SENDING PROMOTIONAL OFFERS MORE EFFICIENTLY @ STARBUCKS

Author: Gabriel Fernandes Luz (@gfluz94)
Date: December 17h, 2021

1. Definition

1.1. Project Overview

The fundamental problem we want to address is customer behavior and responsiveness while using Starbucks rewards mobile app. Once every few days, Starbucks sends out an offer to users of the mobile app. An offer can be merely an advertisement for a drink or an actual offer such as a discount or BOGO (buy one get one free). Some users might not receive any offer during certain weeks.

Every offer has a validity period before it expires. As an example, a BOGO offer might be valid for only 5 days. You'll see in the data set that informational offers have a validity period even though these ads are merely providing information about a product; for example, if an informational offer has 7 days of validity, you can assume the customer is feeling the influence of the offer for 7 days after receiving the advertisement.

Users may respond or not to those offers. For every offer we went via the application, we spend some money. Therefore, it is very important to target well our customers. Additionally, some customers will only make purchases when they receive a promotional offer, and this is crucial for business success.

Knowing to which user a given promotional offer should be sent is an essential tool for the optimization of resources allocated to marketing campaigns, triggering users to buy the company's products and avoiding churn. Hence, this project finds itself in **marketing** and **customer retention** domains.

A similar work where author applied machine learning to predict customer churn can be found here: Customer churn prediction in telecom using machine learning in big data platform.

1.2. Project Statement

Given a promotional offer and its attributes as well as users and their corresponding features, how likely is it that the user will be triggered by the offer?

In summary, this is a **classification problem**, where the main goal is determining the probability with which customers will respond to eventual offers, so that not only revenues can be optimized, but also churn rates minimized.

1.3. Metrics

In order to evaluate the performance of our model, we can divide our metrics according to their context: classification problem and business impact.

Since we are dealing with a classification problem, we are going to look at common and insightful metrics, such as **recall, precision, F1-score, ROC-AUC and cohen-kappa**. It is important to stress that accuracy is not a useful metric, since our **dataset is imbalanced** – i.e., there are more users that do not respond to offers. Therefore, accuracy could be misleading if models are simply predicting that customers will very likely not respond to our campaigns. Ideally, we would want to focus on recall, because we aim to identify all the users who have responded to our promotional offers. However, we cannot achieve it at the expense of poor precision, since it costs us money to send promotional offers. Therefore, we need to cover as much as possible of our respondent users at the same time that we are able to precisely identify them. That is why F1-Score plays an important role to find the best tradeoff of precision and recall, once it is the harmonic mean of these two metrics.

From the business perspective, **conversion rate and lift** are also going to be calculated. In other words, by targeting users who are likely to respond to a given promotional offer, how much more conversion rate we could have achieved during the test period. This particular analysis will be conducted not only in a global level, but also in a granular one – looking by single offers.

2. Analysis

2.1. Data Exploration

During a 30-day test period, data was collected to measure user behavior upon promotional offers that were sent to them. Hence, there is a relational model available, where transactional data reflects user's actions which can be easily related to metadata about users and offers.

There are three main files:

2.1.1. Profile Dataset: *Rewards program users (17000 users x 5 fields)*

Variable	Type	Observation
gender	Categorical	M, F, O, or null
age	Numeric	Missing value encoded as 118
id	String/Hash	Primary key
became_member_on	Date	YYYYMMDD
income	Numeric	Salary

2.1.2. **Portfolio Dataset:** *Offers sent during 30-day test period (10 offers x 6 fields)*

Variable	Type	Observation
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	Primary key

2.1.3. **Transcript Dataset:** *Event log (306648 events x 4 fields)*

Variable	Type	Observation
person	String/Hash	Foreign key (FK) to Profile dataset
event	String	offer received, offer viewed, transaction, offer completed
value	Dictionary	• *offer_id*: (string/hash) FK to Portfolio dataset • *amount*: (numeric) money spent in "transaction" • *reward*: (numeric) money gained from "offer completed"
time	Numeric	Hours after start of test

We needed to design a new dataset that flags users, offers received and whether or not they responded to it – our target variable to build the classification model later on.

For each pair person-offer in a given point of time, we flagged it as either 1 (user responded to the promotional offer) or 0 (user did not respond to the promotional offer). This was our base dataset to which features will be added later on.

Moreover, we also have taken into account that sometimes users received offers, however they did not see it. Even though, they made organic purchases and the offer was completed in the dataset. This was **not** considered as a customer who responded to an offer, since they did not even know about it.

By converting some of the variables already made available and then performing feature engineering, we were able to obtain our final data:

2.1.4. **Final Dataset**: *Labels and Features (76277 entries x 18 variables)*

Variable	Type	Observation
person	String/Hash	Foreign key (FK) to Profile dataset
offer	String/Hash	Foreign key (FK) to Portfolio dataset
time_in_days_received	Numeric	Only to track time of the test period
responded_customer	Numeric	Label: 1 = responded / 0 = no response
gender	Numeric	M, F, O, or null
age	Numeric	Age or null (already transformed from 118)
income	Numeric	Salary
membership_duration	Numeric	How long user has been a member
difficulty	Numeric	Money required to be spent to receive reward
duration	Numeric	Duration of the offer
offer_type	String	bogo, discount, informational
web	Numeric	1 = channel / 0 = not a channel
email	Numeric	1 = channel / 0 = not a channel
mobile	Numeric	1 = channel / 0 = not a channel
social	Numeric	1 = channel / 0 = not a channel
average_purchase	Numeric	How much user has spent so far, on average
frequency	Numeric	How many purchases user has made so far

Finally, this dataset was split into training and testing sets, so we can study only the training data in order to avoid data leakage.

As mentioned previously, we are dealing with an imbalanced dataset: 27.6% of all the customers have actually responded to promotional offers.

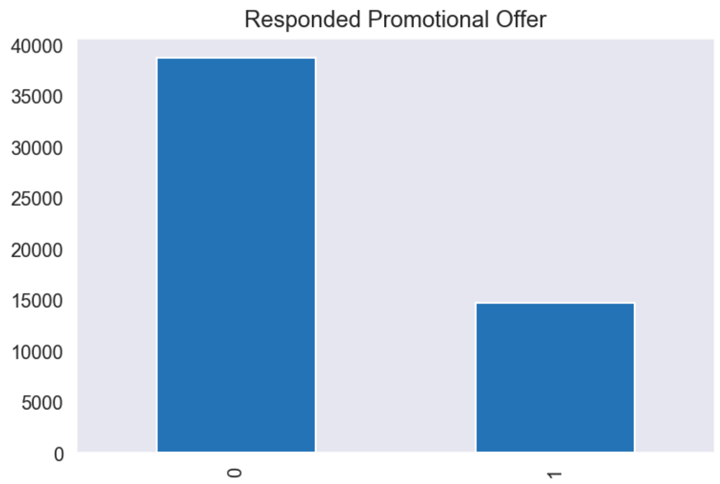

Figure 2.1.1. Label class distribution

Analyzing the distribution of the numeric features, we have the following scenario:

2.1.5. *Numerical Features Statistics*

Measure	age	membership_ duration	reward	difficulty	duration	average_ purchase	frequency
count	46578[1]	53453	53453	53453	53453	53453	53453
mean	54.32	653.42	4.19	7.68	6.49	9.71	3.46
std	17.40	422.44	3.40	5.53	2.20	16.88	3.66
min	18.00	0.00	0.00	0.00	3.00	0.00	0.00
5%	24.00	142.00	0.00	0.00	3.00	0.00	0.00
25%	42.00	343.00	2.00	5.00	5.00	0.00	0.00
50%	55.00	538.00	3.00	7.00	7.00	3.82	3.00
75%	66.00	923.00	5.00	10.00	7.00	16.74	5.00
95%	83.00	1509.00	10.00	20.00	10.00	27.48	11.00
max	101.00	2159.00	10.00	20.00	10.00	962.10	29.00

[1] Missing Values

Most of the variables are well distributed and do not show a lot of outliers. The exceptions are *average_purchase* and *frequency*, since we notice that 95th-percentile and maximum values are very different from each other.

We can conclude that *age* is the variable whose distribution is closer to a normal distribution, even though it is almost bimodal.

The features related to the offers available are limited to certain values, which results in very few representative values.

membership_duration and *frequency* show a skewed distribution, whereas *average_purchase* concetrates almost all values close to zero – in this case, outliers might be a problem.

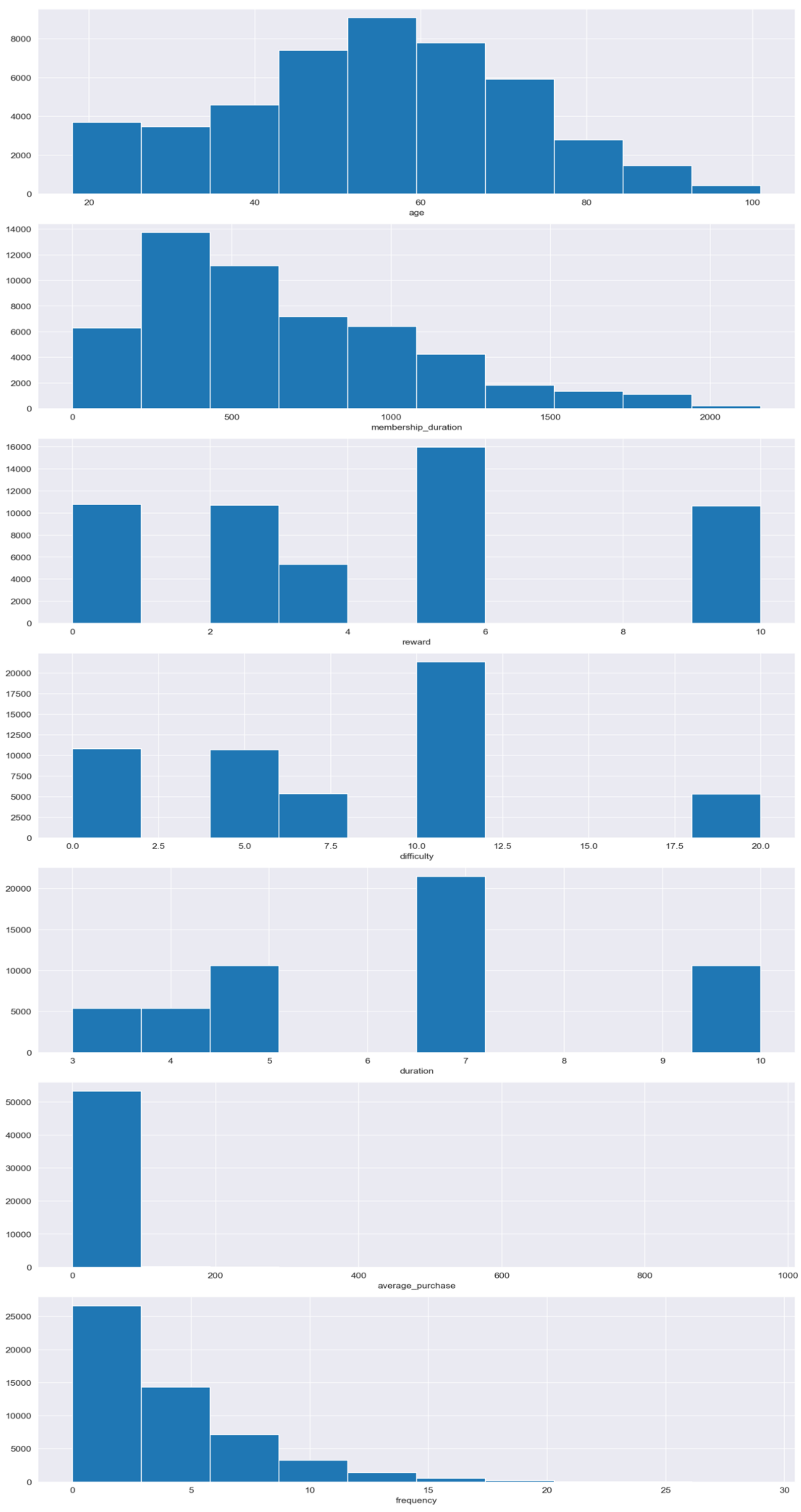

Figure 2.1.2. Numerical Features Distributions

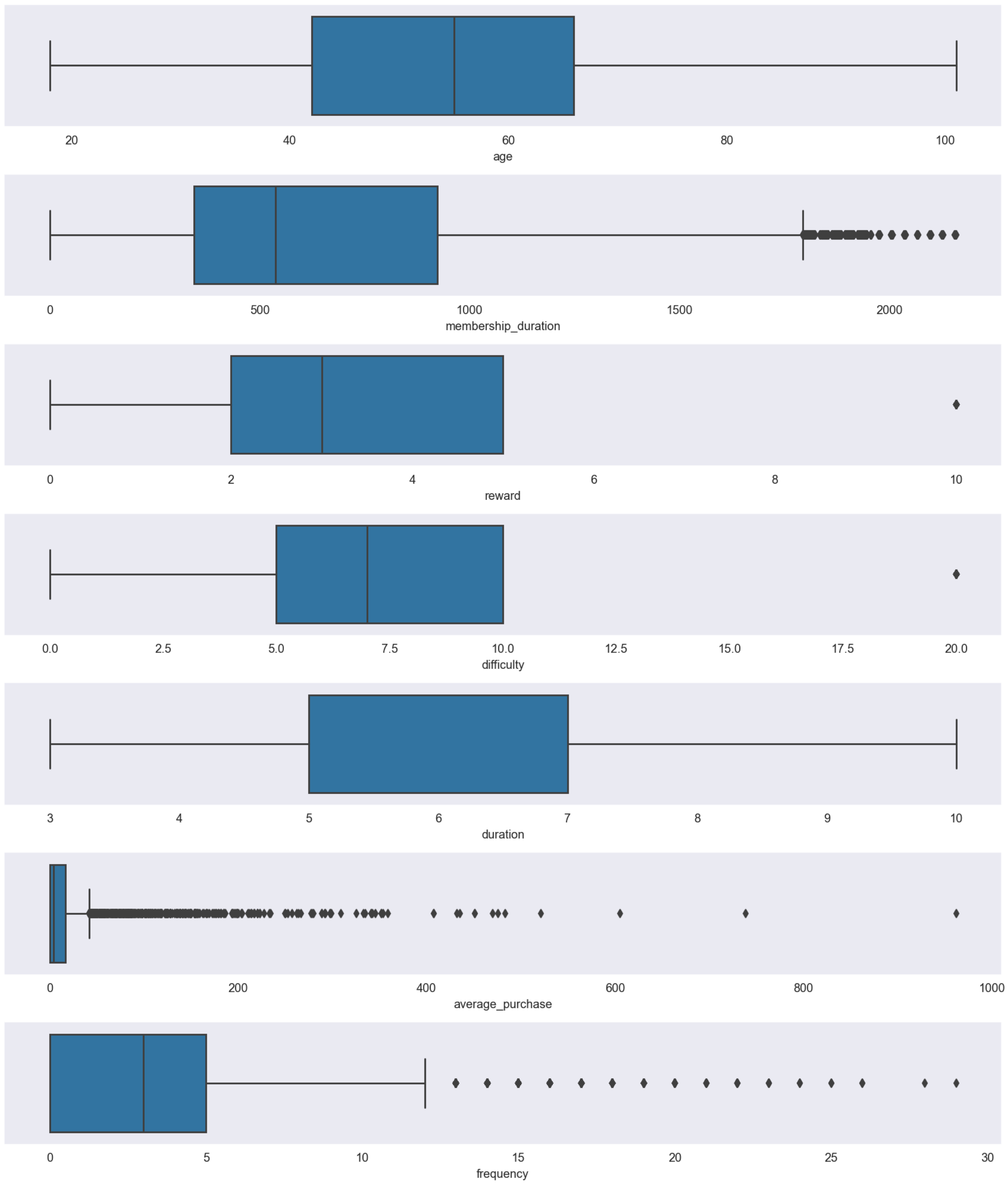

Figure 2.1.3. Numerical Features Boxplots

A further analysis of the numerical features includes a bivariate statistical study which takes into account correlations among features in order to identify highly correlated ones. the only highly-correlated pair is *difficulty* and *duration*. Therefore, depending on the behavior of our target according to both of them, we could decide to follow with only one of them.

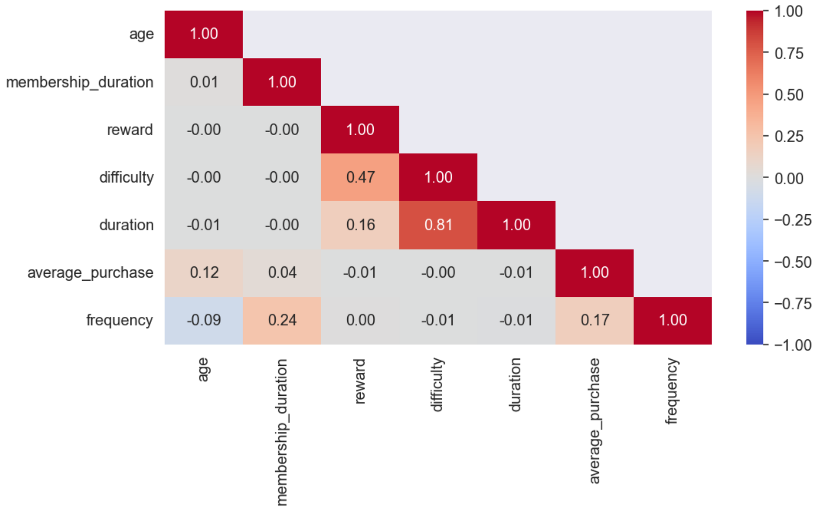

Figure 2.1.4. Correlation Heatmap

The choice between *difficulty* and *duration* is easily solved from the visualization below, where it is easily noticeable how *difficulty* discriminates more against the target variable.

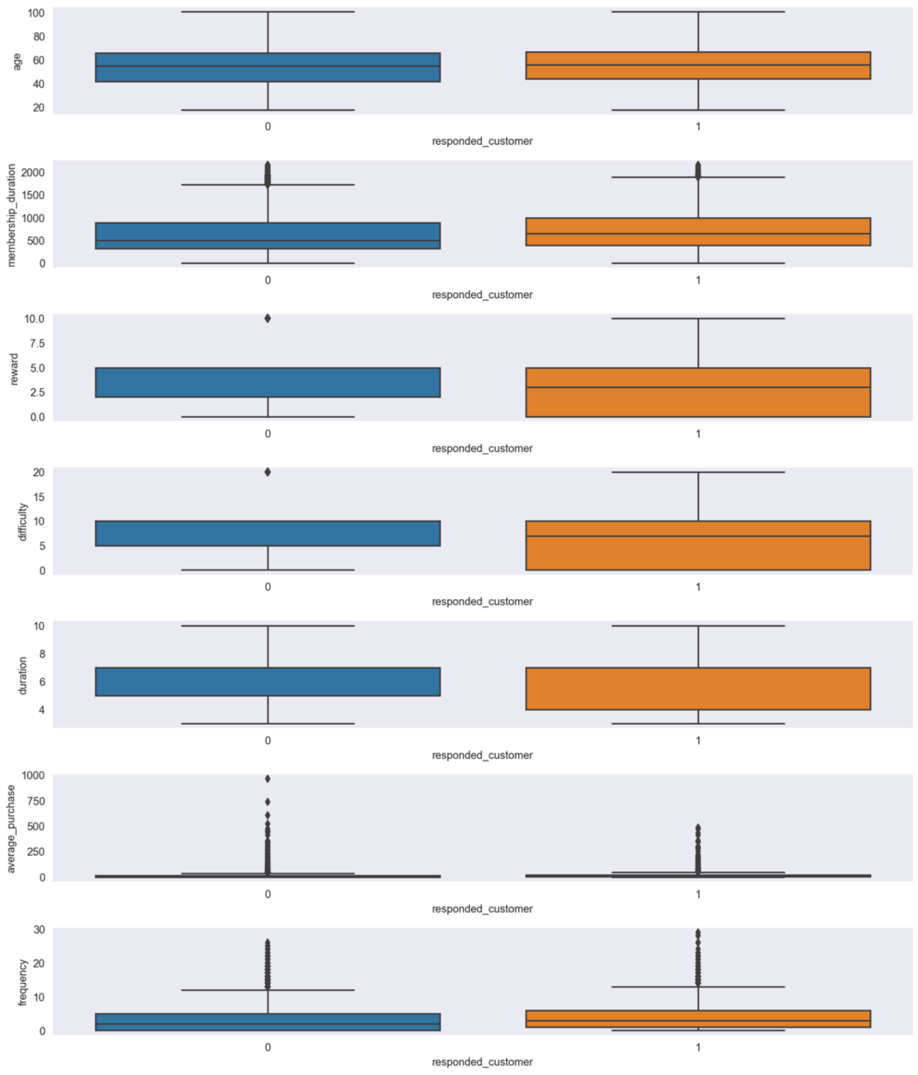

Figure 2.1.5. Boxplot grouped by labels

Finally, regarding categorical features the majority of customers who took part in the 30-day test period are male, and the most common types of offers are BOGO and discount. Informational offers were less often used in this context.

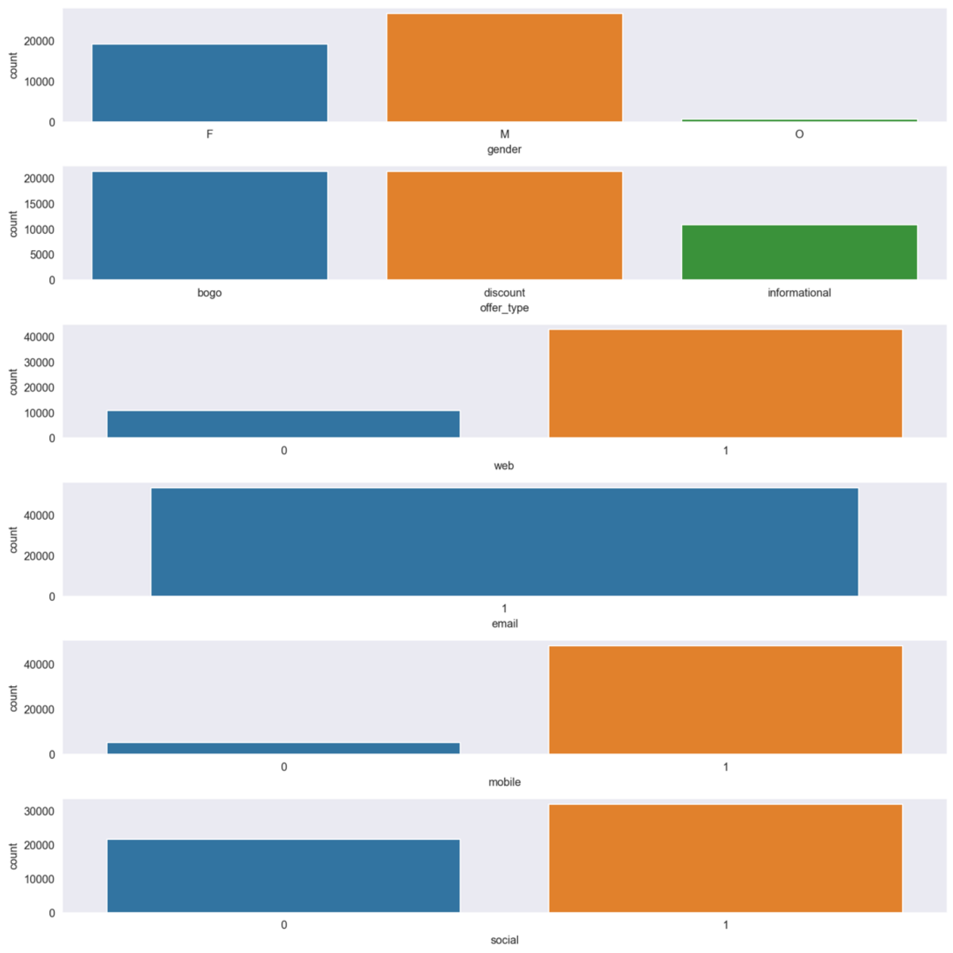

Figure 2.1.6. Count plots for categorical variables

Surprisingly, there is a greater share of respondents in *offer_type* informational in comparison to other types. Regarding *gender*, there are fewer examples of oblivious to conclude anything at all with statistical confidence, so we could infer that females are more prone to responding to a promotional offer than males.

2.2. Algorithms and Techniques

Since the problem is complex by design, we used one of the most advanced algorithms available: **XGBoost**. This model improves on a single decision tree by applying *boosting* technique in order to avoid bias. This is achieved due to the sequential process of adjusting each single tree's output error in the subsequential tree with a given learning rate. Hence, the capability of decreasing bias. However, the risk that plays a large role in this context is increasing variance. In order to avoid it, we need to carefully carry out **hyperparameter tuning**.

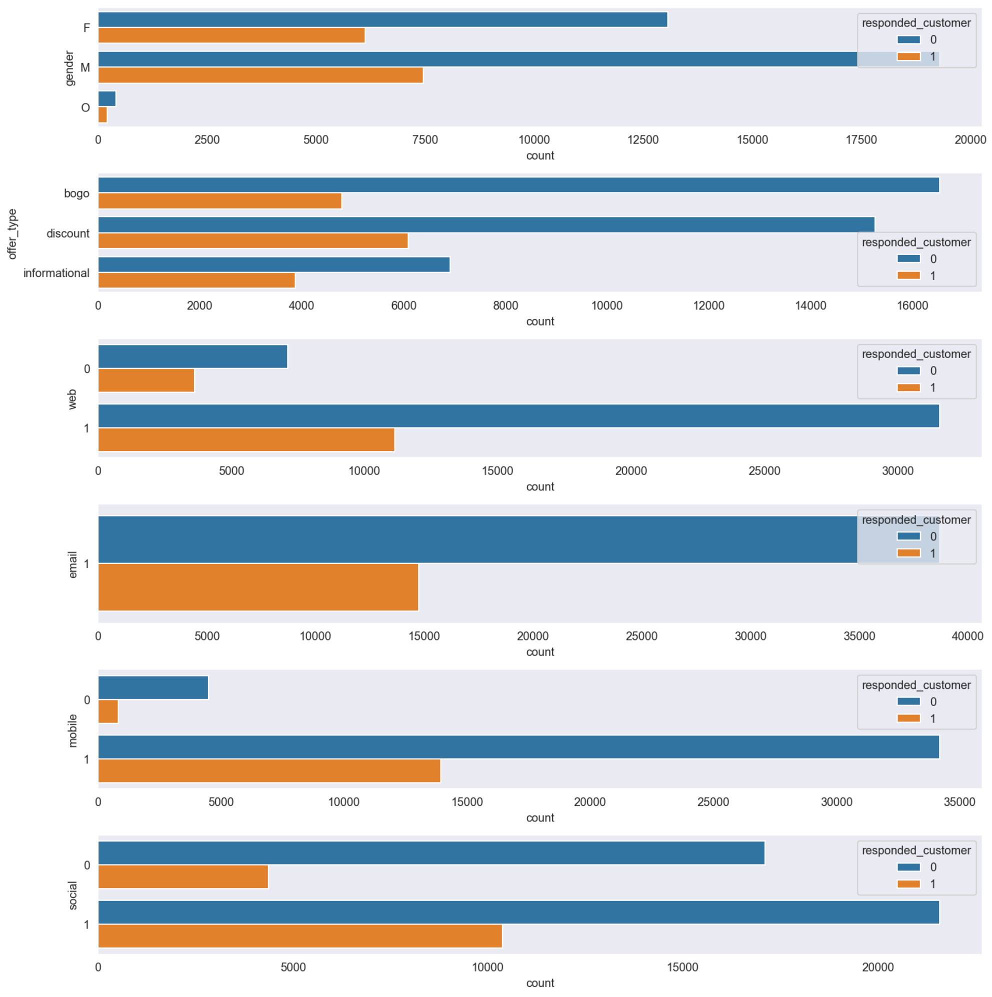

Figure 2.1.7. Bivariate count plots for categorical variables

That is why it is important to split the dataset accordingly – avoiding same pair user-offer appear in evaluation and training, which could lead to data leakage. By doing so and comparing metrics between training and validation sets, we are able to take overfitting into account.

Once the best hyperparameters are then determined accordingly, we can retrain our final model using the whole training set – i.e., including the dataset used for validation. The final step is to estimate final performance on new, previously unseen data in a held-out testing set.

2.3. Benchmark

According to the records obtained during the campaign, the conversion rate – that is, users who actually made a transaction upon an offer receival – is roughly equal to **27.65%**. Hence, our main goal here is to build a model that is capable of predicting the probability that a given customer will respond to an offer, so that It can help us lift the conversion rate by focusing on target groups.

Additionally, we also have conversion rates segregated by campaigns:

2.3.1. Benchmark by promotional Offer

Promotional Offer	BMK Conversion Rates
0b1e1539f2cc45b7b9fa7c272da2e1d7	14.86%
9b98b8c7a33c4b65b9aebfe6a799e6d9	18.96%
2906b810c7d4411798c6938adc9daaa5	18.94%
4d5c57ea9a6940dd891ad53e9dbe8da0	21.09%
ae264e3637204a6fb9bb56bc8210ddfd	22.03%
3f207df678b143eea3cee63160fa8bed	27.89%
f19421c1d4aa40978ebb69ca19b0e20d	27.20%
2298d6c36e964ae4a3e7e9706d1fb8c2	38.65%
5a8bc65990b245e5a138643cd4eb9837	44.07%
fafdcd668e3743c1bb461111dcafc2a4	42.40%

Our real business impact is measured against these values, since it shows how better we perform when targeting our customers.

3. Methodology

3.1. Model Pipeline

After our dataset had been split into training, validation and testing sets and we had properly studied the features and their relationship to the target value, we proceeded to the training loop itself.

First of all, we removed *duration* feature since we chose to use *difficulty* due to the fact that both of them are highly correlated to each other. Then, we divided our preprocessing step according to variable type:

3.1.1. Preprocessing Steps according to Feature Data Type

Variables	Benchmark Conversion Rate
age	1. Imputation of missing values (either using mean or median) 2. Standard scaling
membership_duration	
reward	
income	
difficulty	
frequency	
average_purchase	
offer_type	1. Imputation of missing values (either using mode or constant value) 2. One-Hot Encoding
gender	
web	1. Imputation of missing values (either using mode or constant value)
email	
mobile	
social	

Revisiting our problem of an imbalanced dataset, we had to address this problem during model development. Strategies were:

- **Oversampling:** SMOTE (Synthetic Minority Oversampling TEchnique) to create synthetic examples of the positive class;
- **Weighting**: different weights according to class in loss function.

Using both simultaneously can lead to better results than relying solely on a single of them. Therefore, before proceeding to the model itself – **XGBoost** – we applied SMOTE to increase the number of positive examples.

Then we trained the classification model in two different moments:

- with the training set in order to evaluate performance on the validation set and then adjust hyperparameters;
- with the full training and validation sets in order to estimate final performance on the testing set by using the best hyperparameters.

It is important to emphasize that SMOTE was always only applied to the fitting of the algorithm. For the sake of evaluation, the true proportion of classes was always maintained according to reality.

3.2. Refinement

Since we aim to obtain the best performing model, it is very important that we adjust the hyperparameters accordingly. Therefore, we performed a hyperparameter tuning job so we could find the best ones.

In this context, we chose the following hyperparameters:

- **Resampling ratio**: Since we perform oversampling with SMOTE, this is a hyperparameter that we can adjust to increase training performance;
- **Numerical imputation method**: This is a categorical value that need to be selected to impute missing numerical data: *mean* x *median*;
- **Categorical imputation method**: This is a categorical value that need to be selected to impute missing categorical data: *most_frequent* x *N/A*.
- **XGBoost number of estimators**: Number of estimators used by the algorithm;
- **XGBoost trees' maximum depth**: Maximum depth of each tree estimator used by the algorithm;
- **XGBoost gamma**: Minimum loss reduction required to make a further partition on a leaf node of the tree;
- **XGBoost learning rate**: Learning rate used to adjust estimators' outputs.

Since we want to find the best balance between recall and precision, we aimed to **maximize F1-score**, which is the harmonic mean of these two metrics.

Hence, by following a Bayesian approach to our hyperparameter tuning stage, we were able to reach the following results:

3.2.1. Benchmark by promotional Offer

Promotional Offer	Benchmark Conversion Rate
Resampling ratio	0.8014
Numerical imputation method	Mean
Categorical imputation method	Most Frequent
XGBoost number of estimators	200
XGBoost trees' maximum depth	10
XGBoost gamma	0.2611
XGBoost learning rate	0.2810

This combination of hyperparameters resulted in an F1-Score of 0.8918.

4. Methodology

4.1. Model Evaluation and Validation

As mentioned previously, the final model was trained in the whole training and validation sets with the best hyperparameters. This model had then its performance estimated against the testing set.

First of all, specific classification metrics were calculated by applying bootstrapping within the testing set, since it provides us with an appropriate confidence interval:

4.1.1. Classification Metrics on Testing Set

ACCURACY	F1-SCORE	PRECISION	RECALL	ROC-AUC	COHEN-KAPPA
0.8537±0.0013	0.7337±0.0023	0.7325±0.0025	0.7349±0.0034	0.8942±0.0015	0.6328±0.0031

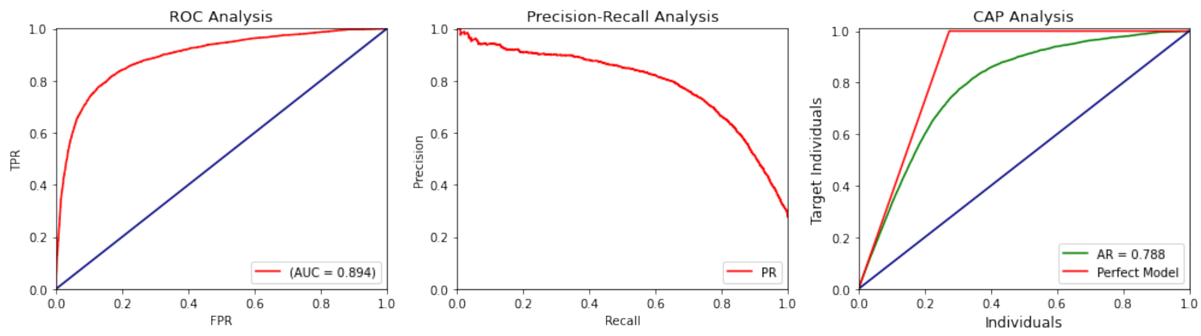

Figure 4.1.1. Classification Evaluation Curves

Since we aim to predict which customer will responde to a given promotional offer, our definition of success from the business point of view is **conversion rate**. Therefore, the **lift** of conversion rates when using our model will indicate how successful the project development is so far.

This analysis was carried out not only globally but also for each individual offer.

4.1.2. Conversion Rates for Promotional Offers

Promotional Offer	BMK	Model	Lift	
0b1e1539f2cc45b7b9fa7c272da2e1d7	14.86%	75.13%	**5.06**	●
9b98b8c7a33c4b65b9aebfe6a799e6d9	18.96%	75.62%	**3.99**	●
2906b810c7d4411798c6938adc9daaa5	18.94%	74.78%	**3.95**	●
4d5c57ea9a6940dd891ad53e9dbe8da0	21.09%	73.08%	**3.46**	●
ae264e3637204a6fb9bb56bc8210ddfd	22.03%	71.76%	**3.26**	●
3f207df678b143eea3cee63160fa8bed	27.89%	75.82%	**2.72**	●
f19421c1d4aa40978ebb69ca19b0e20d	27.20%	69.17%	**2.54**	●
2298d6c36e964ae4a3e7e9706d1fb8c2	38.65%	74.41%	**1.93**	●
5a8bc65990b245e5a138643cd4eb9837	44.07%	75.10%	**1.70**	●
fafdcd668e3743c1bb461111dcafc2a4	42.40%	70.63%	**1.67**	●
GLOBAL	**27.40%**	**73.34%**	**2.68**	●

4.2. Conclusion

From the results presented above, it can be concluded that **the model has intrisic business value**, since it helps us conduct our promotional campaigns in a more targeted way. Not only we are capable of achieving higher conversion rates overall, but we also verify an increase in conversion for every single offer - and the minimum lift value is 1.67 while the maximum is 5.06.

Therefore, by applying the model in order to find out to whom a given promotional offer should be sent we are able to leverage our marketing campaigns and thus achieve higher revenues while incurring less costs with marketing.

5. Productionizing Model

The solution was designed using AWS environment, since it allows us to send the model to production while managing security, access, latency and throughput accordingly. The final design is displayed below:

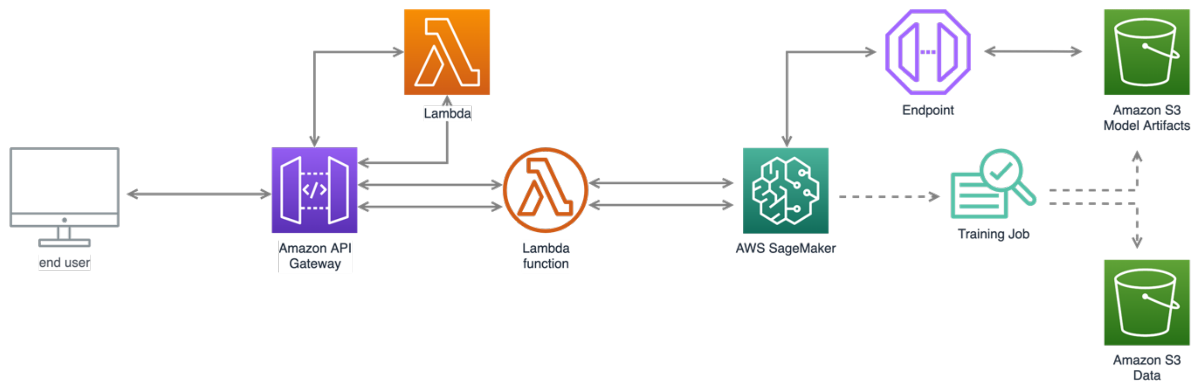

Figure 5.1. Solution Diagram

We expect that third-party users will interact with our classification model. Therefore, we deployed the model within a REST API framework, so that it can act as an intermediary between users and our model by taking inputs from users and pass those inputs to the deployed endpoint. Then, they get outputs from the deployed endpoint and pass them on to users.

In order to request prediction via API, we can execute the following command:

```
curl -X 'POST' https://0wl6qnnrg9.execute-api.us-east-1.amazonaws.com/default/capstone-project-starbucks \
  -H "x-api-key: ${API_KEY}" \
  -d '{
  "data": {
        "person": "0ea1f0bb1ba245abbf709b962d326dd7",
        "offer": "2906b810c7d4411798c6938adc9daaa5",
        "time_in_days_received": 14.0,
        "responded_customer": 0,
        "gender": "M",
        "age":49.0,
        "income": 89000.0,
        "membership_duration": 818,
        "reward": 2,
        "difficulty": 10,
        "duration": 7,
        "offer_type": "discount",
        "web": 1,
        "email": 1,
        "mobile": 1,
        "social": 0,
        "average_purchase": 30.43,
        "frequency": 1.0
    }
}'
```