Starbucks Capstone Project Proposal

1. Introduction This Starbucks Capstone Project is Udacity DS Nanodegree. Udacity partnered with Starbucks to provide a real-world business problem and simulated data mimicking their customer behavior.

2. Domain background Starbucks Corporation is an American coffee company and operates over 30,000 locations worldwide. 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. The company would like to select customers with the highest probability of making a purchase, in contrast, want to know the customers who don't like to receive offers too. In other words, the right offer to the right customer. This project can be handled as a classifical supervising learning problem, so we have our features about the customers, the offers and the transcripts (as we will see in the dataset section) and our label will be if the offer was completed or not (binary classification). Maybe this label column needs to be created after the analysis of all data. My motivation for this project is because here in my city we don't have Starbucks and when I travel I need to go there since I really enjoy their products. So, it would
be nice to work with a dataset's company that I really like and I don't have in my day-to-day.

3. Problem Statement

Building a model that predicts whether or not someone will complete an offer based on given demographics and offer type. Thus there will be two possible outputs from the model, making this problem as a supervising learning problem, more specifically, binary classification problem:

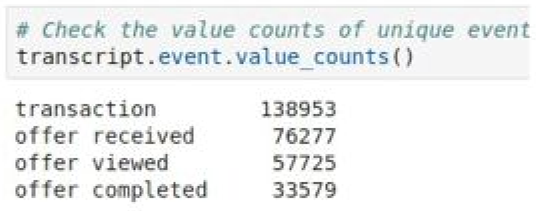
- The customer complete the offer (Yes / 1)
- The customer will not complete the offer (No / 0)

4. Dataset and Inputs

4.1 Dataset overview:

- The program used to create the data simulates how people make purchasing decisions and how those decisions are influenced by promotional offers.
- Each person in the simulation has some hidden traits that influence their purchasing patterns and are associated with their observable traits. People produce various events, including receiving offers, opening offers, and making purchases.
- As a simplification, there are no explicit products to track. Only the amounts of each transaction or offer are recorded.
- There are three types of offers that can be sent: buy-one-get-one (BOGO), discount, and informational. In a BOGO offer, a user needs to spend a certain amount to get a reward equal to that threshold amount. In a discount, a user gains a reward equal to a fraction of the amount spent. In an informational offer, there is no reward, but neither is there a requisite amount that the user is expected to spend. Offers can be delivered via multiple channels.

- The basic task is to use the data to identify which groups of people are most responsive to each type of offer, and how best to present each type of offer.



```
# Check the value counts of unique event
transcript.event.value_counts()
```

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

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. After the preprocessing step with all the exploratory data analysis (EDA), I will combine all the data in an unique data cleaned to use for input to the models. To split into training and testing sets I will use a 70/30 split.

4.2 Data Dictionary:

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)

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)

Transcript.json: Event log (306648 events x 4 fields)

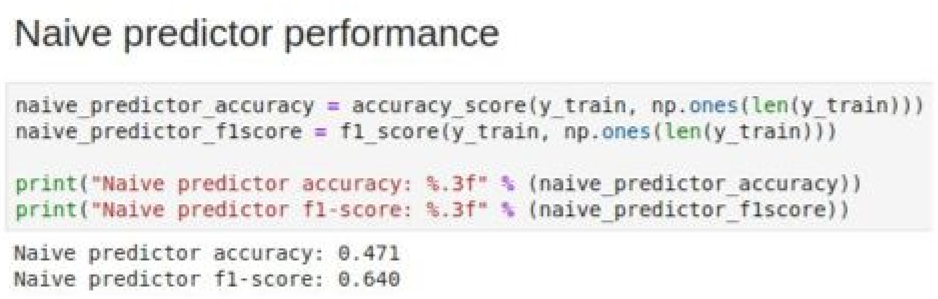
- person: (string/hash)
- event: (string) offer received, offer viewed, transaction, offer completed
- value: (dictionary) different values depending on event type
    - offer id: (string/hash) not associated with any "transaction"
    - amount: (numeric) money spent in "transaction"
    - reward: (numeric)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

Solution Statement

Clean up all the dataset, combine them into a unique data and use it as input of machine learning models. In other words, I will use supervising learning models to determine the propensity for a customer to complete an offer. Specifically, I plan to use the logistic regression and random forest models.

Benchmark Model

First, I will construct a naive predictor model as a dumb basiline for the other models. This naive model will predict positive class for all examples, so we hope that our both models (logistic regression baseline and random forest) have better performance than the dumb model.


Naive predictor performance

```
naive_predictor_accuracy = accuracy_score(y_train, np.ones(len(y_train)))
naive_predictor_f1score = f1_score(y_train, np.ones(len(y_train)))

print("Naive predictor accuracy: %.3f" % (naive_predictor_accuracy))
print("Naive predictor f1-score: %.3f" % (naive_predictor_f1score))
```

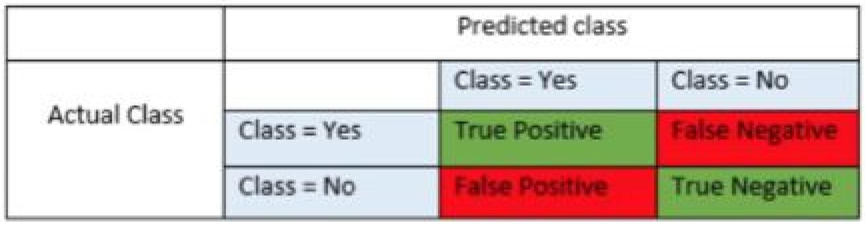
```
Naive predictor accuracy: 0.471
Naive predictor f1-score: 0.640
```

Next, I will create a logistic regression model to use as a solid baseline against the random forest model, since logistic regression is a simple and efficient model to deal with business problems.

Evaluation Metrics

To evaluate the performance of the models I will use the accuracy and f1-score metrics, since in my opinion both recall and precision are important here. Finally, I will compute and plot the confusion matrix and the auc roc curve.

To understand the metrics I will use in this project, we first need to understand the confusion matrix.



		Predicted class	
Actual Class		Class = Yes	Class = No
	Class = Yes	True Positive	False Negative
	Class = No	False Positive	True Negative

In our context:

True Positive (TP): Send an offer and the user will likely use it.
False Positive (FP): Send an offer but the user doesn't want it or doesn't use it.

True Negative (TN): Do not send an offer and the user doesn't want it or doesn't use it.
False Negative (FN): Do not send an offer and the user would have likely used it if we sent it.
Accuracy = TP+TN/TP+FP+FN+TN
Precision = TP/TP+FP
Recall = TP/TP+FN
F1 Score = 2*(Recall * Precision) / (Recall + Precision)

AUC ROC Curve it tells how much a model is capable of distinguishing between classes.

Project Design

General workflow for this project:

1. Create a conda environment on my local computer and install the necessary packages.
2. Migrating the data from Udacity Workspace to my local computer.
3. Initial analysis of the data
    a. Check the shape and the first visualization of some rows
    b. Information about data type of each feature
    c. Count and check unique values
    d. Plots
        i. Histograms
        ii. Distributions
4. Initial cleansing of the data
    a. Rename features
    b. Check and remove null values
    c. Change data types
5. Performance feature engineering to prepare for modeling
    a. One hot encoding
    b. Create new columns
6. Build the models with Randomized Search CV

   a. Naive Predictor Model
   b. Logistic Regression Model
   c. Random Forest Model

7. Evaluate the models
   a. Check accuracy and f1-score metrics
   b. Plot confusion matrix and roc curve
8. Compare the models
   a. Create a dataframe with the values of each evaluate results from each models
9. Upload files to GitHub