Capstone proposal

Domain Background

Optimizing App Offers With Starbucks: Starbucks has previously employed artificial intelligence to expand its business successfully. Some of the examples are :

- Personalizing the experience by recommending orders based on their previous purchases, current weather and if it's a weekday or a holiday.
- Targeted marketing by sending personalized discounts to re-engage customers.
- Determining new store locations based on traffic, demographics and proximity to other starbucks locations.
- Expansion of products into grocery stores by analyzing the at-home consumption of products by customers.

The above examples show how intelligent data analytics can help understand the customer behavior better, subsequently helping the company to re-invent its marketing and product expansion processes.

The proposed project can help Starbucks with the following:

- Optimize marketing spend by sending targeted offers
- Improve conversion rates of marketing/offer campaigns
- Improve customer retention and customer lifetime value (CLV) by sending relevant offer types

Problem Statement

The goal of this project is to:

> Given a customer ID and an offer ID, predict if that combination is likely to result in an *offer completion* using customer's historical data with Starbucks. Since informationals do not have offer completion as such, it is not part of this analysis. Our focus is on completion of BOGO and discount offers.

Datasets and Inputs

The demographic, offer and transaction data is stored in following three json files:

profile.json

Rewards program users (17000 users x 5 fields)

Original features:

- 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)

Derived features:

- days since member: numeric
- year since member: numeric

Missing values:

- Gender: 11.2%
- Age: 11.2%
- Income: 11.2%

Notes:

- Age and income are correlated

portfolio.json

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

Original features:

- 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)

Derived features: Binary flags for individual channels.

Notes:

- Reward is correlated with difficulty and duration.
- Difficulty has negative correlation with social.
- Informationals have zero reward and difficulty.

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

Data excluded for our problem statement:

- Transactions and informationals

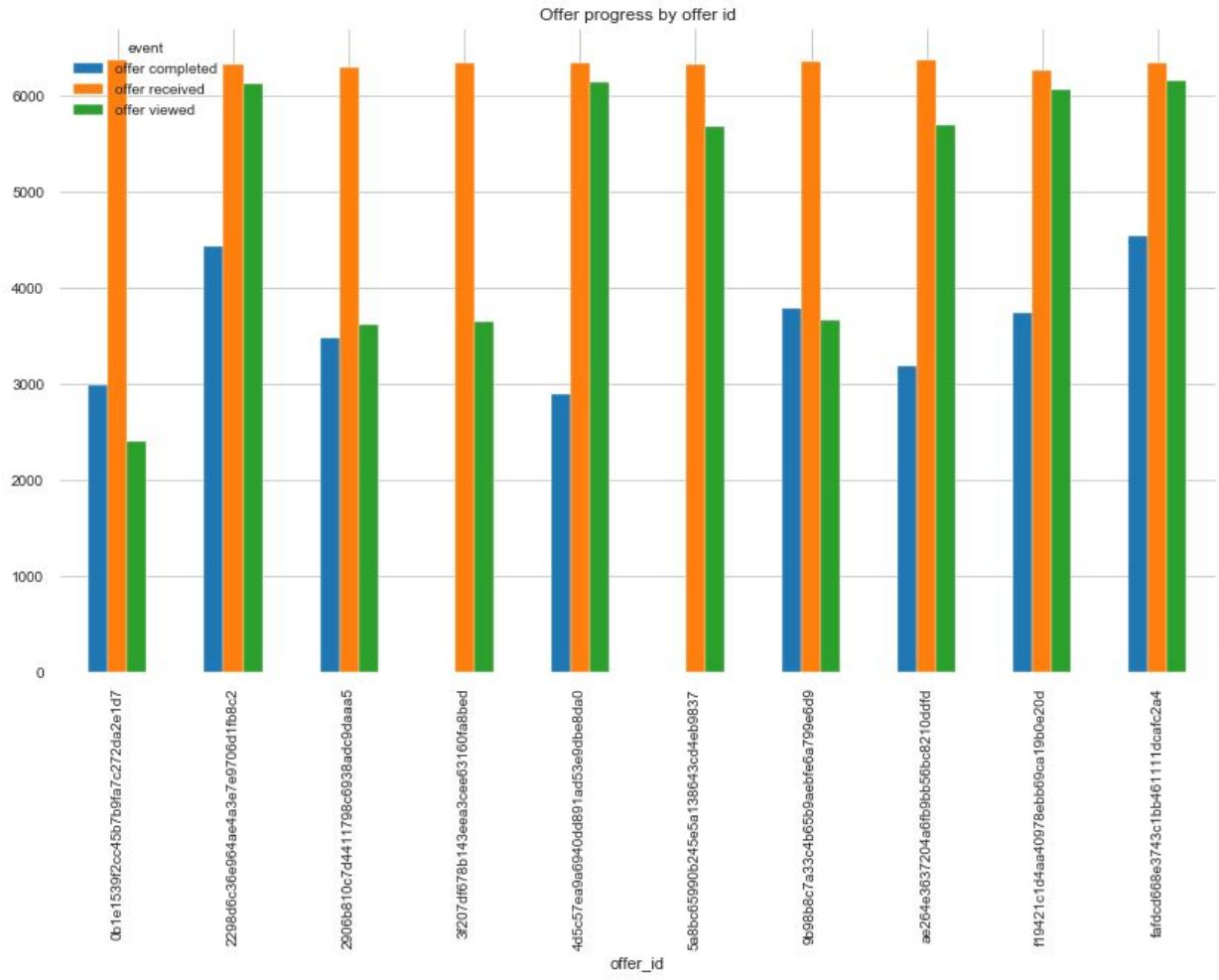
Prepared dataset

Steps:

- Size of data after various steps of cleaning:
    - Initial observations: 167581
    - After removing age=118: 148805
    - After removing informationals: 126145
- Event counts:
    - Original labels:
        - offer received 53201
        - offer viewed 40500
        - offer completed 32444
    - New labels:
        - not completed 93701
        - offer completed 32444

One hot encoded features: Gender, offer ID, offer type

Number of rows: 126145
Count of events by offer IDs:



Solution Statement

- General data preprocessing:
    - Cleaning and exploring data: Removing null values
    - Merging the datasets
- We can predict the probability of offer-completion for an individual by building a binary classifier. This classifier can be further used to find the most relevant offers for a given customer.

  The next steps for building a classifier are:
    - Splitting data into train and test sets
    - Training a classifier and tuning it by changing hyperparameters
    - Measuring the model against evaluation metrics

Benchmark Model

Our benchmark is a naive understanding of customers without an ML model. If we were not to use customer history to target them: how would we usually do it? -- Target them all. This translates to a rule based model which always predicts an offer to be completed. The area under the curve of the ROC curve of this approach is 0.5. Any improvement over this should bring some business value. Our objective is to optimise the AUC as much as possible.

Evaluation Metrics

We plan to use F-score and area under the curve of ROC. F-score considers both the precision and recall to compute the score. Since it is a binary classifier, both the metrics are easy to interpret and are generally good even in cases of class imbalance.

Project Design

Project Workflow:

- Cleaning and exploring the data: visualizing the data, removing null values, merging the three json files, finding the shape of the final dataset (rows, columns)
- Feature engineering: Aggregating and manipulating existing features to get an overall customer history
- Splitting the data into training and testing sets
- Training the classifier (RandomForest Classifier)
- Measuring the classifier against the evaluation metrics
- Tuning the classifier by learning best hyperparameters
- Analyse improvements over the benchmark
- Summarise findings

References

Classification

- Random forest classifier: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html
- Balanced classifiers to address class imbalance: https://imbalanced-learn.readthedocs.io/en/stable/user_guide.html
- Random search CV for hyperparameter tuning and cross validation: https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.RandomizedSearchCV.html

Analysis and visualisations

- Pandas
- Matplotlib
- seaborn