SENDING PROMOTIONAL OFFERS MORE EFFICIENTLY @ STARBUCKS

Author: Gabriel Fernandes Luz (@gfluz94)
Date: December 14th, 2021

Technical Documentation

1. Domain Background

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

2. Problem Statement

Given a promotional offer and its attributes as well as users and their 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.

3. Datasets and Inputs

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:

1. profile.json

Rewards program users (17000 users x 5 fields)

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

2. portfolio.json

Offers sent during 30-day test period (10 offers x 6 fields)

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

3. **transcript.json**

Event log (306648 events x 4 fields)

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

4. Solution Statement

First, data will be carefully analyzed so the business understanding can be fine-grained, and insights can be generated. Then, we will proceed to feature engineering, where useful features will be created from variables. In order to do so, it will be necessary to concatenate and join the datasets properly.

Then, the problem will be developed according to a classification framework – that is, features and labels will be clearly defined.

Since the problem is complex by design, we will use more advanced algorithms – such as, **XGBoost**. The only care we need to take here is avoid overfitting, since such models can adjust too well to training data. 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.

Finally, an algorithm will be tuned in order to find the best possible performance. The final evaluation will be held on a dataset which was not previously used during model design.

Finally, this model will be made available through an API, so end users are able to perform requests to an endpoint to get predictions – probabilities to a given scenario. In order to accomplish it, training and deployment will be carried out in AWS SageMaker with integration with AWS Lambda.

5. Benchmark Model

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:

- **0b1e1539f2cc45b7b9fa7c272da2e1d7**: 14.87%
- **2298d6c36e964ae4a3e7e9706d1fb8c2**: 36.80%
- **2906b810c7d4411798c6938adc9daaa5**: 20.51%
- **3f207df678b143eea3cee63160fa8bed**: 27.53%
- **4d5c57ea9a6940dd891ad53e9dbe8da0**: 21.27%
- **5a8bc65990b245e5a138643cd4eb9837**: 44.36%
- **9b98b8c7a33c4b65b9aebfe6a799e6d9**: 19.67%
- **ae264e3637204a6fb9bb56bc8210ddfd**: 23.11%
- **f19421c1d4aa40978ebb69ca19b0e20d**: 27.12%
- **fafdcd668e3743c1bb461111dcafc2a4**: 41.44%

6. Evaluation Metrics

Technically, the success of the project is measured against classical classification performance metrics, such as recall, precision, accuracy and ROC AUC. These metrics will certainly be translated into scenarios in the business context, helping decision makers leverage the business.

Additionally, lift is 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.

7. Project Design

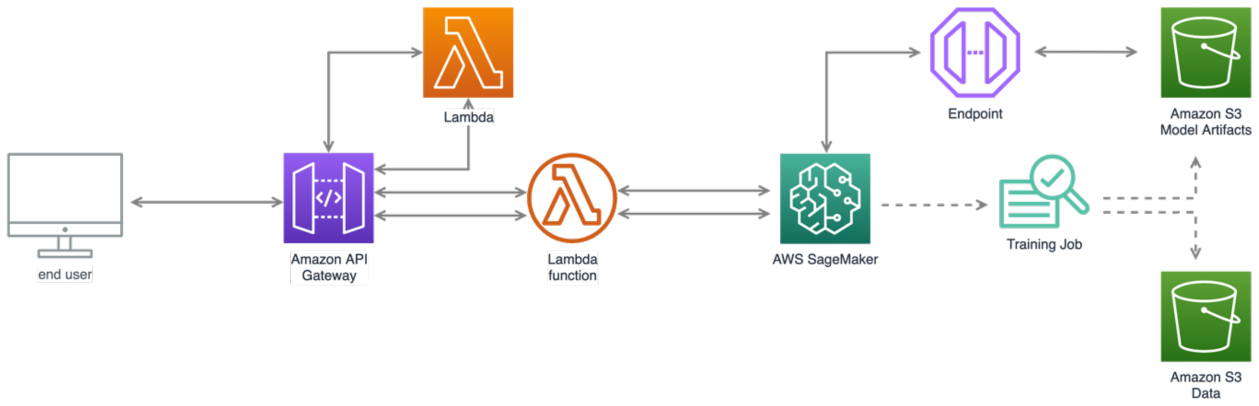
In the model development stage, the train-test split must be carried out very carefully, since we need to avoid data leakage at all costs. In order to do so, we will hash the concatenation of *person_id* and *offer_id*, and based on this new field perform the split of the datasets. By doing so we guarantee that the same user with the same offer won't both appear in training and testing datasets.

Then a classification model will be designed and its hyperparameters will be tuned according to a validation set. Final performance, business metrics and back testing will then be carried out within the test set, because it will resemble the performance for new, previously unseen data.

Once experimentation and validation are over, the training will be performed in AWS SageMaker, so we can save model artifacts accordingly and then deploy an endpoint later on. Finally, a Lambda Function will be developed in order to accept API requests and consequently invoke the endpoints, returning hence predictions to the end user.

In summary, the solution will be designed using AWS environment, since it will allow us to send the model to production while managing security, access, latency and throughput accordingly.

AWS Solution Scheme


Lambda
Endpoint
Amazon S3
Model Artifacts
end user
Amazon API
Gateway
Lambda
function
AWS SageMaker
Training Job
Amazon S3
Data