Starbucks Capstone Project Proposal

Patricio Villanueva
Udacity ML Engineer Nanodegree | March 2021

Introduction

This Starbucks Capstone project is part of the Udacity Machine Learning Engineer Nanodegree. Udacity partnered with Starbucks to provide a real-world business problem and simulated data mimicking their customer behaviour.

Domain Background

Starbucks is a passionate purveyor of coffee and other beverages, headquartered in Seattle, Washington. They have a mobile application where registered users can use it to order coffee for the pickup while mobile, pay in-store directly using the app, and collect rewards points. This app also offers promotions for bonus points to these users. The promotional offer can be merely an advertisement for a drink or an actual offer such as a discount or BOGO (buy one get one free). This project is focused on tailoring the promotional offers for customers based on their responses to the previous offers and find out which of them are most likely to respond to an offer.

Problem Statement

We want to determine which kind of offer, if any, to send to each customer based on their purchases and interaction with the previously sent offers.
Not all users receive the same offer, and that is the challenge to solve using the data set that is provided by Starbucks, which was captured over 30 days. I'll also build a machine learning model that will predict the response of a customer to an offer.

Datasets and Inputs

For this project, we will be using the dataset provided by Starbucks. The data consists of 3 files containing simulated data that mimics customer behaviour on the Starbucks Rewards mobile app.

- ❏ Portfolio.json contains info about the offers
- ❏ Profile.json contains info about the customers
- ❏ Transcript.json contains info about customer purchases and interaction with the offers.

We have information about 17,000 customers and a transcript containing 306,534 purchases and offer interactions.
A customer can interact with an offer by receiving it, viewing it, or completing it. It is possible for a customer to complete some offers without viewing them.
To split the customer data into training/testing sets I will use a 70/30 split for the customers.
To determine this I looked at the value counts for all events listed in transcript.json.

- `transaction 138953`
- `offer received 76277`
- `offer viewed 57725`
- `offer completed 33579`

We are primarily concerned with whether a customer completes the offers they have received. To determine what percentage completed their offers, I performed the following calculation:

(76,277 − 33,579) / 76,277 = 42,698 / 76,277 = 55.97%

That means that 55.79% of the people did not complete their offers, while 44.03% received offers and did complete them. These percentages are close enough to consider this a balanced dataset.

Below is the schema and explanation of each variable in the files: :

1) portfolio.json

This contains information about the offers sent over a 30-day trial period, each having a unique id.
(10 offers x 6 features defined below)

- reward (int) – the reward is given in dollars for completing an offer.
- channels (list of strings) – platform the offer was sent on. Combination of email, mobile, social, and web.
- difficulty (int) – minimum dollar amount customer must spend to complete an offer.
- duration (int) – time in days that the offer is valid.
- offer_type (string) – a type of offer. Either BOGO, discount, or informational.
- id (string) – unique hash identifying the specific offer.

2) profile.json

This contains customer information assigning each a unique id.
(17,000 customers x 5 features defined below)

- gender (string) – ‘M’, ‘F’, or ‘O’ for others. If none provided, the default value is null.
- age (int) – age of the customer. If none provided, the default value is 118 (the birth year is 1900).

- id (string) – unique hash identifying the specific customer.
- became_member_on (int) – date customer became a member. YYYYMMDD format.
- income (float) – customer's income. If none provided, the default is null.

3) transcript.json

This contains each customer's purchases and whether they received, viewed, or completed an offer.

- (306,534 events x 4 features defined below)
- person (string) – unique customer id.
- event (string) – Type of event that occurred. Either "transaction", "offer received", "offer viewed", or "offer completed".
- value (dictionary of strings) – either an offer id or transaction amount, depending on the event.
- time (int) – time in hours since the start of the test. The range is between t=0 and t=714

The portfolio.json contains offer_type column, which describes the types of offers that Starbucks is looking to potentially send its customers:

1. BOGO (Buy-One-Get-One): This offer enables a customer to receive an extra and equal product at no additional cost. The customer must spend a certain threshold in order to make this reward available.
2. Informational: This offer doesn't necessarily include a reward, but rather an opportunity for a customer to purchase a certain object given a requisite amount of money.
3. Discount: With this offer, a customer is given a reward that knocks a certain percentage off the original cost of the product they're choosing to purchase, subject to limitations.

Solution Statement

To solve this, I will use supervised learning models to determine the propensity for a customer to complete an offer.
I plan to implement Logistic Regression, Random Forest and Artificial Neural Network algorithms for my models.
Logistic Regression is a common technique used for binary classification problems, in this case however we want to classify our customers into four possible classes to see which promotion is valid. For this reason, this model will be considered as a base model and compare it with the others.
Random Forest attempt to find the best sets of rules to classified the customers, and defined which promotion should we send them if any.
Neural Networks are universal function approximators. This means they can approximate any smooth polynomial function, allowing them to better find the boundaries in high-dimensional data. I plan to use a neural network with 4 layers. The first layer is the input layer and will contain nodes for each feature. The second and third layer are the hidden

layers and they will test different numbers of hidden nodes. The final layer is the output layer and will consist of four nodes using a softmax activation function. This will output a value between 0 and 1, where 1 if the customer is likely to use the offer, and 0 otherwise. I will determine during my testing where the threshold should be on determining whether to send the offer or not.

Benchmark Model

For a benchmark model, I will first build a logistic regression model and compare all other models against that. This should give me a solid benchmark to compare against since logistic regression is quite efficient in dealing with the majority of business scenarios

Evaluation Metrics

The evaluation metric for this problem will be accuracy since we want to be as accurate as possible on what promotion is useful to each customer.
We will be also taking into consideration how the F1 and recall metrics to see how bad it performs with the mispredicted data.

Project Design

Below is my high-level workflow:

1) Perform a high-level exploration of the data.

    a. Understanding the data, missing values, minimum & maximum values, etc.

2) Clean the data. Below are the current items I see that need cleaned.

    a. Renaming columns, drop useless columns, correcting data formats and merging the 3 dataframes into one

3) Data Exploration and Visualization:

    a. Visualization of the data such as age, income, gender, offers received, viewed and completed among others.

4) Modeling:

    a. Imputation, One Hot Encoding and Scaling the data.

    b. Build other models using different supervised-learning algorithms (Logistic Regression, Random Forest, Neural Networks).

    c. Run the model against the test set.

    d. Use the evaluation metrics on each model and compare them with each other.

5) Compile results into a report and blog post.

6) Upload files to GitHub with README.md file.