Machine Learning Engineer Nanodegree
Udacity

Capstone Project
Report

Konstantinos Trikaliotis
February 2022

Table of Contents

I. Definition

a. Project Overview

This project stems from the marketing system used by Starbucks to stay in touch with its customers. Starbucks is America's largest coffee shop chain and a major organization that wants to attract new customers and motivate existing customers to consume more products by offering special offers and promotions through personalized targeted offers.

The main goal is to incentivize and reward customers who are registered on its platform by periodically sending individual messages / notifications through multiple channels such as email, social networks, web or directly through the Starbucks app. These messages / notifications contain offers related to its products and are divided into three main categories:

- BOGO: Buy-one-get-one is a type of voucher where users can get a reward equivalent to a threshold amount if they spend a certain amount of money.
- Discount: In the discount offer, the user earns a reward equal to a fraction of the amount spent.
- Information: In an informational offer, there is no actual reward, but also no mandatory amount the user is needed to spend.

Therefore, companies want to increase the profits generated by these marketing campaigns by implanting recommendation platforms / algorithms which will suggest what kind of vouchers should be offered to each individual customer, by looking at several factors which would lead to a successful offer. A simple definition of a successful offer is made up of two characteristics:

1. The customer has received, viewed, and completed the offer
2. The Customer’s spending will increase after or during the offer period.

An example of a failure would be a customer who would buy a product of $10 without seeing any offer and would receive a 20% discount on that amount due to an unseen offer received. This is not a successful offer at all, as the customer receives the offer without changing buying behavior or increasing spending, and as a result the company loses profits.

Considering all the details that must be observed by a company's marketing team, we can understand the difficulty of this matter. As a result, advanced machine learning recommendation algorithms are an essential tool for any marketing team, as they are able to extract historical insights from customers’ behavior and determine the

most relevant offer for each one of them, by just looking at some basic transaction and demographic data.

I personally consider this a really fascinating area, as providing better marketing campaigns cannot only benefit businesses by increasing their profits, but also the customers who would receive more relevant incentives based on their consumption behavior. Perhaps this win-win is the key to sustaining economic growth while people can afford better services and products.

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

More specifically, the data is a simulation of how people make purchasing decisions and how those decisions are driven by advertising offers. The program used to create the data simulates how people make purchasing decisions and how those decisions are influenced by advertising offers. Not all customers get the same offer, and this is the challenge I will try to solve with this data set.

Every purchase decision completed by each customer is influenced by some hidden characteristics. Consumers produce numerous events, including receiving offers, opening offers, and making purchases.

For simplicity, there are no explicit products to track, but only the amount of each transaction or offer is recorded.

The data is contained in three files:

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

These datasets are preprocessed (cleaned, transformed etc.) and combined in one final dataframe which contains all customers' demographic information, offers' features and the final label (whether an offer was successful for the corresponding customer). In the picture below we can see the first 5 rows of this final dataset:

	customer_age	M	customer_income	2013	2014	2015	2016	2017	2018	bogo	discount	informational	offer_duration_days	offer_difficulty	offer_reward	mobile	social	web	successful_offer
0	0.686747	0	0.777778	0	0	0	0	1	0	1	0	0	0.571429	0.25	0.5	1	0	1	1
1	0.602410	1	0.444444	0	0	0	0	0	1	1	0	0	0.571429	0.25	0.5	1	0	1	1
2	0.566265	1	0.255556	0	0	0	0	0	1	1	0	0	0.571429	0.25	0.5	1	0	1	1
3	0.457831	0	0.644444	0	0	0	0	0	1	1	0	0	0.571429	0.25	0.5	1	0	1	1
4	0.457831	0	0.644444	0	0	0	0	0	1	1	0	0	0.571429	0.25	0.5	1	0	1	0

I have found some interesting articles, blogs and repositories that have helped me a lot in making suggestions for this project. These are in the reference section below.

b. Project Statement

Any huge enterprise such as Starbucks, invests an excessive amount of cash in advertising campaigns expecting to increase its profitsthrough attracting new clients or by having a higher profit on the same existing customers than the assumed earlier. Therefore, it is important to identify the most relevant offers to the right customers for a successful campaign.

Nevertheless, a category of customers may not even see the offer provided to them. This is really likely to be a problem with the selected channel but may be related to other demographic characteristics of the customer (e.g. old customers do not want to try new products even if there are offers related to them).

Moreover, there is another category of customers who buy nothing, despite seeing an offer. This might be due to a problem with the offer sent, or it may not be a customer that should be considered as a target for this specific voucher type.

In other cases, customers may empathize with the offer, try new products, or spend more than usual and these are the situations that need to be identified, tracked and suggested by a model, platform or algorithm in general.

The problem this project is trying to solve is to identify / predict the most appropriate offer for each individual customer. That is, finding an offer that is likely to get your customers to buy more Starbucks products and increase the organization's profits.

In the context of this project the definitions of an appropriate and not appropriate offer are:

- An appropriate offer is that one where the consumer sees the offer provided and buys merchandise under its influence, finishing the offer lifecycle.
  Offer Received -> Offer Viewed -> Transaction occurred -> Offer Completed

- Not appropriate offer examples:
  - Customer does not see an offer
  - Customer sees the offer but does not complete it, as the offer did not force or convinced the customer to buy the product

- Customer purchases some goods and completes an offer and receives a reward before viewing that offer, as the consumer was not affected by this offer when he / she decided to make a transaction.

To address the above problem, this project suggests implementing a machine learning model able to analyze and extract useful information about the behavior of the customer, by analyzing the transcription data of their interaction with the Starbucks app.

The problem will be approached, as a binary classification problem, where the label is 1 if a customer is eligible for a specific offer and 0 if he / she is not. Moreover, the project will investigate if it is possible to identify and use the information of any money lost from some offers provided. There are specific demographic groups of customers, which have received a discount, without even opening the corresponding offer which means that the customer was not aware of that offer and his / her behavior did not change (the customer did not increase his / her spending).

The most common algorithms used for binary classification models are SVM (support vector machine), decision tree, random forests, ensemble methods and neural networks. This project will focus on creating an ensemble boosting tree algorithm and more specifically, an XGBoost model which is considered as a really powerful and easy to be trained model.

The final goal is to be able to create a marketing system that will help the marketing team at Starbucks to identify the most appropriate offer for each customer.

Below we can see the theoretical workflow followed to approach the solution stated above:

1. **Data Loading and Exploratory Data Analysis (EDA)**
   The data will be extracted and loaded in the correct format in a Jupyter notebook. Then, statistical data visualization will be performed to summarize the main characteristics of each feature / column and identify any statistical issues.

2. **Data Pre-processing**
   a. Data Cleaning: Look at the visualization created on the previous step and identify patterns and any potential issues. Find the best way to address any data issue such as null values on some columns or categorical features that need to be encoded.
   b. Data Transformation: Find the best way to combine all three tables into one summarized table which will be used as the main data source for features engineering and creating the final training and testing sets.

3. **Feature Engineering**
   Looking at the exploratory data analysis performed in the first step and determining which features are going to be used for predicting the outcome of our dependent variable. In addition, identifying features that can be engineered using the existing independent variables and performing feature reduction methods (PCA, etc .) if needed.

4. **Preparing and splitting the data into training, validation, and testing sets**
   After completing the feature engineering and having the final table which will be used to feed the models, the last step is to split the data into training, validation, and testing sets:
   - training set: This is the largest sample dataset and will be used to initially fit all models
   - validation set: This sample dataset will be used as an unbiased evaluation set of all models during the training process and will help to define the best hyperparameters for the final XGBoost algorithm
   - testing set: This sample dataset will be used for the final comparison of all the models and for the general evaluation of their performance on a real-world scenario. This will be an unseen dataset and will be used once the model is completely trained.

5. **Training and testing the performance of the benchmark model**
   The datasets created in step 4 will be used to train and test the performance of the benchmark model

6. **Constructing a powerful XGBoost model**
   a. Training and testing a random XGBoost classifier, by using some initial hyperparameters
   b. Hyperparameter tuning, using the training and validation sets during the training
   c. The final tuned and trained model will be used for evaluating the performance of the model in the testing set

7. **Conclusion**
    a. Comparing and evaluating the results of the final XGBoost and benchmark model by using the accuracy as a main metric and by looking at the confusion matrices for any significant differences
    b. Presenting final predictions and model usability in a real-world scenario

c. Metrics

As mentioned above, the metric that will be used to evaluate the performance of all models is the accuracy. Moreover, the confusion matrix will be plotted on each model, to identify any potential issues in the false positives or false negatives. By using the same evaluation metrics on all models, we make sure that the comparison between them is quantified and we are comparing “apples with apples”.

The accuracy is one of the most known and most used evaluation metrics and it has been used by many other people completing similar projects. This was one of the main reasons that has been selected as the main evaluation metric on this project too.

Definition of accuracy:

- Accuracy is one metric for evaluating classification models. Informally, accuracy is the fraction of predictions our model got right. Formally, accuracy has the following definition:

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}}$$

- For binary classification, accuracy can also be calculated in terms of positives and negatives as follows:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

II. Analysis

a. Data Exploration and Exploratory Visualization

Data exploration through data visualizations provides insights about the features' distribution, characteristics and potential inconsistencies. As a result, this section can be considered as one of the most useful and important parts of the machine learning workflow.

Data exploration helps us to identify any initial patterns between the independent and the dependent variables and at the same time helps us to identify relationships amongst the independent variables too (such as features which may be correlated with each other and needs to be excluded as they may have a bad influence on the model's performance).

In the next pages we can see various visuals around some characteristics and statistics (like the size, amount, min, max, mean, etc.) of the three main datasets used for this project. Moreover, we will see some exploratory visualizations performed on the final table which was used for training/validation and testing of the final models.

I. **portfolio** - containing offer ids and meta data about each offer (duration, type, etc.)
   - *id* (string/hash) - offer id
   - *offer_type* (string) - type of offer i.e. BOGO, discount, informational
   - *difficulty* (int) - money required to be spent to complete an offer
   - *offer_type* (string) - type of offer ie BOGO, discount, informational
   - *reward* (int) - reward given for completing an offer
   - *duration* (int) - time for offer to be open, in days
   - *channels* (list of strings)

	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

Above we can see the whole dataset which consists of 10 different offers. The types of offers are divided in three different categories which are:

- **BOGO**: Buy-one-get-one is a type of voucher where users can get a reward equivalent to a threshold amount if they spend a certain amount of money.
- **Discount**: In the discount offer, the user earns a reward equal to a fraction of the amount spent.
- **Information**: In an informational offer, there is no actual reward, but also no mandatory amount the user is needed to spend.

Moreover, as you can see from the picture above each offer has a different level of *difficulty* and *duration*. Also, there are three different ways to send an offer and these are through 4 available *channels* (web, email, mobile social).

Also, it is clear that feature *channels* can be considered as categorical feature and one hot encoding should be performed, in order to bring value. Same for offer_type which is one of our main columns.

Dataset Statistics

Dataset statistics

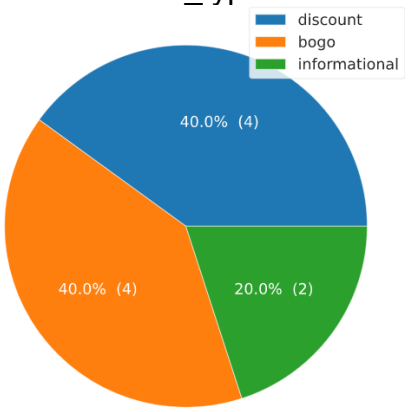
Number of variables	5
Number of observations	10
Missing cells	0
Missing cells (%)	0.0%
Duplicate rows	0
Duplicate rows (%)	0.0%
Total size in memory	1.8 KiB
Average record size in memory	181.2 B

Variable types

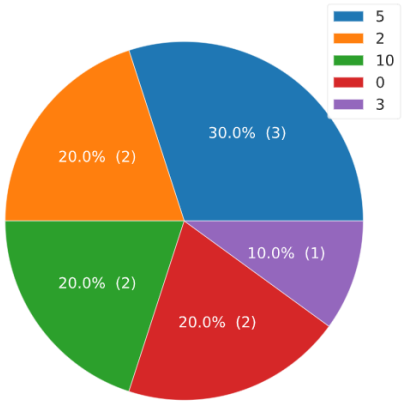
Categorical	4
Unsupported	1

- There are no missing values or duplicate rows
- All features seem to have a good data type without any problem

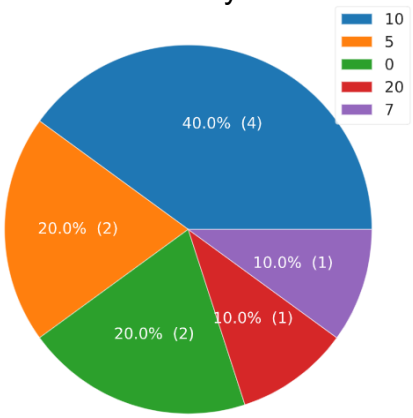
offer_type



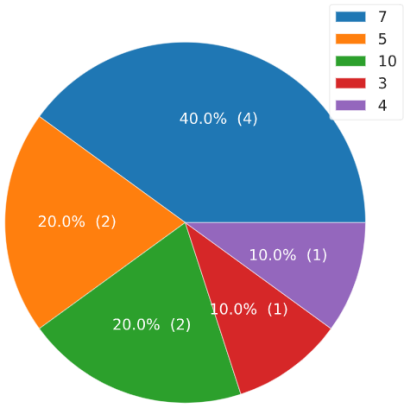
reward



difficulty



duration



- There are 4 discount, 4 bogo and 2 informational offer
- *Rewards*, *duration* and *difficulty* seem to have 5 distinct values each
- Moreover, *duration* and *difficulty* have same characteristics

---

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

	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
...	...	...	...	...	...
16995	F	45	6d5f3a774f3d4714ab0c092238f3a1d7	20180604	54000.0
16996	M	61	2cb4f97358b841b9a9773a7aa05a9d77	20180713	72000.0
16997	M	49	01d26f638c274aa0b965d24cefe3183f	20170126	73000.0
16998	F	83	9dc1421481194dcd9400aec7c9ae6366	20160307	50000.0
16999	F	62	e4052622e5ba45a8b96b59aba68cf068	20170722	82000.0

17000 rows × 5 columns

Profile dataset has 17000 rows (should be distinct customers) and 5 features. *Gender* is considered as a categorical feature and should be mapped to distinct columns using one hot encoding.

Also, *became_member_on* should be engineered and changed to a date format in order to make more sense.

Moreover, it is clear that there are some missing values in the dataset which should be identified and handled (to be excluded or replaced).

Dataset Statistics

	gender	age	id	became_member_on	income
count	14825	17000.000000	17000	17000	14825.000000
unique	3	NaN	17000	1716	NaN
top	M	NaN	dd9594e6ca65431db74456ae27a298e4	2017-12-07 00:00:00	NaN
freq	8484	NaN	1	43	NaN
first	NaN	NaN	NaN	2013-07-29 00:00:00	NaN
last	NaN	NaN	NaN	2018-07-26 00:00:00	NaN
mean	NaN	62.531412	NaN	NaN	65404.991568
std	NaN	26.738580	NaN	NaN	21598.299410
min	NaN	18.000000	NaN	NaN	30000.000000
25%	NaN	45.000000	NaN	NaN	49000.000000
50%	NaN	58.000000	NaN	NaN	64000.000000
75%	NaN	73.000000	NaN	NaN	80000.000000
max	NaN	118.000000	NaN	NaN	120000.000000

Dataset statistics

Number of variables	4
Number of observations	17000
Missing cells	4350
Missing cells (%)	6.4%
Duplicate rows	12
Duplicate rows (%)	0.1%
Total size in memory	1.4 MiB
Average record size in memory	84.6 B

Variable types

Categorical	1
Numeric	2
DateTime	1

Warnings

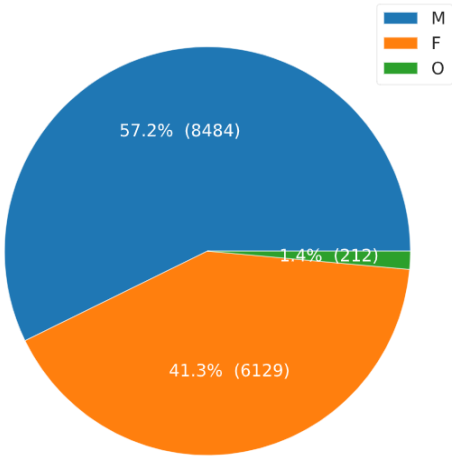
Dataset has 12 (0.1%) duplicate rows	Duplicates
`gender` has 2175 (12.8%) missing values	Missing
`income` has 2175 (12.8%) missing values	Missing

- There are no extreme values for any of the columns except *age* where the value 118 seems to be presented when there is no data for that customer
- Moreover, there are 4350 missing cells which come from 2175 for both *gender* and *income* columns. This number is 12.8% of the total population. Given the fact that the missing values are below 15%, they will be excluded from the final analysis later on in order to not have a bad impact on the models

- Also, there are 12 duplicate rows which should be identified and dropped from the dataset. These duplicates must be generated due to some data processing issues.

Gender

Value	Count	Frequency (%)
M	8484	49.9%
F	6129	36.1%
O	212	1.2%
(Missing)	2175	12.8%



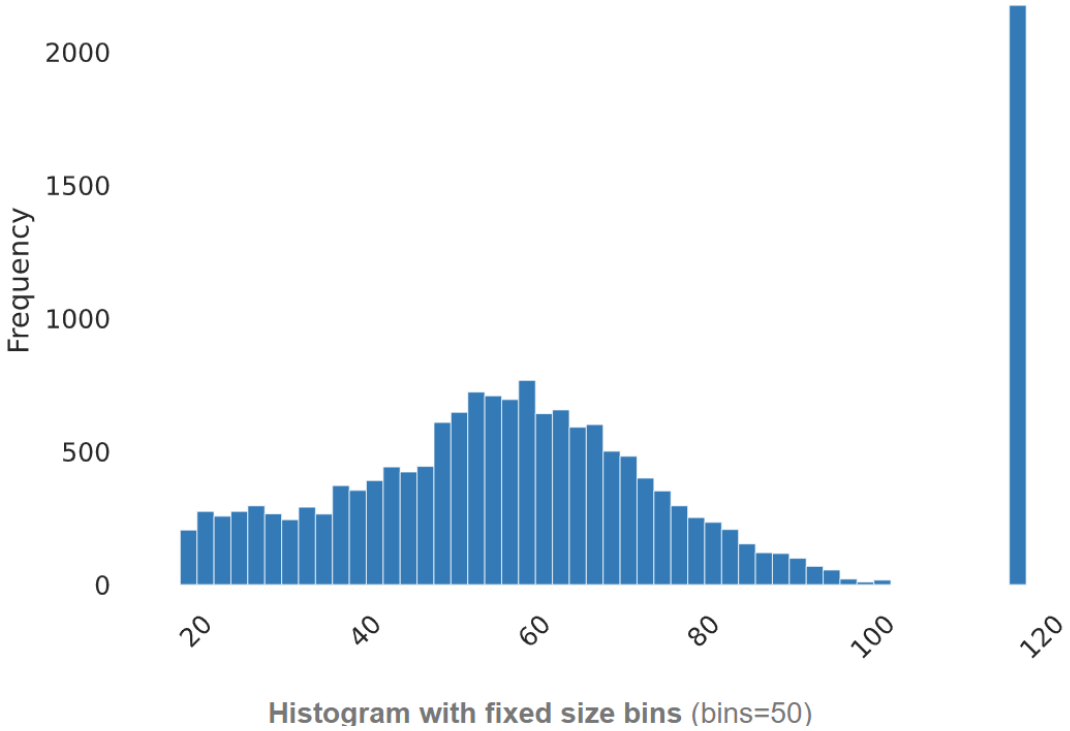
- 12.8% missing values
- If we do not consider these missing values:
    - Majority of customers are Men with 57.2% of the total population
    - Females cover the 41.3% of the population
    - There are 212 customers (1.4%) with gender as Other

Age

118	2175	12.8%
58	408	2.4%
53	372	2.2%
51	363	2.1%
54	359	2.1%
59	359	2.1%
57	353	2.1%
52	351	2.1%
55	350	2.1%
56	342	2.0%

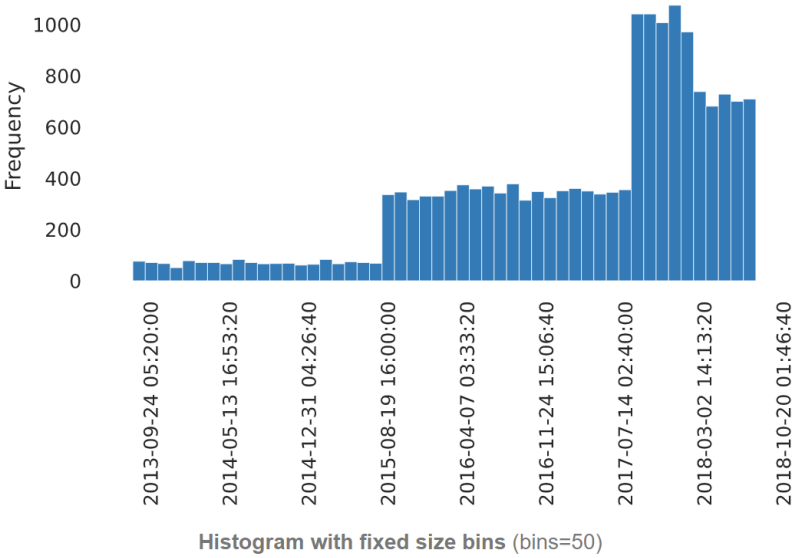
Quantile statistics

Minimum	18
5-th percentile	24
Q1	45
median	58
Q3	73
95-th percentile	118
Maximum	118
Range	100
Interquartile range (IQR)	28



- Age seems to have an extreme value of 118 years for all missing values (12.8%)
- Majority of customers seem to be around 50 to 60 years old, with 58 years to be the most frequent age
- There are no children in our dataset (minimum age is 18 years).
- Age seems to be equal distributed between 18 to 100 years (excluding the 118 years)

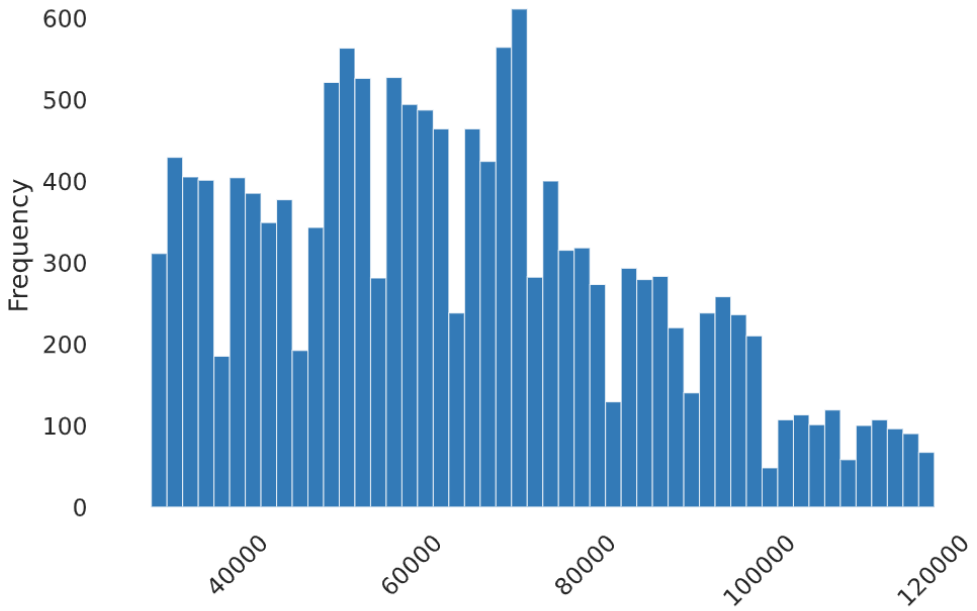
Became_member_on



- It seems that the volume of customers change per year and most customers became members between 2017 and 2018
- Looking at the distribution above, we can see that the biggest value in this column is in the year and this can be extracted and used as a categorical feature (one hot encoding to 2013, 2014, 2015, 2016, 2017 and 2018)

Income

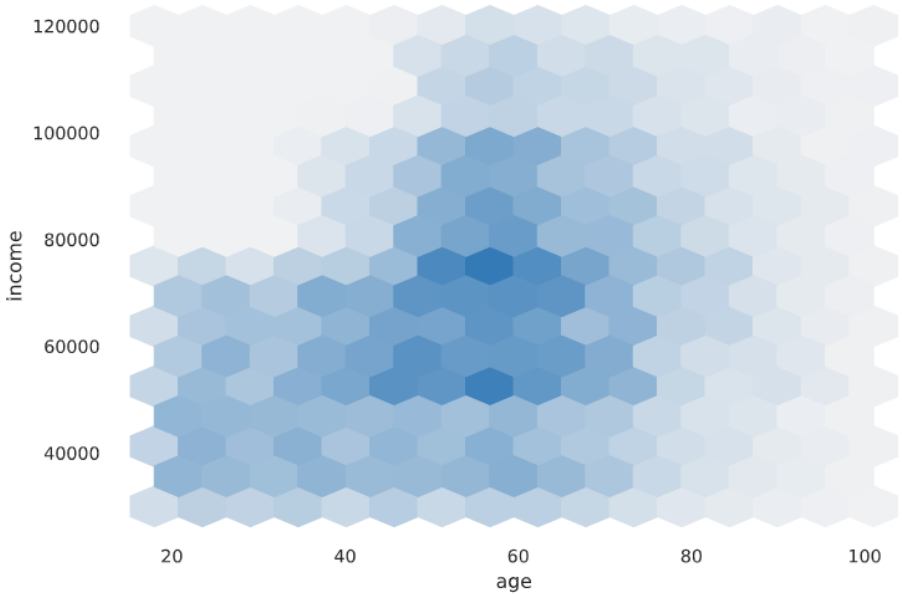
Distinct	91	**Minimum**	30000
Distinct (%)	0.6%	**Maximum**	120000
Missing	2175	**Zeros**	0
Missing (%)	12.8%	**Zeros (%)**	0.0%
Infinite	0	**Negative**	0
Infinite (%)	0.0%	**Negative (%)**	0.0%
Mean	65404.99157	**Memory size**	132.9 KiB



Histogram with fixed size bins (bins=50)

- 12.5% are missing values
- Mean income seems to be around 65k
- Income distribution seems to be left skewed with majority of people earn no more than 80k

Interactions



- Income above 80k seems to be distributed in people above 35 years old and older people have a bigger income in general

---

III. **transcript** - records for transactions, offers received, offers viewed, and offers completed
    - *event* (str) - record description (i.e. transaction, offer received, offer *viewed*, etc.)
    - *person* (str) - customer id
    - *time* (int) - time in hours since the start of the test. The data begins at time t=0
    - *value* - (dict of strings) - either an offer id or transaction amount depending on the record

	person	event	value	time
0	78afa995795e4d85b5d9ceeca43f5fef	offer received	{'offer id': '9b98b8c7a33c4b65b9aebfe6a799e6d9'}	0
1	a03223e636434f42ac4c3df47e8bac43	offer received	{'offer id': '0b1e1539f2cc45b7b9fa7c272da2e1d7'}	0
2	e2127556f4f64592b11af22de27a7932	offer received	{'offer id': '2906b810c7d4411798c6938adc9daaa5'}	0
3	8ec6ce2a7e7949b1bf142def7d0e0586	offer received	{'offer id': 'fafdcd668e3743c1bb461111dcafc2a4'}	0
4	68617ca6246f4fbc85e91a2a49552598	offer received	{'offer id': '4d5c57ea9a6940dd891ad53e9dbe8da0'}	0
...	...	...	...	...
306529	b3a1272bc9904337b331bf348c3e8c17	transaction	{'amount': 1.5899999999999999}	714
306530	68213b08d99a4ae1b0dcb72aebd9aa35	transaction	{'amount': 9.53}	714
306531	a00058cf10334a308c68e7631c529907	transaction	{'amount': 3.61}	714
306532	76ddbd6576844afe811f1a3c0fbb5bec	transaction	{'amount': 3.5300000000000002}	714
306533	c02b10e8752c4d8e9b73f918558531f7	transaction	{'amount': 4.05}	714

306534 rows × 4 columns

Transcript dataset has 306534 rows (should be events per customer, time and value) and 4 features.

Value feature will be engineered for this exploratory data analysis steps and instead of have an offer id as a value for the offer_id key, it would have the corresponding offer type (bogo, discount, informational)

	person	event	value	time	value_eda
0	78afa995795e4d85b5d9ceeca43f5fef	offer received	{'offer id': '9b98b8c7a33c4b65b9aebfe6a799e6d9'}	0	{"offer id": "bogo"}
1	a03223e636434f42ac4c3df47e8bac43	offer received	{'offer id': '0b1e1539f2cc45b7b9fa7c272da2e1d7'}	0	{"offer id": "discount"}
2	e2127556f4f64592b11af22de27a7932	offer received	{'offer id': '2906b810c7d4411798c6938adc9daaa5'}	0	{"offer id": "discount"}
3	8ec6ce2a7e7949b1bf142def7d0e0586	offer received	{'offer id': 'fafdcd668e3743c1bb461111dcafc2a4'}	0	{"offer id": "discount"}
4	68617ca6246f4fbc85e91a2a49552598	offer received	{'offer id': '4d5c57ea9a6940dd891ad53e9dbe8da0'}	0	{"offer id": "bogo"}

Dataset Statistics

	person	event	value	time	value_eda
count	306534	306534	306534	306534.000000	306534
unique	17000	4	5121	NaN	5114
top	94de646f7b6041228ca7dec82adb97d2	transaction	{'offer id': '2298d6c36e964ae4a3e7e9706d1fb8c2'}	NaN	{"offer id": "bogo"}
freq	51	138953	14983	NaN	55948
mean	NaN	NaN	NaN	366.382940	NaN
std	NaN	NaN	NaN	200.326314	NaN
min	NaN	NaN	NaN	0.000000	NaN
25%	NaN	NaN	NaN	186.000000	NaN
50%	NaN	NaN	NaN	408.000000	NaN
75%	NaN	NaN	NaN	528.000000	NaN
max	NaN	NaN	NaN	714.000000	NaN

Dataset statistics

Number of variables	4
Number of observations	306534
Missing cells	0
Missing cells (%)	0.0%
Duplicate rows	396
Duplicate rows (%)	0.1%
Total size in memory	72.5 MiB
Average record size in memory	247.9 B

Warnings

Dataset has 396 (0.1%) duplicate rows	Duplicates
`person` has a high cardinality: 17000 distinct values	High cardinality
`value_eda` has a high cardinality: 5114 distinct values	High cardinality
`time` has 15561 (5.1%) zeros	Zeros

- There are 396 duplicate rows (0.1%)
- There are 17000 distinct customers which do not match the number of distinct customers seen in the profile dataset. As a result we will need to remove any row which includes a customer which does not appear in the profile dataset

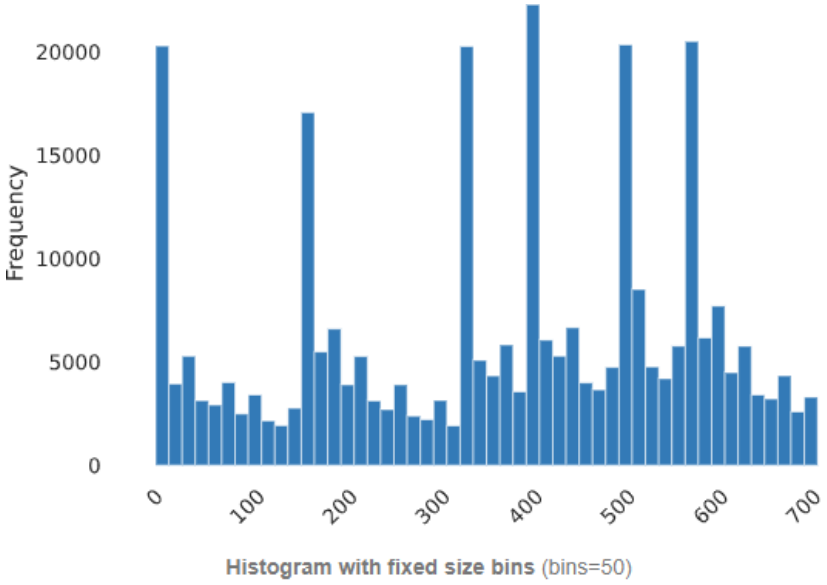
Event

Common Values

Value	Count	Frequency (%)
transaction	138953	45.3%
offer received	76277	24.9%
offer viewed	57725	18.8%
offer completed	33579	11.0%

- Most events seem to be related to transactions (45.3%)
- There are 76277 offers received and just 33579 which have been completed. This is really interesting as shows the fact that many people receive an offer but do not complete it

Time



Value	Count	Frequency (%
408	17030	5.6%
576	17015	5.6%
504	16822	5.5%
336	16302	5.3%
168	16150	5.3%
0	15561	5.1%
414	3583	1.2%
510	3514	1.1%
582	3484	1.1%
588	3222	1.1%
Other value	193851	63.2%

- As it can be seen above, time is for 700 hours (about a month)
- There are some peak periods for events which are the
    - 0 time (start of the dataset)
    - 168 hours later (7 days/1 week)
    - 336 hours later (14 days/2 weeks)
    - 408 hours later (17 days)
    - 504 hours later (21 days/3 weeks)
    - 576 hours later (24 days/1 week after 408 hours)

Value_eda

Common Values

Value	Count	Frequency (%)
{"offer id": "bogo"}	55948	18.3%
{"offer id": "disc…	51988	17.0%
{"offer id": "infor…	26066	8.5%
{"offer_id": "fafd…	5317	1.7%
{"offer_id": "229…	5156	1.7%
{"offer_id": "9b9…	4354	1.4%
{"offer_id": "f19…	4296	1.4%
{"offer_id": "290…	4017	1.3%
{"offer_id": "ae2…	3688	1.2%
{"offer_id": "0b1…	3420	1.1%

- Bogo is the most common offer appear (18.3%)
- Moreover, we can see that offer id can appear as offer_id too and should be handled

---

IV. **merged_df** - merged dataset which has been created after preprocessing (cleaning, transforming etc) and combining the three main tables which has been described above. This dataset used to construct the final dataset used to feed the machine learning models

	customer_age	customer_gender	customer_income	customer_registration_year	offer_type	offer_duration_days	offer_difficulty	offer_reward	email	mobile	social	web	successful_offer
0	75	F	100000.0	2017	bogo	7	5	5	1	1	0	1	1
1	68	M	70000.0	2018	bogo	7	5	5	1	1	0	1	1
2	65	M	53000.0	2018	bogo	7	5	5	1	1	0	1	1
3	65	M	53000.0	2018	bogo	7	5	5	1	1	0	1	1
4	56	F	88000.0	2018	bogo	7	5	5	1	1	0	1	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...
66496	48	M	58000.0	2018	bogo	5	10	10	1	1	1	1	1
66497	44	F	81000.0	2016	bogo	5	10	10	1	1	1	1	1
66498	47	M	94000.0	2017	bogo	5	10	10	1	1	1	1	0
66499	61	F	60000.0	2014	bogo	5	10	10	1	1	1	1	1
66500	58	F	78000.0	2016	bogo	5	10	10	1	1	1	1	1

66501 rows × 13 columns

Merged dataset has 66501 rows and 13 columns (12 features and 1 dependent variable which is the label)

Dataset Statistics

	customer_age	customer_gender	customer_income	customer_registration_year	offer_type	offer_duration_days	offer_difficulty	offer_reward	email	mobile	social	web	successful_offer
count	66501.000000	66501	66501.000000	66501.000000	66501	66501.000000	66501.00000	66501.000000	66501.0	66501.000000	66501.000000	66501.000000	66501.000000
unique	NaN	3	NaN	NaN	3	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
top	NaN	M	NaN	NaN	discount	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
freq	NaN	38129	NaN	NaN	26654	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
mean	54.369258	NaN	65371.618472	2016.622021	NaN	6.507571	7.71417	4.198824	1.0	0.898859	0.598517	0.799612	0.567119
std	17.395430	NaN	21623.288473	1.198364	NaN	2.204416	5.54754	3.398100	0.0	0.301518	0.490202	0.400294	0.495478
min	18.000000	NaN	30000.000000	2013.000000	NaN	3.000000	0.00000	0.000000	1.0	0.000000	0.000000	0.000000	0.000000
25%	42.000000	NaN	49000.000000	2016.000000	NaN	5.000000	5.00000	2.000000	1.0	1.000000	0.000000	1.000000	0.000000
50%	55.000000	NaN	64000.000000	2017.000000	NaN	7.000000	10.00000	5.000000	1.0	1.000000	1.000000	1.000000	1.000000
75%	66.000000	NaN	80000.000000	2017.000000	NaN	7.000000	10.00000	5.000000	1.0	1.000000	1.000000	1.000000	1.000000
max	101.000000	NaN	120000.000000	2018.000000	NaN	10.000000	20.00000	10.000000	1.0	1.000000	1.000000	1.000000	1.000000

Dataset statistics

Number of variables	13
Number of observations	66501
Missing cells	0
Missing cells (%)	0.0%
Duplicate rows	9431
Duplicate rows (%)	14.2%
Total size in memory	16.9 MiB
Average record size in memory	265.8 B

Variable types

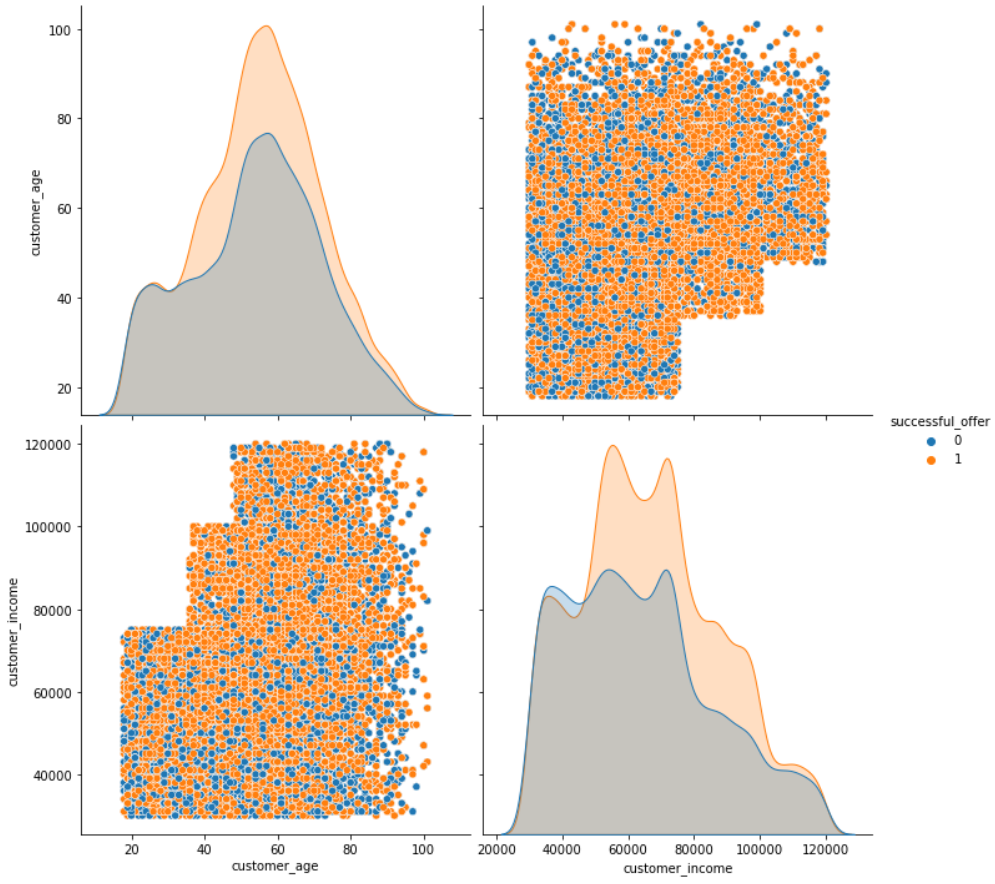
Numeric	3
Categorical	10

Warnings

`email` has constant value "1"	Constant
Dataset has 9431 (14.2%) duplicate rows	Duplicates

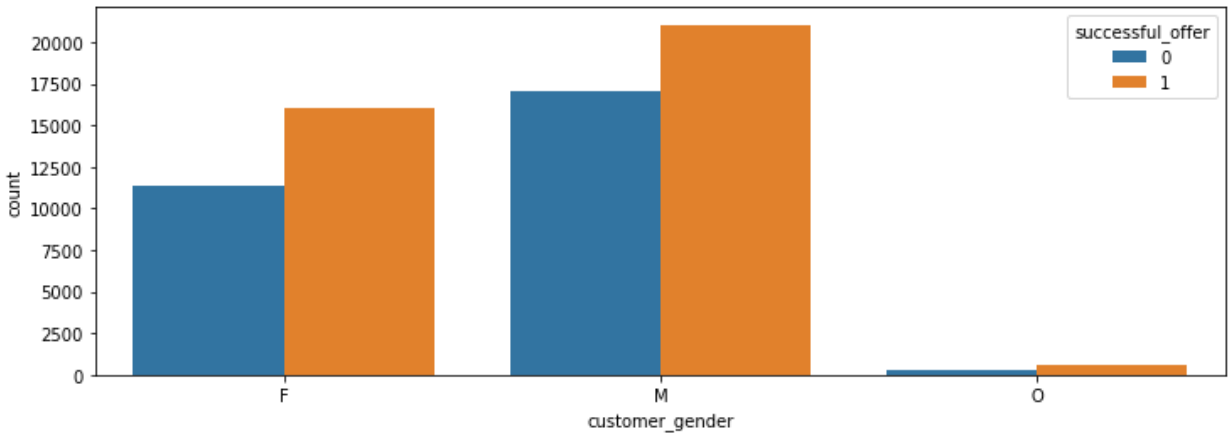
- No extreme values have been identified from the statistics above as the datasets have been cleaned before combined
- There are some duplicate rows which should be dropped (14.2%)
- Moreover, we can see that categorical features should be encoded and numerical features should be scaled
- No missing values
- Also, there are 12 duplicate rows which should be identified and dropped from the dataset. These duplicates must be generated due to some data processing issues.

Customer Age & Customer Income



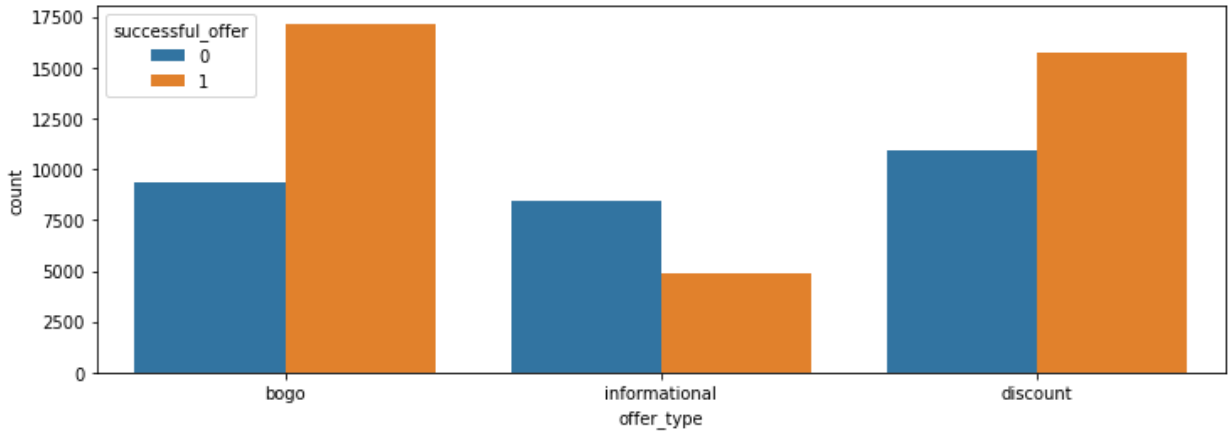
- The above chart shows that customer income increases as customer age increases. This is something that has been expected as older people is more likely to get more money
- Moreover, there is no indication of any pattern of the income or the customer age with the final label

Gender



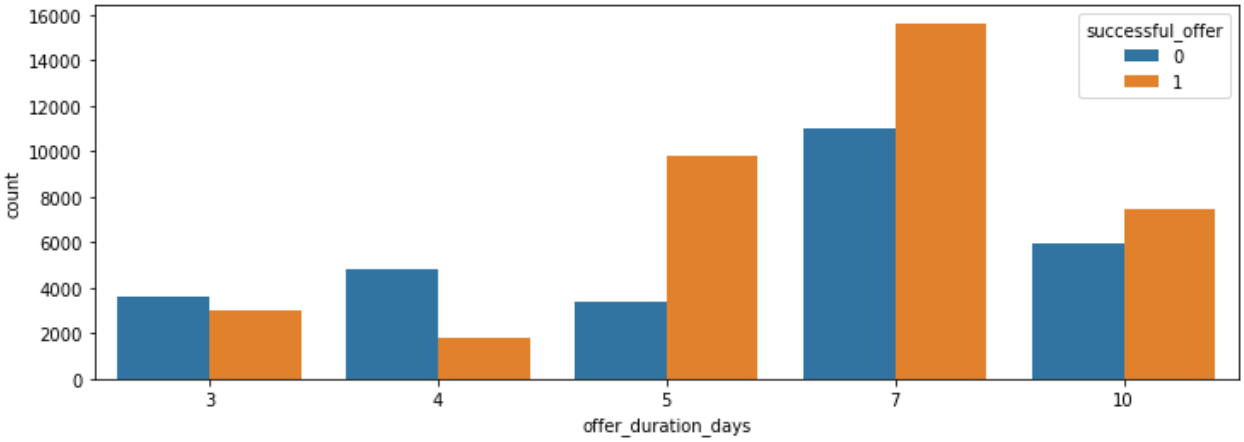
- As we can see there is dependency between the dependent variable and the *Gender* of the customers
- Moreover, we can see that the biggest proportion of the customers are Men
- Also, the number of customers identified as “O” *Gender* are about 2% of the total population and as a result this feature should be dropped after the encoded of this feature

Offer Type



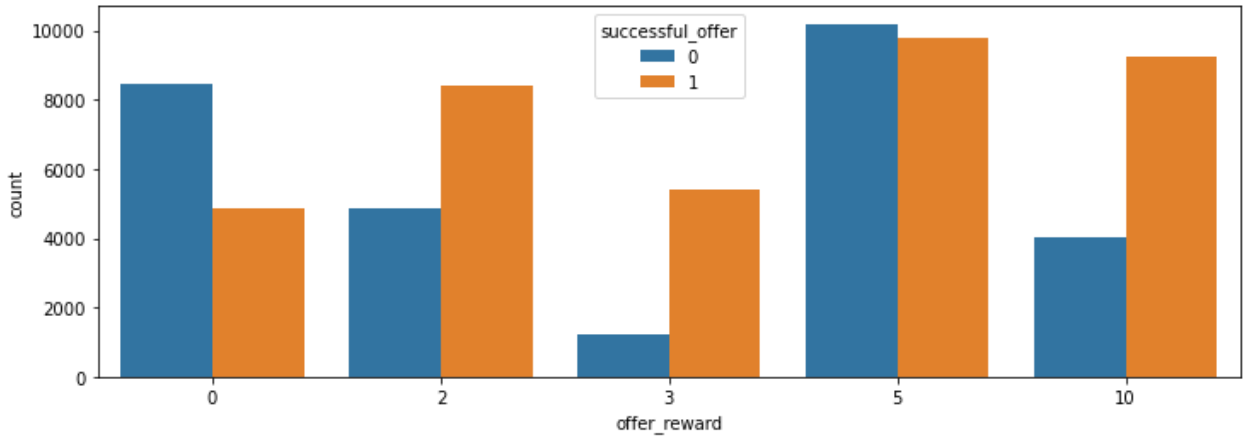
- It is clear that BOGO and Discount offers are more likely to be successful
- Moreover, BOGO offers seem to have about 17500 successful offers and no more than 10000 unsuccessful
- On the other hand, informational offers do not have much success

Offer Duration



- Offers with duration 3 and 4 are more likely to be unsuccessful. This is something that is expected as the customers may not have the time to spend the money needed for a discount offer or to complete a BOGO offer
- Offers with duration about 5 days are really likely to be successful

Offer Reward



- It is clear that offers with reward 2, 3 and 10 are really likely to be successful
- However, offers with a 0 reward or 5 are more likely to do not be completed with success

Registration Year

- Most customers seem to be registered in the latest years (2016, 2017 & 2018) but there is no clear pattern between the registration year and the label

b. Algorithms and Techniques

As it can be seen from the final dataset above, the problem that this project is going to solve is a binary classification problem. This means that the final label takes only two values which in our case is 0 for an unsuccessful offer and 1 for a successful offer.

However, there are many features which have to be processed and transformed before they can be fed in the machine learning algorithms and the detailed steps followed can be seen in the next section (**Methodology**).

Also, after the preprocessing is completed the final dataset need to be splitted into three separate datasets which are:

- *Training set:* Dataset to be used to train the models
- *Validation set:* Dataset to be used to tune the XGBoost algorithm
- *Test set:* Dataset to be used to test the performance of the trained models

*__Detailed steps can bee seen in the following section (Methodology)__

So, the techniques that are used to use this project are techniques related to a supervised machine learning problem, where we have several features and one known label. There are many algorithms which are suitable for solving binary classification problems like this one, but the one selected to be used is an **XGBoost classifier**.

XGBoost is an ensemble boosting tree algorithm which is considered as a really powerful and easy to be trained model. Moreover, it is known as one of the best

algorithms for binary classification problems where there is a validation set ready to be used for tuning the hyperparameters.

Default hyperparameters used to construct an initial XGBoost model and the HyperparameterTuner object from AWS used to find the best hyperparameters for the final model

c. Benchmark

A decision tree classifier will be used as a benchmark model. Both models (decision tree and XGBoost) will be trained and tested at identical datasets. In addition, the same evaluation metrics will be used, so that the results can be compared.

An XGBoost algorithm is an ensemble method, which consists of individual simple models and, more specifically, decision trees. This is why a simple decision tree, which is considered as a really great algorithm for binary classification problems, has been selected as the best benchmark model in our problem.

III. Methodology

a. Data Preprocessing

There are some issues identified from the Exploratory Data Analysis above and needed to be handled before splitting the dataset into training, validation and testing sets. On this section we will go through the steps followed to preprocess and clean each dataset and the steps followed to prepare the final dataset after merging the three main datasets together

I. portfolio

- No actual need of cleaning the values of any of the columns as there are no null values or outliers on the dataset
- There is a need to rename the existing column names to more representative feature names, which will be used later on in the final dataframe. The changes in the names can be seen below:
    - *id -> offer_id*
    - *reward -> offer_reward*
    - *duration -> offer_duration_days*
    - *difficulty -> offer_difficulty*
- There is a categorical column which needs to be encoded. This is the *channels* column
- The final step is to remove the *channels* column and add the encoded columns created
- The resulted cleaned and transformed version of the portfolio dataset can be seen below:

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

II. profile

- There are no extreme values for any of the columns. However, there are approximately 12.8% of missing values which should be removed.
- The *became_member_on* columns cannot be used as it is in the final datasets. So, the year will be extracted to be used only as it seems that the most value is captured there
- There is a need to rename the existing column names to more representative feature names, which will be used later on in the final dataframe. The changes in the names can be seen below:
    - *id -> customer_id*
    - *age-> offer_reward*
    - *became_member_on -> customer_registration_year*
    - *income-> customer_income*
- The resulted cleaned and transformed version of the profile dataset can be seen below:

	customer_id	customer_age	customer_gender	customer_income	customer_registration_year
0	0610b486422d4921ae7d2bf64640c50b	55	F	112000.0	2017
1	78afa995795e4d85b5d9ceeca43f5fef	75	F	100000.0	2017
2	e2127556f4f64592b11af22de27a7932	68	M	70000.0	2018
3	389bc3fa690240e798340f5a15918d5c	65	M	53000.0	2018
4	2eeac8d8feae4a8cad5a6af0499a211d	58	M	51000.0	2017
...	...	...	...	...	...
14820	6d5f3a774f3d4714ab0c092238f3a1d7	45	F	54000.0	2018
14821	2cb4f97358b841b9a9773a7aa05a9d77	61	M	72000.0	2018
14822	01d26f638c274aa0b965d24cefe3183f	49	M	73000.0	2017
14823	9dc1421481194dcd9400aec7c9ae6366	83	F	50000.0	2016
14824	e4052622e5ba45a8b96b59aba68cf068	62	F	82000.0	2017

14825 rows × 5 columns

III. transcript

- No actual need of cleaning the values of any of the columns as there are no null values or outliers identified
- Need to remove the *value_eda* columns which has been created in the Exploratory Data Analysis step
- Need to rename the key values in the value column from *offer id* to *offer_id,* for any instance that this is true
- Need to remove any customer that does not appear in the profile dataset. It seems that some customers ids are not included in the profile dataset and need to be removed
- Need to rename the existing column names to more representative feature names:
  - *person -> customer_id*
  - *time -> time_hours*
- The resulted cleaned and transformed version of the transcript dataset can be seen below:

	customer_id	event	value	time_hours
0	78afa995795e4d85b5d9ceeca43f5fef	offer received	{'offer_id': '9b98b8c7a33c4b65b9aebfe6a799e6d9'}	0
1	e2127556f4f64592b11af22de27a7932	offer received	{'offer_id': '2906b810c7d4411798c6938adc9daaa5'}	0
2	389bc3fa690240e798340f5a15918d5c	offer received	{'offer_id': 'f19421c1d4aa40978ebb69ca19b0e20d'}	0
3	2eeac8d8feae4a8cad5a6af0499a211d	offer received	{'offer_id': '3f207df678b143eea3cee63160fa8bed'}	0
4	aa4862eba776480b8bb9c68455b8c2e1	offer received	{'offer_id': '0b1e1539f2cc45b7b9fa7c272da2e1d7'}	0
...	...	...	...	...
272757	24f56b5e1849462093931b164eb803b5	offer completed	{'offer_id': 'fafdcd668e3743c1bb461111dcafc2a4', 'reward': 2}	714
272758	b3a1272bc9904337b331bf348c3e8c17	transaction	{'amount': 1.5899999999999999}	714
272759	68213b08d99a4ae1b0dcb72aebd9aa35	transaction	{'amount': 9.53}	714
272760	a00058cf10334a308c68e7631c529907	transaction	{'amount': 3.61}	714
272761	76ddbd6576844afe811f1a3c0fbb5bec	transaction	{'amount': 3.5300000000000002}	714

272762 rows × 4 columns

IV. final_df

- Categorical features:
  - One hot encode *offer_type* column
  - One hot encode *customer_gender* column
  - One hot encode *customer_registration_year* column
- Numerical features:
  - Scale and normalize *customer_age* column
  - Scale and normalize *customer_income* column
  - Scale and normalize *offer_duration_days* column
  - Scale and normalize *offer_difficulty* column
  - Scale and normalize *offer_reward* column
- Gender *O* has less than 2% of data and as result this column should be dropped because it does not bring any value to the final models
- Channel *email* appears to all offers, so 100% of its values are constant. As a result this column should be dropped
- Gender *M* is highly correlated with gender *F*, so one of the two has to be dropped. This is because highly correlated features do not work well in machine learning models and may negatively influence the performance of them
- There are some duplicate rows which should be dropped
- The resulted cleaned and transformed version of the final dataset can be seen below:

	customer_age	M	customer_income	2013	2014	2015	2016	2017	2018	bogo	discount	informational	offer_duration_days	offer_difficulty	offer_reward	mobile	social	web	successful_offer
0	0.686747	0	0.777778	0	0	0	0	1	0	1	0	0	0.571429	0.25	0.5	1	0	1	1
1	0.602410	1	0.444444	0	0	0	0	0	1	1	0	0	0.571429	0.25	0.5	1	0	1	1
2	0.566265	1	0.255556	0	0	0	0	0	1	1	0	0	0.571429	0.25	0.5	1	0	1	1
3	0.457831	0	0.644444	0	0	0	0	0	1	1	0	0	0.571429	0.25	0.5	1	0	1	1
4	0.457831	0	0.644444	0	0	0	0	0	1	1	0	0	0.571429	0.25	0.5	1	0	1	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
55447	0.361446	1	0.311111	0	0	0	0	0	1	1	0	0	0.285714	0.50	1.0	1	1	1	1
55448	0.313253	0	0.566667	0	0	0	1	0	0	1	0	0	0.285714	0.50	1.0	1	1	1	1
55449	0.349398	1	0.711111	0	0	0	0	1	0	1	0	0	0.285714	0.50	1.0	1	1	1	0
55450	0.518072	0	0.333333	0	1	0	0	0	0	1	0	0	0.285714	0.50	1.0	1	1	1	1
55451	0.481928	0	0.533333	0	0	0	1	0	0	1	0	0	0.285714	0.50	1.0	1	1	1	1

55452 rows × 19 columns

b. Implementation

I started this project by reading the three provided datasets and exploring their characteristics for any anomalies or interesting patterns, through exploratory data analysis techniques. The main focus in the beginning was to fully understand the data provided by Udacity and Starbucks, so as to decide what will be the problem I will solve and what data and features are available.

After taking a closer look in the exploratory data analysis of each dataset, I started performing some initial data cleaning and transformation steps. These can be considered as initial steps, as the final steps for transforming the datasets, implementing when the three datasets combined together. Some of the cleaning and transformation steps are:

- Drop any duplicate rows
- Replace or drop missing values
- Rename the features' names
- One hot encoding categorical features
- Remove any unnecessary columns

Moreover, during the cleaning and transformation process, the label of our final dataset has been constructed. The idea of this label was to be a binary label which will indicate if an offer was successful or not. In order to do this we have used the data provided in the transcript dataset with the below logic:

- Offer Received -> Offer Viewed -> Offer Completed ← **successful offer**

The next step was to combine the cleaned and transformed datasets into one final dataset which has been used for further exploratory data analysis. Exploratory data analysis has been performed in order to identify any correlation between the independent features, but at the same time any correlation of our target variable with the available features.

Then the final table has been created by cleaning and transforming the merged dataframe using the patterns and issues identified in the exploratory data analysis. The techniques used to transform the numerical and categorical features are the "*MinMaxScaler", "LabelBinarizer"* and *"MultiLabelBinarizer"* provided by sklearn package.

After having the final dataset ready, I had to split the data into training, validation and testing sets and at the same time I created a metrics_report function to be used for evaluating all models using the same metrics.

It has been observed that the training dataset was not well-balanced. So, during the splitting process a logic has been used in order to adjust and balance the dataset on the offer type level. This means that the number of successful and unsuccessful offers has been adjusted with a ratio of 1:1 for each type of offer (BOGO, discount, informational.

The main metric used to evaluate the models' performance was the accuracy as it is the most known and most used evaluation metric and it has been used by many other people completing similar projects. Moreover, as we do not care more for the False Positive Rate or the False Negative Rate, there was no need to focus on the recall or the precision of each model. The definition of the accuracy can be seen below (more details can be found in the Metrics section above.

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}}$$

Then, after having the three final datasets ready (training, validation & testing sets) I have tried several models using the sklearn package to evaluate their performance. It was clear that ensemble tree based algorithms provided the best results.

So, a Decision tree algorithm has been used as a Benchmark model and an XGBoost classifier has been selected as the final model to be used to solve the problem defined above. The reason behind the selection of these two algorithms can be found in the Analysis section above.

Moreover, the idea was to construct a model for each type of offer and at the same time to have an overall model for all offers combined together. I wanted to do that as I wanted to create an automated process, where we will be able to see what is the probability of successfully completing each offer type, for each customer. This would be able to provide more details to the marketing team before making the final decision about the offer type to send on each customer. However, the output of a Decision tree is not a number from 0 to 1 and as a result this was not possible, but an XGBoost classifier provides the probability for each result and I was able to use this in order to get more information for each offer type.

So, in total the final models used for the summarized results of this project were:

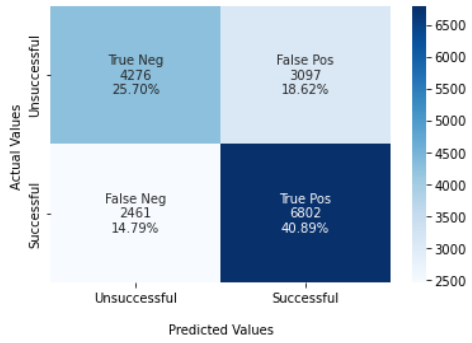
1. XGBoost model using all offer types
2. XGBoost model using only BOGO offers
3. XGBoost model using only discount offers
4. XGBoost model using only informational offers

Decision tree example

```
predictions = pd.read_csv(os.path.join(data_dir, 'all_
y_pred = [round(num) for num in predictions.squeeze().

metrics_report(y_test, y_pred)
```

```
The overall accuracy is:66.59053%
              precision    recall  f1-score   support

           0       0.63      0.58      0.61      7373
           1       0.69      0.73      0.71      9263

    accuracy                           0.67     16636
   macro avg       0.66      0.66      0.66     16636
weighted avg       0.66      0.67      0.66     16636
```

Seaborn Confusion Matrix with labels




```
predictions.values
```

```
array([[0.64045531],
       [0.66430914],
       [0.67150784],
       ...,
       [0.69443542],
       [0.55591857],
       [0.7079705 ]])
```

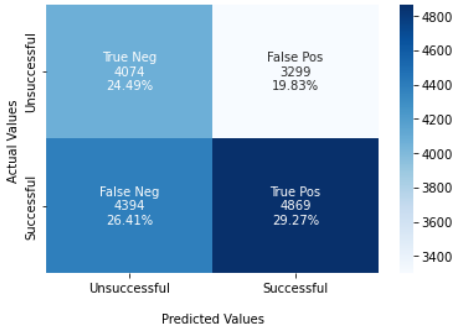
XGBoost example

```
from sklearn import tree

clf = tree.DecisionTreeClassifier()
clf = clf.fit(X_train, y_train)
```

```
y_pred = clf.predict(X_test)
metrics_report(y_test, y_pred)
```

```
The overall accuracy is:53.75691%
              precision    recall  f1-score   support

           0       0.48      0.55      0.51      7373
           1       0.60      0.53      0.56      9263

    accuracy                           0.54     16636
   macro avg       0.54      0.54      0.54     16636
weighted avg       0.55      0.54      0.54     16636
```

Seaborn Confusion Matrix with labels




```
clf.predict_proba(X_test)
```

```
array([[0., 1.],
       [0., 1.],
       [1., 0.],
       ...,
       [0., 1.],
       [0., 1.],
       [0., 1.]])
```

c. Refinement

First step was to build an XGBoost classifier using the xgboost package provided publicly available in python. I used this initial model with default (random) parameters to see if it performs and if it performs better compared to the benchmark model.

After looking at the results and deciding that the model I will use is an XGBoost classifier, I decided to use the machines provided through AWS to complete the hyperparameter tuning job. In the pictures below you can see the default parameters used to fit the initial model and the tuning job. The metric used to find the best performing model (best hyperparameters) was the accuracy.

Default parameters

```
xgb.set_hyperparameters(max_depth=5,
                        eta=0.2,
                        gamma=4,
                        min_child_weight=6,
                        subsample=0.8,
                        objective='binary:logistic',
                        early_stopping_rounds=10,
                        num_round=200)
```

Hyperparameter tuning job

```
from sagemaker.tuner import IntegerParameter, ContinuousParameter, HyperparameterTuner

xgb_hyperparameter_tuner = HyperparameterTuner(estimator = xgb, # The estimator object to use as the basis for the training jobs.
                                               objective_metric_name = 'validation:rmse', # The metric used to compare trained models.
                                               objective_type = 'Minimize', # Whether we wish to minimize or maximize the metric.
                                               max_jobs = 20, # The total number of models to train
                                               max_parallel_jobs = 3, # The number of models to train in parallel
                                               hyperparameter_ranges = {
                                                   'max_depth': IntegerParameter(3, 12),
                                                   'eta'      : ContinuousParameter(0.05, 0.5),
                                                   'min_child_weight': IntegerParameter(2, 8),
                                                   'subsample': ContinuousParameter(0.5, 0.9),
                                                   'gamma': ContinuousParameter(0, 10),
                                               })
```

Hyperparameters selected

Hyperparameters

Key	Value
_tuning_objective_metric	validation:rmse
early_stopping_rounds	10
eta	0.05489787560936668
gamma	7.305262631680812
max_depth	4
min_child_weight	7
num_round	200
objective	binary:logistic
subsample	0.5167894667665632

The same logic and process followed for the four XGBoost models defined and trained. Details about the four models can be found in the previous section (Implementation)

IV. Results

a. Model Evaluation and Validation

After completing all the steps described in the Methodology section above, I chose to use the best performing model based on the hyperparameter tuning job performed on AWS.

As I have already explained in the section above, the final models used on this project were four in total (1 overall and 1 for each offer type) and the process for finding the best parameters was the same for all of them. The algorithm used was an XGBoost classifier for all.

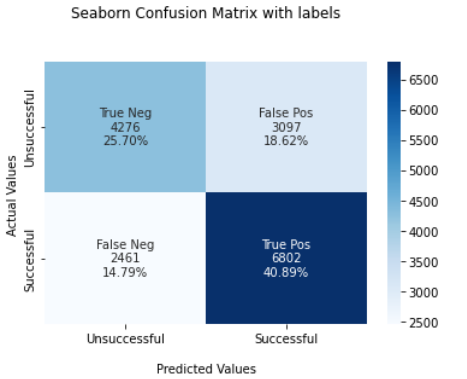
All models were evaluated on unseen data in order to reflect a real world situation where we want to see what is the most appropriate offer for each customer.

The results show that all models were able to identify patterns and generalize well on unseen data. Results were way better compared to the Benchmark models' used on both training, validation and testing sets.

XGBoost Classifier using all offers

- Accuracy is more than 66.5% which means that the model predicted correctly 11,078 Cases against 5.558 errors.
- The number of True Positives is really big and covers more than 40% of all cases
- Moreover, both precision and recall are above 58% which saws a strong predictive power for both classes

```
The overall accuracy is:66.59053%
              precision    recall  f1-score   support

           0       0.63      0.58      0.61      7373
           1       0.69      0.73      0.71      9263

    accuracy                           0.67     16636
   macro avg       0.66      0.66      0.66     16636
weighted avg       0.66      0.67      0.66     16636
```

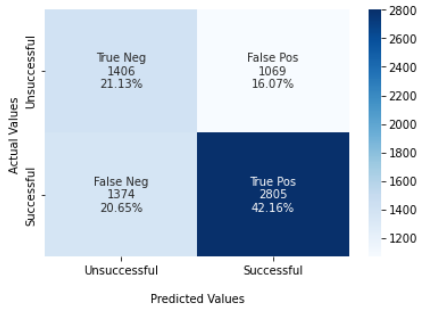


XGBoost Classifier using BOGO offers only

- Results on this model are pretty much the same with the overall model
- Main difference is that the number of True and False Negatives are really close which means that the model is not too good on indentifying unsuccessful offers.
- The number of True Positives is really big and covers more than 42% of all cases

The overall accuracy is:63.28524%

	precision	recall	f1-score	support
0	0.51	0.57	0.54	2475
1	0.72	0.67	0.70	4179
accuracy			0.63	6654
macro avg	0.61	0.62	0.62	6654
weighted avg	0.64	0.63	0.64	6654

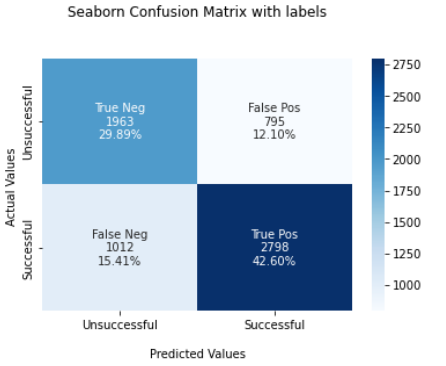


XGBoost Classifier using discount offers only

- Accuracy on this model is more quite better with an average score of more than 72.4%
- Moreover, the number of false positives is less than 12.5% with a precision close to 80%
- The number of True Positives is really big Onces again and more specifically around 42.6%

The overall accuracy is:72.48782%

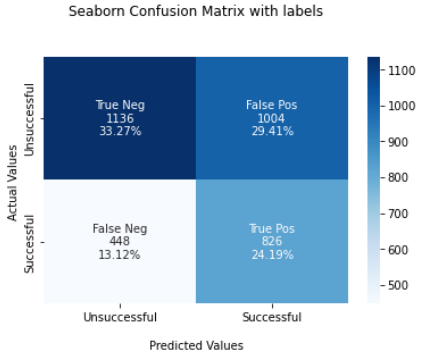
	precision	recall	f1-score	support
0	0.66	0.71	0.68	2758
1	0.78	0.73	0.76	3810
accuracy			0.72	6568
macro avg	0.72	0.72	0.72	6568
weighted avg	0.73	0.72	0.73	6568



XGBoost Classifier using informational offers only

The overall accuracy is:57.46924%

	precision	recall	f1-score	support
0	0.72	0.53	0.61	2140
1	0.45	0.65	0.53	1274
accuracy			0.57	3414
macro avg	0.58	0.59	0.57	3414
weighted avg	0.62	0.57	0.58	3414



- Accuracy on this model is the worst compared to the other three models and it is about 57%
- Results show that informational offers are not behaving the same as the other two offers, which was something expected
- Finally, the bad performance of the model can be identified from the confusion matrix where the number of False Positives are more than the number of True Positives (1,004 vs 826)

After completing all the tests for all models as described in the Methodology section, I am convinced that the XGBoost models used to solve this problem are performing really good with an accuracy way better compared to the Benchmark model.

However, informational offers seem to not have the same behavior as the other two, but BOGO and discount offers showed that they have some clear underlying patterns which the models were able to identify and correctly predict their outcome on unseen data.

Moreover, this project determines the likelihood of a customer to complete each offer and we will see more details on that in the next section (Conclusion).

b. Justification

In this project, two different models have been trained to recognize patterns in the dataset and predict the outcome of all offers together or of each offer individually. Moreover, the overall goal of this project was to provide the likelihood of one customer to successfully complete an offer based on its type, but only one of the models was able to provide this output.

So, in total eight models have been trained and tested on different datasets. Four models for the benchmark approach (1 for all offers and 1 for each type of offer) and four models for the proposed approach. Below we can see the results of each models trained and tested in the corresponding dataset.

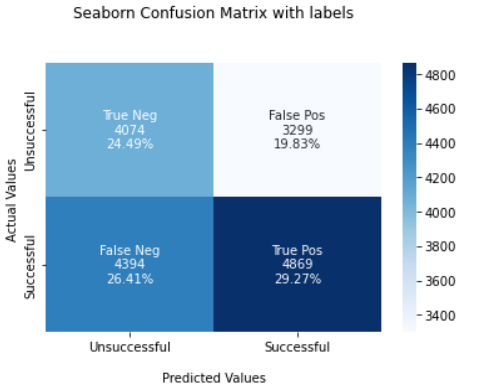
Benchmark

The benchmark model used is a Decision Tree classifier which is considered one of the most known and used models for binary classification problems.

Decision Tree with all offers

The overall accuracy is:53.75691%

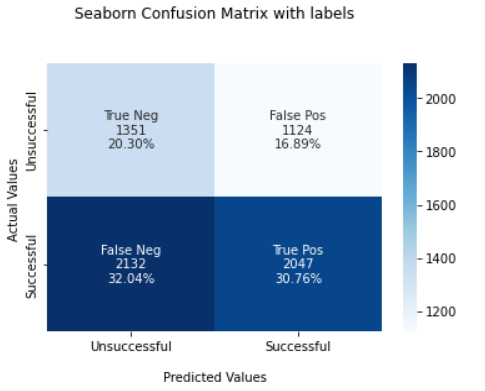
	precision	recall	f1-score	support
0	0.48	0.55	0.51	7373
1	0.60	0.53	0.56	9263
accuracy			0.54	16636
macro avg	0.54	0.54	0.54	16636
weighted avg	0.55	0.54	0.54	16636



Decision Tree with BOGO offers

The overall accuracy is:51.06703%

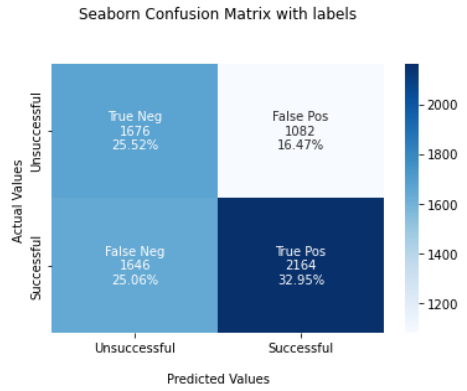
	precision	recall	f1-score	support
0	0.39	0.55	0.45	2475
1	0.65	0.49	0.56	4179
accuracy			0.51	6654
macro avg	0.52	0.52	0.51	6654
weighted avg	0.55	0.51	0.52	6654



Decision Tree with all discount offers

The overall accuracy is:58.46529%

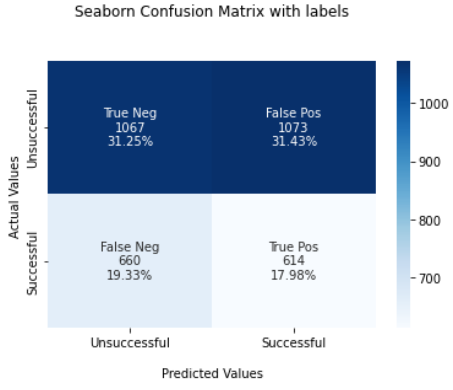
	precision	recall	f1-score	support
0	0.50	0.61	0.55	2758
1	0.67	0.57	0.61	3810
accuracy			0.58	6568
macro avg	0.59	0.59	0.58	6568
weighted avg	0.60	0.58	0.59	6568



Decision Tree with informational offers

The overall accuracy is:49.23843%

	precision	recall	f1-score	support
0	0.62	0.50	0.55	2140
1	0.36	0.48	0.41	1274
accuracy			0.49	3414
macro avg	0.49	0.49	0.48	3414
weighted avg	0.52	0.49	0.50	3414



Looking at the results provided above from the benchmark model and at the results in the *Model Evaluation and Validation* section above, we can clearly see that the tuned Hyperparameter XGBoost classifier performs way better.

Moreover, the results from the Decision Trees classifiers are really close to a random choice (accuracy close to 50%) for each case individually and this indicates that the problem this project tries to solve is really difficult and a simple model is not able to identify the underlying patterns.

So, it is clear that the proposed XGBoost classifiers are really good for solving this particular problem and the parameters used and tuned are well selected.

V. Conclusion

a. Free-Form Visualization

So, now we can imagine how the models constructed and trained on this project will be used in a real world scenario and in a day to day business case.

If we just looked at historical data, we would predict the probability of a customer to complete a particular offer. By doing that, we would not take into account if a customer is maybe interested in another offer, or if the customer is saturated with a specific offer.

Having this in mind, the overall idea behind this project was to create an automated process that could be used by the marketing team, when a customer list is available and they want to identify the most appropriate offer for each one of them.

Below we can see a screenshot of four unseen customers provided in the automated function and the results for each one of them:

```
--------------------------------------------------Customer with index 26 --------------------------------------------------
If the customer receives only a discount offer, the likelihood of succefull completing the offer is: 0.33
This customer is suitable for a bogo offer with a confidence of: 37.87%
This customer is suitable for a discount offer with a confidence of: 34.44%
This customer is suitable for a informational offer with a confidence of: 46.39%

******************** Most suitable offer for this customer is a INFORMATIONAL offer ********************


--------------------------------------------------Customer with index 27 --------------------------------------------------
If the customer receives only a bogo offer, the likelihood of succefull completing the offer is: 0.45
This customer is suitable for a bogo offer with a confidence of: 36.56%
This customer is suitable for a discount offer with a confidence of: 67.09%
This customer is suitable for a informational offer with a confidence of: 28.89%

******************** Most suitable offer for this customer is a DISCOUNT offer ********************


--------------------------------------------------Customer with index 28 --------------------------------------------------
If the customer receives only a discount offer, the likelihood of succefull completing the offer is: 0.82
This customer is suitable for a bogo offer with a confidence of: 53.87%
This customer is suitable for a discount offer with a confidence of: 82.4%
This customer is suitable for a informational offer with a confidence of: 45.64%

******************** Most suitable offer for this customer is a DISCOUNT offer ********************


--------------------------------------------------Customer with index 29 --------------------------------------------------
If the customer receives only a discount offer, the likelihood of succefull completing the offer is: 0.26
This customer is suitable for a bogo offer with a confidence of: 31.02%
This customer is suitable for a discount offer with a confidence of: 26.61%
This customer is suitable for a informational offer with a confidence of: 29.81%

******************** Most suitable offer for this customer is a BOGO offer, BUT THE LIKELIHOOD OF COMPLETING IS BELOW 0.5. BETTER TO SEND AN INFORMATIONAL OFFER ONLY *
```

As we can see there are 5 different outputs for each customer:

- The first output is the likelihood associated with the offer suggested for each customer (marketing team can provide the offer suggested for each customer and see what is the likelihood of successfully completing it).

- The next three results are related to the confidence associated with each offer type based on the probability of the customer to successfully complete each offer type respectively.
- Then, the final and the most important output is the most suitable offer type for each customer. This output is based on each model's output, for the customer provided. Moreover, if the likelihood for all offers is below 0.5, the final output will include a message to indicate this. An example of that can be seen in the picture above for the customer with index 29.

Now, let's take a closer look to 2 different cases from the picture above:

1. *Customer 26*
    - The offer that this customer was suggested to get was a discount offer. As we can see the first model which has information from all offer types, suggested that the likelihood of successfully completing a discount offer is 0.33
    - The three individual models provided scores fluctuate from 34.44% to 46.39%. Lowest score was for a discount offer, which was the one suggested from the team
    - The most suitable offer seems to be an informational offer with confidence 46.39%
    - Marketing team can decide if they want to send only an informational offer to the customer, or maybe send a bogo offer as well which is the second best

2. *Customer 29*
    - The offer that this customer was suggested to get was a discount offer. As we can see the first model suggested that the likelihood of successfully completing a discount offer is really low and close to 0.26
    - The three individual offers all provided really low scores, with the lowest being a discount offer with confidence 26.61%
    - Most suitable offer seems to be a BOGO offer but the automated framework suggested to consider an informational offer as an option as the confidence for the BOGO offer is below 50%

Looking at the details for two different customers we can get a feeling of the power of machine learning in making correct decisions, by looking at historical data. Marketing teams can benefit a lot from information like these as they can make more accurate decisions and get more value.

b. Reflection

It was a pleasure working on this project as the final capstone project for the Machine Learning Engineer nanodegree. This project gave me the opportunity to apply in practice all the knowledge I have acquired from the nanodegree, but also from my academic and working experience.

I had already worked and completed several other projects, but this one was one of the most interesting projects I have ever done, as the data associated with it was real data coming from the Starbucks app. Moreover, the nature of the project was amazing as it gave me the freedom to apply any algorithm and technique I want in order to solve any problem I believe is interesting/important to be solved.

However, this was really tricky at the same time, as I really struggled to decide what would be the problem to solve using the provided datasets. One of the most difficult parts from me was to decide what a successful offer would look like and construct the label accordingly

I do believe that my suggested automated framework, which uses four different models to provide useful information to the end user and also to compare the results from the different offer types, is really powerful and is able to save a lot of time from the marketing team in Starbucks. At the same time, by using the framework provided in my notebook, the marketing team can increase the value associated with each offer by sending more relevant offers to each customer based on their characteristics.

Finally, this project was an outstanding opportunity for me to learn new techniques and do my own research on how to apply machine learning models on marketing related data. Also, this project gave me the opportunity to use my AWS and machine learning knowledge acquired from the Machine Learning Engineer nanodegree in practice and I really enjoyed it.

c. Improvement

I do believe that the solution provided on this project is really powerful and accurate, but there is room for improvement.

More specifically, there is financial data provided on the transcript dataset which can be analyzed and be used for identifying the value associated with each offer type

per customer. This would provide some extra information in the model which would be able to analyze the value associated with each offer.

For example there are some customers which are expected to not increase their spendings and do not change their behavior when successfully completing an offer. These customers should not be considered a good group for sending any BOGO or discount offer even if they have successfully completed these offers in the past.

So, this the most important improvement can be added on this project and I am planning to use the available financial data in the next couple of months and see if there is a significant change in the results associated with each customer.

VI. References

[1] Nicolas Guan, "Who Might Respond to Starbucks' offer?"

[2] Silvio Mori Neto, "Starbuck's Capstone Challenge"

[3] Cheb Wabgm Cgebgyuan Deng, Suzhen Wang, "Imbalance-XGBoost: leveraging weighted and focal losses for binary label-imbalanced classification with XGBoost"

[4] Classification: Accuracy

[5] Josh Xin Jie Lee, "Implementing a Profitable Promotional Strategy for Starbucks with Machine Learning"

[6] Didrik Nielsen, "Why Does XGBoost Win "Every" Machine Learning Competition?"