Reward, Difficult, and Duration

Scaling.



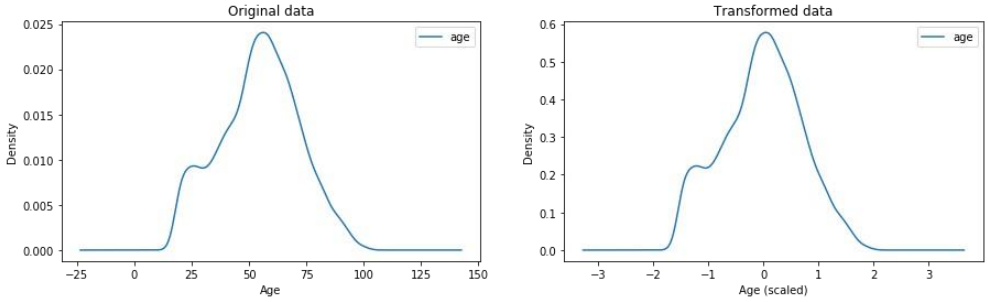
Resulting dataset

id	reward	difficulty	duration	bogo	discount	informational	mobile	social	web
ae264e3637204a6fb9bb56bc8210ddfd	1.57	0.35	-0.04	1	0	0	1.0	1.0	0.0
4d5c57ea9a6940dd891ad53e9dbe8da0	1.57	0.35	-0.34	1	0	0	1.0	1.0	1.0
3f207df678b143eea3cee63160fa8bed	-1.57	-1.68	-0.56	0	0	1	1.0	0.0	1.0
9b98b8c7a33c4b65b9aebfe6a799e6d9	0.13	-0.06	-0.04	1	0	0	1.0	0.0	1.0
0b1e1539f2cc45b7b9fa7c272da2e1d7	0.13	1.97	1.68	0	1	0	0.0	0.0	1.0
2298d6c36e964ae4a3e7e9706d1fb8c2	-0.04	0.10	-0.04	0	1	0	1.0	1.0	1.0
fafdcd668e3743c1bb461111dcafc2a4	-0.18	0.35	1.68	0	1	0	1.0	1.0	1.0
5a8bc65990b245e5a138643cd4eb9837	-1.57	-1.68	-1.95	0	0	1	1.0	1.0	0.0
f19421c1d4aa40978ebb69ca19b0e20d	0.13	-0.06	-0.34	1	0	0	1.0	1.0	1.0
2906b810c7d4411798c6938adc9daaa5	-0.18	0.35	-0.04	0	1	0	1.0	0.0	1.0

ii. Profile

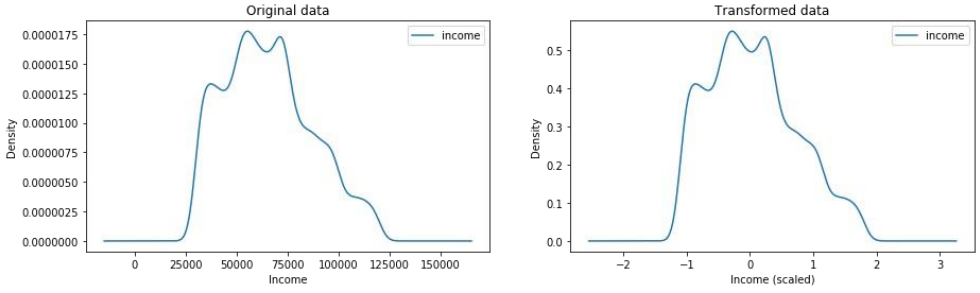
Age

Scaling.



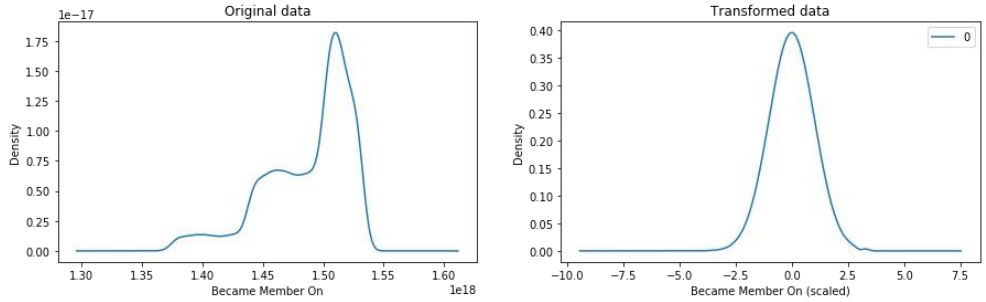
Income

Scaling.



Became Member On

Quantile transformation.



Resulting dataset

id	age	became_member_on	income	missing_data	F	M	O	U
0610b486422d4921ae7d2bf64640c50b	0.00	-0.04	1.63	0	1	0	0	0
78afa995795e4d85b5d9ceeca43f5fef	0.83	-0.13	1.23	0	1	0	0	0
e2127556f4f64592b11af22de27a7932	0.54	1.26	0.23	0	0	1	0	0
389bc3fa690240e798340f5a15918d5c	0.42	0.88	-0.33	0	0	1	0	0
2eeac8d8feae4a8cad5a6af0499a211d	0.12	0.44	-0.40	0	0	1	0	0
...	...	...	...	...	...	...	...	...
6d5f3a774f3d4714ab0c092238f3a1d7	-0.42	1.56	-0.30	0	1	0	0	0
2cb4f97358b841b9a9773a7aa05a9d77	0.25	2.14	0.30	0	0	1	0	0
01d26f638c274aa0b965d24cefe3183f	-0.25	-0.27	0.33	0	0	1	0	0
9dc1421481194dcd9400aec7c9ae6366	1.17	-0.82	-0.43	0	1	0	0	0
e4052622e5ba45a8b96b59aba68cf068	0.29	-0.03	0.63	0	1	0	0	0

iii. Transcript

Although the features in this dataset have been analyzed and some issues exposed, no transformation is necessary since this project is concerned in determine whether an offer would be completed regardless of the amount spent by the customer.

Apart from that, the data-cleaning process should remove from the dataset those inconsistent registers identified, as well as the customers who did not make a purchase in the analyzed period.

b. Implementation

I began this project by analyzing the data in the DataExploration notebook. I identified some issues in the dataset which I had to tackle before feeding the neural networks, as exposed in the section *Data Exploration and Exploratory Visualization*.

To discover the better algorithm to transform the data, besides studying feature engineering techniques, I tested some algorithms from the scikit-learn library (package sklearn.preprocessing) until choosing the ones that produced the more Gaussian-like distributions for the features, as demonstrated in section *Data Preprocessing*. These are the tested algorithms:

- MinMaxScaler
- Normalizer
- PowerTransformer
- QuantileTransformer
- RobustScaler
- StandardScaler

Having defined how to transform the data, I started to work in the Feature Engineering notebook. I transformed the data, filtered out some problematic customers, and create the features to feed the networks.

In this phase, the most complex piece of code is the function that classify the offers as appropriate or not. I created some auxiliary columns and tested some alternatives using Pandas to increase performance. However, this function requires many queries to determine if the offers received were completed in the subsequent period, and to define if they were viewed beforehand. In the end, this function resulted in a more complex and more time consumption algorithm that I would like it to be. Still, running on a CPU, this function takes around 20 minutes to complete, which it is acceptable since this part of the code only runs once.

Having created the features and labels, I built the data loaders and save them into a file, so that I would load them in a separate notebook to train the networks.

I created my networks flexible enough to be changed via parameters in the constructor call. That allowed me to adjust the size and depth of my models easily in order to test different settings until defining the best model to fit my data.

The first model I built and trained was the Linear Network. Since it has a simpler architecture, I could analyze if the features I had chosen were good enough for the networks converge. I first obtained an accuracy of around 63%,

so I concluded that I could move to the next step, despite being aware that this model should be improved.

After that, I defined and tested a vanilla Recurrent Neural Network and reached a quite similar accuracy than the benchmark model, the Linear Network.

Regarding the learning process, the loss was calculated by the Cross- Entropy algorithm available in the PyTorch framework. After the backpropagation of the loss, the Adam optimizer was employed to adjust the network weights.

As soon as I had all the pieces of my project working, it was time to enhance the performance to reach higher accuracy. The steps I took to do that are better explained in the next section, *Refinement*.

As defined in the section *Metrics*, I evaluated the models during the training phase by calculating their accuracy on a dataset distinct of that used in the learning process. In addition to that, I included other metrics relevant to this scenario: precision and recall. These metrics were calculated using the functions available in the scikit-learn library (package sklearn.metrics).

Since this project aims to define the offers most likely to be completed, precision is the metric more relevant. Precision indicates how many positive labels were correctly classified against those classified as positive but it is a negative case actually.

$$\texttt{accuracy}(y, \hat{y}) = \frac{1}{n_{\text{samples}}} \sum_{i=0}^{n_{\text{samples}}-1} 1(\hat{y}_i = y_i)$$

$$\text{precision} = \frac{tp}{tp + fp}$$

$$\text{recall} = \frac{tp}{tp + fn}$$

Where:

- $\hat{y}$ is the predicted value of the *i*-th sample and *y* is the corresponding true value.
- *tp* (True Positive) is the number of correctly classified sample when the category is *positive*.
- *fp* (False Positive) is the number of wrongly classified sample when the category is *negative*.
- *fn* (False Negative) is the number of wrongly classified sample when the category is *positive*.

The training dataset is not particularly well-balanced, containing 34.072 positive labels and 42.200 negative labels. To adjust these numbers and give more emphasis to precision than recall, I defined distinct weights to each class, providing this proportion to the Cross-Entropy Loss algorithm.

In the end, the resulting models have the following configuration:

- Linear Neural Network
  - Input: 16 features
  - 3 hidden layers fully-connected, 128 nodes each
  - Output: 2 logits values
  - Activation function: ReLU, after each hidden layer
  - Dropout: 20%, after each activation function
  - Optimizer: Adam
  - Learning rate: 0.0001
  - Weight decay: 0.0001

- Recurrent Neural Network
  - Input: 16 features
  - 2 recurrent layers, 128 nodes each
  - 1 fully-connected layer, 128 nodes
  - Output: 2 logits values
  - Activation function: ReLU, after each layer
  - Dropout: 20% after each activation function
  - Optimizer: Adam
  - Learning rate: 0.0001
  - Weight decay: 0.0001

The last part implemented was the System Demonstration notebook, which picks some customers from the dataset to illustrate how the system was conceived to work.

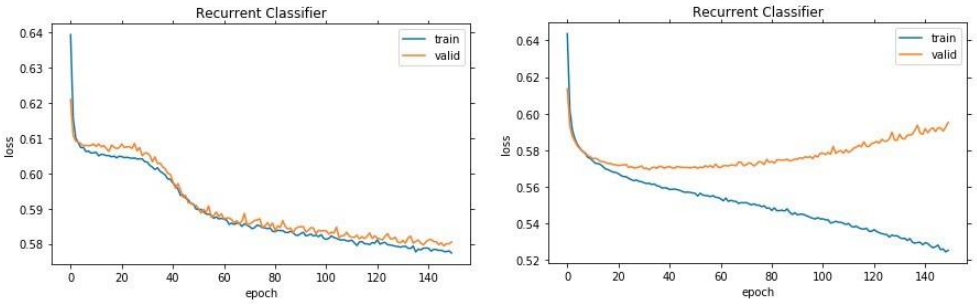
Although the implementation process described above seems to be very linear, it was necessary several iterations, indeed. As many parts of the code are interconnected, it is common to go back some steps to test a new hypothesis.

c. Refinement

First, I built a recurrent network using the RNN module provided by the PyTorch framework because this is the most straightforward model to build and train. Having a starter model, I could make changes to verify if the performance improved or not.

As the first alteration, I changed the activation function from TanH to ReLU, which enhanced the accuracy of around 1% but also led the model to overfit the training data.

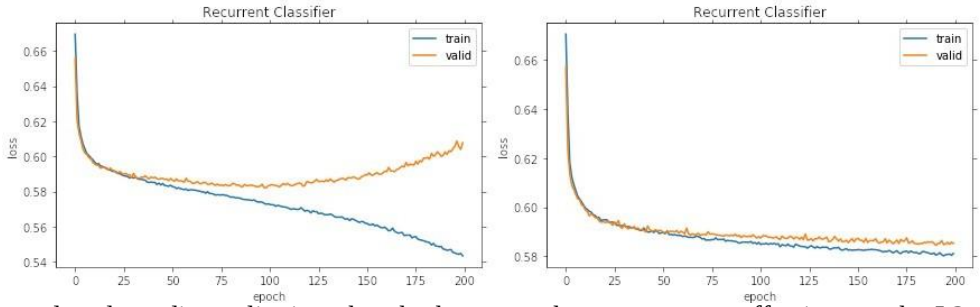
The figure below shows the training loss to the RNN with TanH activation on the left and the RNN with ReLU activation on the right.



After that, I replaced the vanilla RNN for the GRU (Gated Recurrent Unit) model, and then for the LSTM (Long Short-Term Memory). This plain alteration did not show much improvement, reaching the same level of accuracy for the three models.

However, GRU and LSTM are prone to overfit, and this showed up quite fast. In the following figure, the graph on the left shows GRU memorizing the training set very early, diverging when tested with the validation set. To overcome this problem, I applied L2-Regularization by setting the weight decay parameter present in Adam optimizer to 0.0001, generating the graph on the right.

To deal with the overfitting problem, I raised dropout to 50% as well as introduced gradient clipping, but both approach were not as effective as the L2-Regularization.



I tried initializing the hidden states as a full tensor of ones, as well as full of zeros. The best performance was obtained when initializing the hidden states to a random uniform distribution between [-0.003, 0.003].

When using Adagrad instead of Adam, the overfitting issue was accentuated. Raising the weight decay to control that problem prevented convergence. Thus, I opted to use Adam because it leads to more stable training. In addition to that, I adjusted the learning rate and weight decay to many different values, until fixing both parameters to 0.0001.

Furthermore, I tested many configurations of the size and depth of the networks. None of them reached an accuracy of 70%.

IV. Results

a. Model Evaluation and Validation

Having trained and compared many configurations as described in the sections *Implementation* and *Refinement*, I chose as the final model that one which reached a good balance between accuracy, precision, recall, and complexity (understood as size and depth).

The final model was evaluated by using unseen data. Thus this situation can be interpreted as new customers to the system.

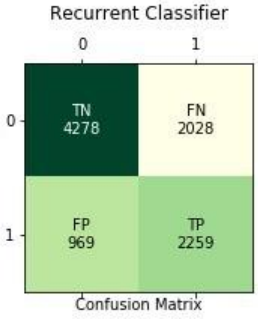
The results show that the model was able to find patterns and generalize information to a certain degree that enables it to have a similar performance over the three distinct datasets: training, validation, and test.

The final accuracy was 68.57%, which means that the model predicted correctly 6537 cases against 2997 errors.

The precision was 0.6998, which means that about 70% of the cases where the model predicted a higher probability of having an offer completed, they actually were.

The graphs below show the table with the metrics and the confusion matrix for the final recurrent model during the evaluation phase.

	Recurrent Classifier
accuracy (%)	68.57%
precision	0.6998
recall	0.5269
loss	0.5655



Having performed all the tests described in the *Refinement* section, I am convinced that the final model reached a good accuracy considering this dataset and the problem proposed. Also, this project determines the likelihood of a customer to complete an offer received, but sometimes the customer does not act in the most expected way. It might also be one reason why the networks did not reach higher accuracy.

b. Justification

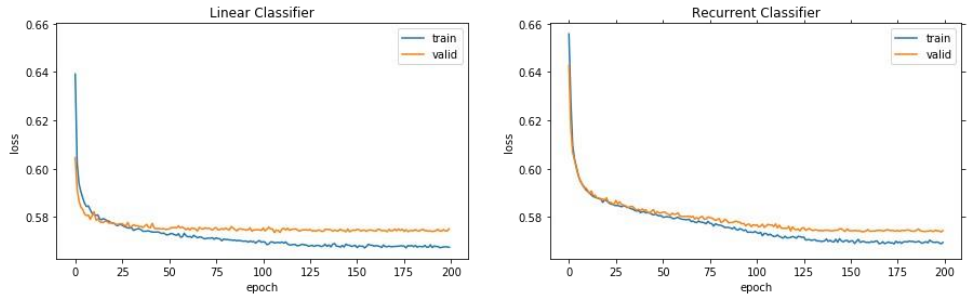
In this project, two distinct neural networks were trained to identify patterns in the dataset and predict the likelihood of one customer to complete one offer received.

The benchmark model is a Linear Feed-Forward Network which analyzes static data: one customer and one offer, regardless of past events.

The proposed model is a Recurrent Network that analyzes the same input data from the former network but takes into consideration the customer history. That means that the environment is dynamic, and the current prediction depends on past events. In this way, the same offer sent to the same customer has a distinct completion probability when considered in a distinct moment due to the change in the customer's internal state.

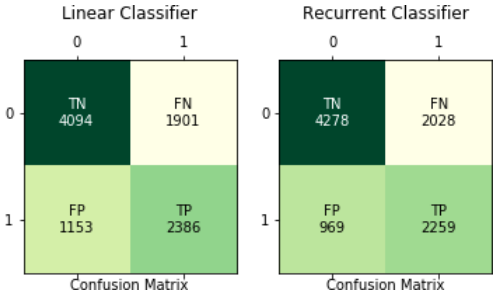
The results obtained in this project demonstrate that both models have similar performance when predicting the current offer for a specific customer. Hence, these results prove that the Recurrent Network understands the hidden patterns as well as the Linear Network. Moreover, these results support the correctness of the predictions made by the RNN for future situations.

The following graph shows the losses measured during the training phase. Both models reached a similar final loss with a slightly better result for the RNN.



The table below, on the left side, shows the accuracy, precision, and recall measured for the final trained models. On the right side, the confusion matrices. These numbers were taken from an unseen dataset, proving that both models could generalize the information seen in the training phase.

	Linear Classifier	Recurrent Classifier
accuracy (%)	67.97%	68.57%
precision	0.6742	0.6998
recall	0.5566	0.5269
loss	0.5646	0.5655



V. Conclusion

a. Free-Form Visualization

So now, imagine it is time for another round of offer sending.

How should the direct marketing system choose the most appropriate offer to send to each customer?

If you just looked at the past information, you would choose the offer which each customer completed the most. By doing that, you are not considering if the customer is saturated with this offer or even if there is another one more appealing for them.

One could group customers in clusters and calculate the average acceptance over that segment of the population. This situation might distort the concept of direct marketing since customers are not considered individually but generically.

A neural network generalizes information, as well. However, it makes calculations over each individual characteristic present in the input data. Hence, the system would target much more in each one of the customers than in groups of them.

Nonetheless, a linear network would suggest the same offer over again.

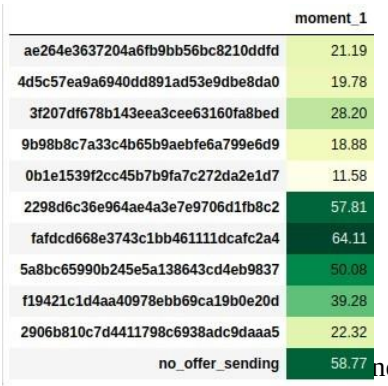
On the other hand, the recurrent model developed in this project considers the individual characteristics of each customer as well as their past experiences to suggest the most appropriate offer to the current time.

To illustrate the system in use, I picked one customer from the test dataset (unseen data) to demonstrate how the proposed model would have predicted the most appropriate offer at each moment.

The following table shows the offers sent to that customer and the respective result: if it was completed or not.

Moment	Offer ID	Completed
1	3f207df678b143eea3cee63160fa8bed	Yes
2	no_offer_sending	Yes
3	no_offer_sending	No
4	no_offer_sending	Yes
5	4d5c57ea9a6940dd891ad53e9dbe8da0	No
6	9b98b8c7a33c4b65b9aebfe6a799e6d9	No

The initial customer state is the situation where the customer has never received one offer. That means, there is no past event registered to this customer. Hence, you can realize that in this specific moment, both the linear and the recurrent models would make quite similar predictions.



	moment_1
ae264e3637204a6fb9bb56bc8210ddfd	21.19
4d5c57ea9a6940dd891ad53e9dbe8da0	19.78
3f207df678b143eea3cee63160fa8bed	28.20
9b98b8c7a33c4b65b9aebfe6a799e6d9	18.88
0b1e1539f2cc45b7b9fa7c272da2e1d7	11.58
2298d6c36e964ae4a3e7e9706d1fb8c2	57.81
fafdcd668e3743c1bb461111dcafc2a4	64.11
5a8bc65990b245e5a138643cd4eb9837	50.08
f19421c1d4aa40978ebb69ca19b0e20d	39.28
2906b810c7d4411798c6938adc9daaa5	22.32
no_offer_sending	58.77

The figure on the left shows the likelihood of each offer being completed by the customer. Based only on the personal demographic data, the system calculated in 64% the probability to the offer *fafdcd668e3743c1bb461111dcafc2a4*.

However, as shown in the table above, the current system sent the offer 3f207df678b143eea3cee63160fa8bed. And the customer actually completed it, despite the probability of only 28%.

nd moment, the customer already has one register in history. Consequently, the RNN can take it into consideration. The predictions for the second moment are displayed in the figure at the right. Notice that there is a probability of 66% that this customer makes a purchase even if he or she does not receive an offer.

The logs in the dataset show that this was what happened, indeed.

	moment_2
ae264e3637204a6fb9bb56bc8210ddfd	20.31
4d5c57ea9a6940dd891ad53e9dbe8da0	18.13
3f207df678b143eea3cee63160fa8bed	31.95
9b98b8c7a33c4b65b9aebfe6a799e6d9	20.04
0b1e1539f2cc45b7b9fa7c272da2e1d7	10.73
2298d6c36e964ae4a3e7e9706d1fb8c2	58.23
fafdcd668e3743c1bb461111dcafc2a4	62.81
5a8bc65990b245e5a138643cd4eb9837	55.24
f19421c1d4aa40978ebb69ca19b0e20d	40.39
2906b810c7d4411798c6938adc9daaa5	22.81
no_offer_sending	66.18

	moment_3
ae264e3637204a6fb9bb56bc8210ddfd	23.07
4d5c57ea9a6940dd891ad53e9dbe8da0	24.54
3f207df678b143eea3cee63160fa8bed	34.33
9b98b8c7a33c4b65b9aebfe6a799e6d9	19.80
0b1e1539f2cc45b7b9fa7c272da2e1d7	10.46
2298d6c36e964ae4a3e7e9706d1fb8c2	65.59
fafdcd668e3743c1bb461111dcafc2a4	70.18
5a8bc65990b245e5a138643cd4eb9837	58.99
f19421c1d4aa40978ebb69ca19b0e20d	46.60
2906b810c7d4411798c6938adc9daaa5	25.03
no_offer_sending	47.02

Then, for the next moment, the RNN predicted that it was necessary to send some offer so that it would incentivize the customer to make a transaction. The offer fafdcd668e3743c1bb461111dcafc2a4 had the highest probability of completion.

However, the current system did not send any offer, expecting the customer to buy something anyhow, which did not happen.

Notice that the RNN detected this situation. In the past moment, it was a good strategy not sending any offer but, at this moment, the probability of this costumer to make a purchase with no offer was only 47%.

And the analysis goes on in this same manner for the other moments. The figure below shows the predictions made for this customer at each time the original system sent an offer. Besides that, one last prediction was made at the moment 7, suggesting the next offer sending.

	moment_1	moment_2	moment_3	moment_4	moment_5	moment_6	moment_7
ae264e3637204a6fb9bb56bc8210ddfd	21.19	20.31	23.07	26.90	26.05	26.70	24.51
4d5c57ea9a6940dd891ad53e9dbe8da0	19.78	18.13	24.54	26.08	26.62	21.00	21.45
3f207df678b143eea3cee63160fa8bed	28.20	31.95	34.33	34.22	35.31	34.30	33.74
9b98b8c7a33c4b65b9aebfe6a799e6d9	18.88	20.04	19.80	22.78	21.50	21.73	19.26
0b1e1539f2cc45b7b9fa7c272da2e1d7	11.58	10.73	10.46	12.32	11.09	11.16	8.48
2298d6c36e964ae4a3e7e9706d1fb8c2	57.81	58.23	65.59	64.70	65.09	60.91	61.09
fafdcd668e3743c1bb461111dcafc2a4	64.11	62.81	70.18	68.56	68.51	63.59	63.52
5a8bc65990b245e5a138643cd4eb9837	50.08	55.24	58.99	58.74	60.61	63.74	63.68
f19421c1d4aa40978ebb69ca19b0e20d	39.28	40.39	46.60	48.14	48.43	45.16	45.48
2906b810c7d4411798c6938adc9daaa5	22.32	22.81	25.03	25.87	25.25	22.47	20.61
no_offer_sending	58.77	66.18	47.02	62.85	56.94	79.46	67.71

Having understood how the system makes predictions, we can now move forward to one next scenario.

Imagine that now it is time to decide which offer should be sent to each customer. Then, we should want to feed the RNN with the customers' history, so that we would have the predictions for the most appropriate offer for each one of them.

The figure below shows the predictions made for one batch of customers (picked from the unseen dataset), considering the scenario following the end of the period in which the data was collected. That means each customer has a history containing six offer sendings.

Each column represents one customer. Each row, one offer. Each cell, the probability of completion for that offer given that customer.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
ae264e3637204a6fb9bb56bc8210ddfd	37.11	40.32	41.18	32.69	2.47	41.95	37.59	23.12	24.51	46.21	36.26	35.90	3.45	17.06	47.38	43.60
4d5c57ea9a6940dd891ad53e9dbe8da0	33.69	48.60	33.41	31.28	2.21	46.96	43.69	20.52	21.45	41.23	41.77	34.63	2.39	15.67	59.31	56.06
3f207df678b143eea3cee63160fa8bed	42.43	23.50	45.66	31.48	18.93	24.76	20.26	24.07	33.74	40.27	20.99	42.44	20.83	18.32	32.18	38.73
9b98b8c7a33c4b65b9aebfe6a799e6d9	29.10	32.90	29.53	27.90	4.99	33.11	29.00	19.54	19.26	29.94	27.80	24.79	6.19	14.76	39.23	41.96
0b1e1539f2cc45b7b9fa7c272da2e1d7	13.38	20.99	14.96	16.61	1.43	21.56	17.47	11.53	8.48	14.96	16.97	10.71	1.66	8.72	22.48	24.11
2298d6c36e964ae4a3e7e9706d1fb8c2	72.80	62.52	74.49	55.39	14.77	57.64	50.03	39.38	61.09	74.81	51.56	73.88	18.70	31.16	74.09	80.05
fafdcd668e3743c1bb461111dcafc2a4	75.40	65.83	76.84	54.27	12.98	57.21	48.69	38.69	63.52	76.21	51.74	74.90	17.77	30.25	76.01	82.50
5a8bc65990b245e5a138643cd4eb9837	70.49	30.88	77.67	49.23	34.07	34.16	27.56	40.04	63.68	72.36	29.32	74.46	40.79	30.79	42.10	48.78
f19421c1d4aa40978ebb69ca19b0e20d	59.16	52.90	60.39	46.23	9.36	50.85	45.41	32.64	45.48	62.70	45.46	59.16	11.61	25.19	64.76	67.71
2906b810c7d4411798c6938adc9daaa5	30.01	29.78	29.92	26.34	5.34	28.00	23.16	18.73	20.61	28.98	24.24	26.63	5.97	14.27	37.23	43.45
no_offer_sending	74.66	49.26	86.03	76.47	42.61	67.40	58.87	56.57	67.71	82.57	48.00	74.10	60.71	48.94	48.58	50.52

From the figure above, we can extract highly pertinent pieces of information.

Notice that the offer *0b1e1539f2cc45b7b9fa7c272da2e1d7* has a faint probability of being completed by any of the customers in this batch. Maybe, this offer should be removed from the portfolio.

On the other hand, the offers 2298d6c36e964ae4a3e7e9706d1fb8c2, fafdcd668e3743c1bb461111dcafc2a4, 5a8bc65990b245e5a138643cd4eb9837, and f19421c1d4aa40978ebb69ca19b0e20d are widely accepted by customers in general. Thus, other offers could be created based on these ones.

The customers number *4* and number *12* have low probabilities of completing any of the offers. Perhaps, we should not send any offer to them, as this would represent some costs to the company with a very low expectation of return on investment (ROI).

Finally, we can notice that every customer in this batch has a fairly high probability of making a purchase even if they do not receive an offer. That may suggest that the previous campaign was successful enough to lead customers to buy Starbucks products, regardless of receiving an offer.

b. Reflection

Working in this project was a great pleasure for me. Besides being a quite appealing subject, I could put in practice a lot of knowledge I have acquired.

Since I have graduated in the Deep Learning and Deep Reinforcement Learning Nanodegree Programs at Udacity, I had already implemented many machine learning projects before starting this one.

However, contrary to those ones, this project was very open, with no rigid requirements nor fixed goals. Basically, I received a dataset to analyze, identify a problem, and propose and develop a suitable solution.

That makes this project truly real-life, quite similar to the problems we have to face when working in the software development industry.

Thus, I had to apply my background knowledge, acquired during my professional life, in order to propose and develop a solution that would fit in a tight schedule but would still be relevant.

This project was an invaluable experience on how it is planing, developing, and delivering a system like this one. Also, it was an outstanding opportunity to consolidate my knowledge in Machine Learning and Artificial Intelligence.

c. Improvement

Although I consider that this project has achieved a very relevant solution, I can visualize several other projects to improve the direct marketing system developed here.

As important as identifying the best offer to the customers is determining which ones generate a higher profit to the company. Since there is financial information in the datasets, maybe this should be the focus of one next project.

The system developed in this project looks to the past in order to suggest the best offer to send at the current moment. Nevertheless, this is a myopic approach because it does not consider long-term effects. Sometimes, an offer seems to be not appropriate for that moment, but it would generate more purchases in the future.

This scenario would be addressed by applying Reinforcement Learning, where the agent looks at offers that maximize the expected future reward (the company's profit).

I visualize the most extensive direct marketing system as being the association of the model developed in this project and both improvements mentioned above.

The final system would be able to look to the past to understand the user's internal state in the current moment, bootstrap the expected future profit for each offer, and then suggest the most appropriate one to the present date.

VI. References

[1] D. Silver, L. Newnham, D. Barker, S. Weller, and J. McFall,"Concurrent Reinforcement Learning from Customer Interactions," in *ICML*, 2013.

[2] G. Theocharous, P. S. Thomas, and M. Ghavamzadeh, "Personalized Ad Recommendation Systems for Life-Time Value Optimization with Guarantees," in *IJCAI*, 2015.

[3] X. Li *et al.*, "Recurrent Reinforcement Learning: A Hybrid Approach," *ArXiv*, vol. abs/1509.03044, 2015.

[4] X. Zhao, L. Zhang, Z. Ding, D. Yin, Y. Zhao, and J. Tang, "Deep Reinforcement Learning for List-wise Recommendations," *ArXiv*, vol. abs/1801.00209, 2017.

[5] J. Jin, C. Song, H. Li, K. Gai, J. Wang, and W. Zhang, "Real-Time Bidding with Multi-Agent Reinforcement Learning in Display Advertising," in *CIKM '18*, 2018, doi: 10.1145/3269206.3272021.

[6] G. Zheng *et al.*, "DRN: A Deep Reinforcement Learning Framework for News Recommendation," in *WWW*, 2018, doi: 10.1145/3178876.3185994.

[7] S.-Y. Chen, Y. Yu, Q. Da, J. Tan, H.-K. Huang, and H.-H. Tang, "Stabilizing Reinforcement Learning in Dynamic Environment with Application to Online Recommendation," in *KDD '18*, 2018, doi: 10.1145/3219819.3220122.

[8] F. Liu *et al.*, "Deep Reinforcement Learning based Recommendation with Explicit User-Item Interactions Modeling," *ArXiv*, vol. abs/1810.12027, 2018.

[9] J. Zhang, J. Yin, D. Lee, and L. Zhu, "Deep Reinforcement Learning for Personalized Search Story Recommendation," *ArXiv*, vol. abs/1907.11754, 2019.