Udacity Machine Learning Engineer - Capstone Proposal

Project: Optimising discounting on Starbucks products based on loyalty reward app data

Matthew Gela

Domain background

Starbucks is an international coffee shop chain, with over 30,000 stores worldwide. As with all customer facing companies, it is in their interest to understand customer behaviour, and how to influence it in such a way as to maximise the value they can get from a customer (usually quantified by the amount of revenue they can make from a customer, but also in referrals, etc.).

Starbucks has had a long running loyalty rewards programme, digitised in the last few years in the form of the Starbucks Rewards app, which has been highly successful with about half of the smartphone owners who regularly use restaurant loyalty apps using the Starbucks app **[1]**.

Starbucks are interested in understanding behaviour in response to promotional offers shown to customers in order to run their sales campaign. Discounting an item without sufficient uptake in demand for that item leads to a loss in revenue, so it is in their interest to ensure that they can set the discount at a level where it would lead to an overall increase in revenue.

Starbucks have collected large amounts of data about their customers, and how they have behaved in the past in response to promotional offers on various items. This can be mined and used in a machine learning context to understand more about their customers, and to predict how they may behave in response to promotional offers on a particular item.

In particular, Starbucks could be able to use this data to provide personalised offers **[2]** to customers such that they are most likely to complete a transaction in response to receiving that offer, thus driving increased engagement with the app and ultimately increased revenue.

Problem statement

The problem is to identify which promotion type would be the most effective to offer to a Starbucks app user such that it will incentivise them to make a purchase. Additionally, we will want to take into account the following:

- some users may not want any offers at all (may act adversely to receiving offers), and so for these users

- Some users will be very active in making transactions even without being prompted by offers, and so for these customers we may not want to provide offers

<u>The dataset and inputs</u>

Data provided by Udacity.

This data set contains simulated data that mimics customer behavior on the Starbucks rewards mobile app. Once every few days, Starbucks sends out an offer to users of the mobile app, such as an advertisement for a drink or an actual offer such as a discount or BOGO. Not all users receive the same offer.

There are three datasets provided: transaction, customer profile, and offer portfolio:

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

b) **Customer profile dataset**

    Demographic data on 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

c) **Offer portfolio dataset**

    Contains offer ids and information about each offer - duration, type, etc.

    - id (string) - offer id
    - offer_type (string) - type of offer ie BOGO, discount, informational

- difficulty (int) - minimum required spend to complete an offer
- reward (int) - reward given for completing an offer
- duration (int) - time for offer to be open, in days
- channels (list of strings)

These datasets will need to be joined to create a single dataset such that each line represents a unique customer-promotion combination. We can perform some EDA and pre-processing of the data to understand and prepare it for modelling.

Solution statement

Building a recommender system will be the approach I will take to solve the project problem stated above. The recommender system will be able to score how a customer will react to each promotion, and thus suggest the best promotion to offer to the customer.

Recommender systems can be built using various approaches, including collaborative, content-based, demographic and knowledge-based filtering. There are a number of well known issues with only employing one of these filtering techniques, such as the cold-start problem (i.e. where the system has no information about new users or promotions), or the data sparsity problem (i.e. users do not get offered many promotions), and so using any of these systems in isolation can struggle to generate good recommendations in these situations.

In order to overcome these issues, a combination of these methods are more frequently used to create a hybrid recommender system **[3]**. Given the data provided on transactions, customers, and offers, it is likely some combination of collaborative, content-based and demographic filtering **[4]** would make the best use of the information we have, but this will be explored during the capstone project.

Benchmark model

I will benchmark the hybrid recommender systems against pure collaborative filtering and content-based filtering approaches.

Evaluation metrics

Commonly used evaluation metrics for recommender systems **[5]** are:

1. Mean Average Precision - this tells us how relevant the list of recommended items are for a customer, i.e. what proportion of the recommended offers did the customer interact with to complete a transaction

2. Mean Average Recall - this tells us how well the recommender is able to recommend the offers where the customer has actually interacted with to complete a transaction

Note that we should only take into account recommended promotions that the customer has interacted with when calculating these metrics.

It is likely we will optimise for one of these, while setting a minimum threshold for the other, or we could optimise for a harmonic mean of the two metrics (f1).

Other interesting metrics we could use as satisficing metrics are:

- Coverage: this will tell us the percentage of items that the model is able to recommend on the test set; ideally, we would like a good spread of recommended items
- Personalisation: using a similarity metric (e.g. cosine similarity) could show us whether users are mainly being given the same set of recommended offers, or whether they are actually being tailored to the individual

Project design outline

The project will take the following steps:

Data Engineering

1. Data exploration and validation
    - understand the structure and schema of the three datasets (transaction, customer and offer), and how they can be linked together
    - Explore each dataset individually to understand the feature distributions, correlations, etc.
    - Identify missing data input and choose strategy for dealing with data (imputation, deletion, etc.)
2. Data cleaning and preprocessing
    - Missing value imputation
    - Data type conversions

ML Model Engineering

3. Feature engineering
    - Extracting feature variables from columns of the data

Note that the dataset preparation and feature engineering may vary somewhat by recommender method used; i.e. collaborative filtering models require only the user/promotion matrix; content-based filtering requires features related to properties of the offer promotions; demographic-based filters require features related to information about the users.

4. Split into training and test data

- Splitting the data into the dataset we will use to train the recommender, and then the dataset we will use to evaluate the different recommender models against performance metrics

5. Model training
    - Train the various recommender models
6. Model evaluation & hyperparameter tuning
    - I will evaluate the performance of the hybrid recommender models built in order to choose the one with optimal performance, and then compare the performance of this against the benchmark models

ML Model Deployment

7. Model serving
    - I will use Amazon SageMaker to deploy my recommender system, using the steps taught in the course

References

[1] How Customers Use Food Delivery and Restaurant Loyalty Apps
https://themanifest.com/mobile-apps/how-customers-use-food-delivery-restaurant-loyalty-apps

[2] Mutanen, Teemu & Nousiainen, Sami & Liang, He. (2010). Personalized Recommendation on Discount Coupons. 10.5176/978-981-08-7466-7_kd-18.

[3] Çano, E. and Morisio, M., 2017. Hybrid recommender systems: A systematic literature review. *Intelligent Data Analysis*, *21*(6), pp.1487-1524.

[4] Christensen, Ingrid & Schiaffino, Silvia. (2014). A Hybrid Approach for Group Profiling in Recommender Systems. JOURNAL OF UNIVERSAL COMPUTER SCIENCE. 20. 507-533.

[5] Popular evaluation metrics in recommender systems explained
https://medium.com/qloo/popular-evaluation-metrics-in-recommender-systems-explained-324ff2fb427d