title: "Starbuck_Project"
date: 2023-07-18

Section 1: Project Definition

Project Overview

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. 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.

Not all users receive the same offer, and that is the challenge to solve with this data set.

Your task is to combine transaction, demographic and offer data to determine which demographic groups respond best to which offer type. This data set is a simplified version of the real Starbucks app because the underlying simulator only has one product whereas Starbucks actually sells dozens of products.

Every offer has a validity period before the offer 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.

You'll be given transactional data showing user purchases made on the app including the timestamp of purchase and the amount of money spent on a purchase. This transactional data also has a record for each offer that a user receives as well as a record for when a user actually views the offer. There are also records for when a user completes an offer.

Keep in mind as well that someone using the app might make a purchase through the app without having received an offer or seen an offer.

Problem Statement

Not all users receive the same offer, and that is the challenge to solve with this data set.

Metrics

We use ACCURACY as classification metric.

ACCURACY is the the result of: (TRUE_POSITIVE+TRUE_NEGATIVE)/(total cases)

Section 2: Analysis

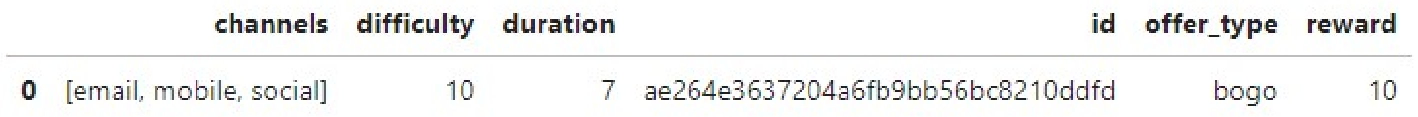
Data Exploration

The data is contained in three files:

* portfolio.json - containing offer ids and meta data about each offer (duration, type, etc.)
* profile.json - demographic data for each customer
* transcript.json - records for transactions, offers received, offers viewed, and offers completed
Here is the schema and explanation of each variable in the files:

* portfolio.json

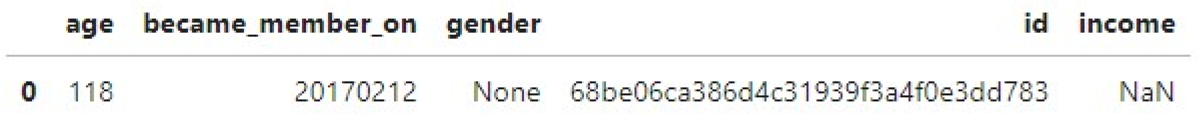
id (string) - offer id
offer_type (string) - type of offer ie BOGO, discount, informational
difficulty (int) - minimum required spend to complete an offer
reward (int) - reward given for completing an offer
duration (int) - time for offer to be open, in days
channels (list of strings)



	channels	difficulty	duration	id	offer_type	reward
0	[email, mobile, social]	10	7	ae264e3637204a6fb9bb56bc8210ddfd	bogo	10

* profile.json

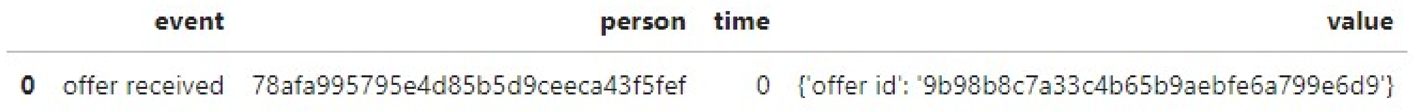
age (int) - age of the customer
became_member_on (int) - date when customer created an app account
gender (str) - gender of the customer (note some entries contain 'O' for other rather than M or F)
id (str) - customer id
income (float) - customer's income



	age	became_member_on	gender	id	income
0	118	20170212	None	68be06ca386d4c31939f3a4f0e3dd783	NaN

* transcript.json

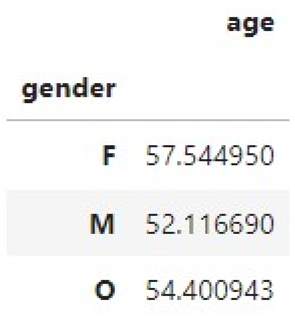
event (str) - record description (ie transaction, offer received, offer viewed, etc.)
person (str) - customer id
time (int) - time in hours since start of test. The data begins at time t=0
value - (dict of strings) - either an offer id or transaction amount depending on the record



	event	person	time	value
0	offer received	78afa995795e4d85b5d9ceeca43f5fef	0	{'offer id': '9b98b8c7a33c4b65b9aebfe6a799e6d9'}

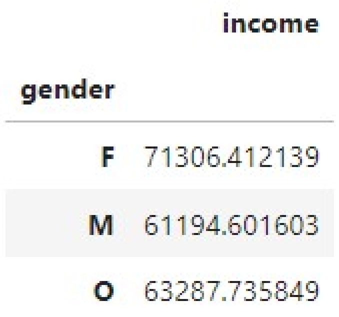
Data Visualization

Age per gender:



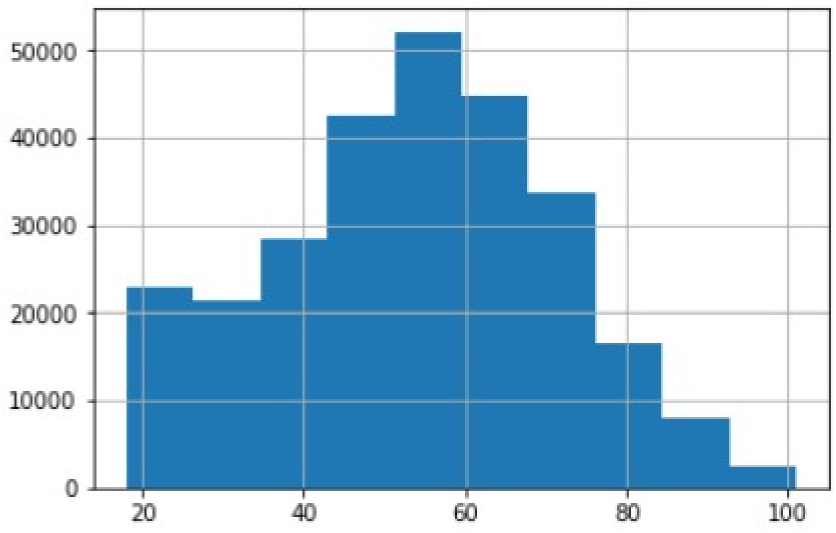
	age
gender	
F	57.544950
M	52.116690
O	54.400943

Income per gender:



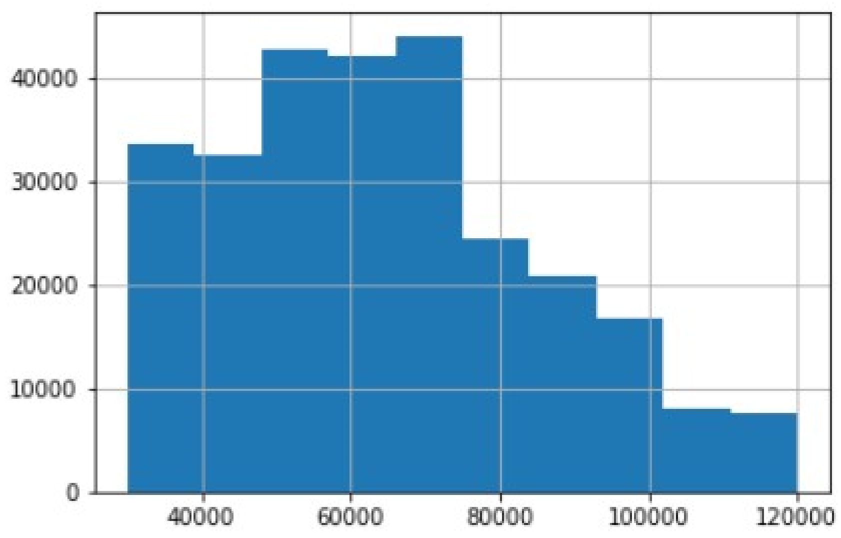
	income
gender	
F	71306.412139
M	61194.601603
O	63287.735849

Average of age:



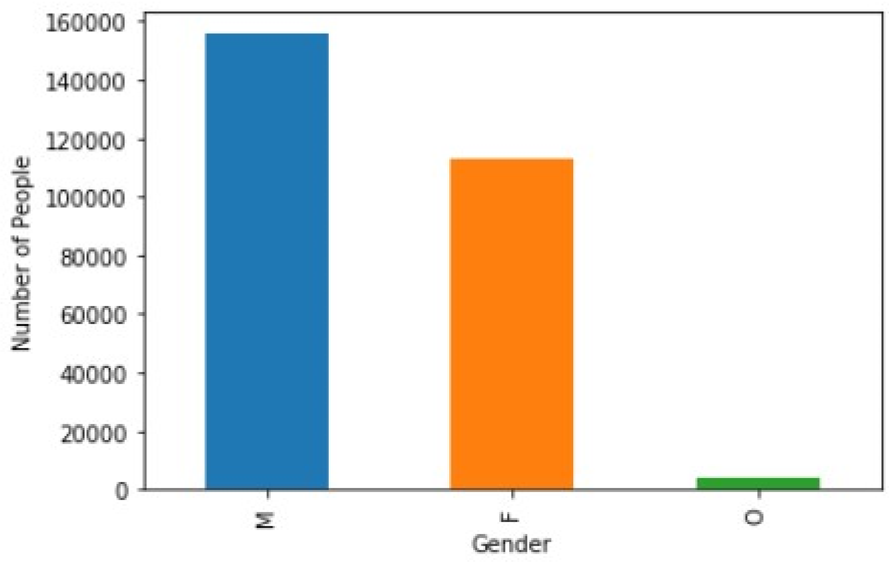
The average age of the users is: 50-60 years

The average of incomes:

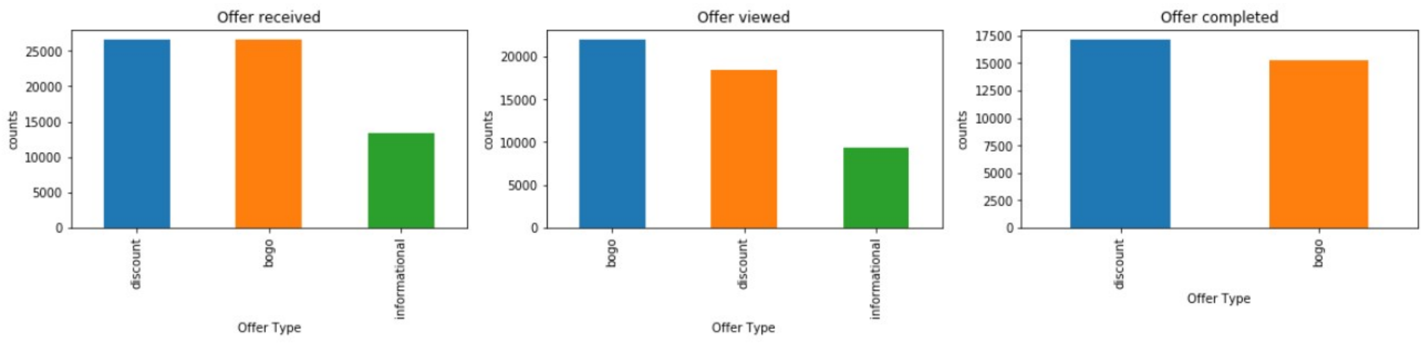


The average of the incomes is: 60000-70000

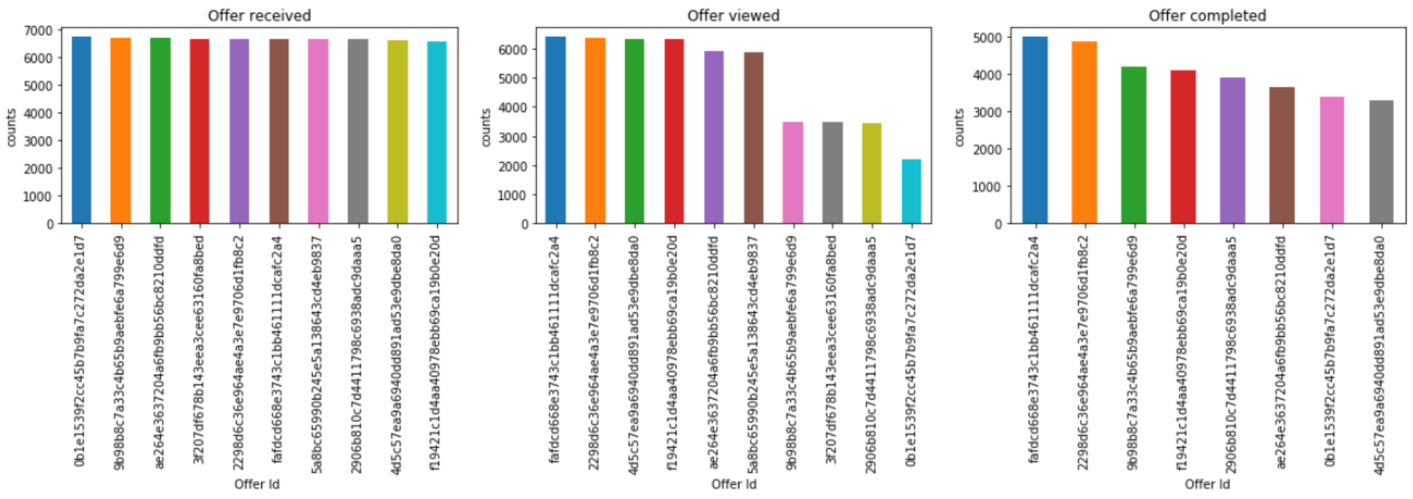
Number of people per genre:



Offer type:



```
offer received

discount         26664

bogo             26537

informational    13300

Name: offer_type, dtype: int64

offer viewed

bogo             22039

discount         18461

informational     9360

Name: offer_type, dtype: int64

offer completed

discount    17186

bogo        15258
```



Section 3: Methodology

Data Preprocessing

1. We use One-hot-encoding. In this case we encoding the gender and offer_type column

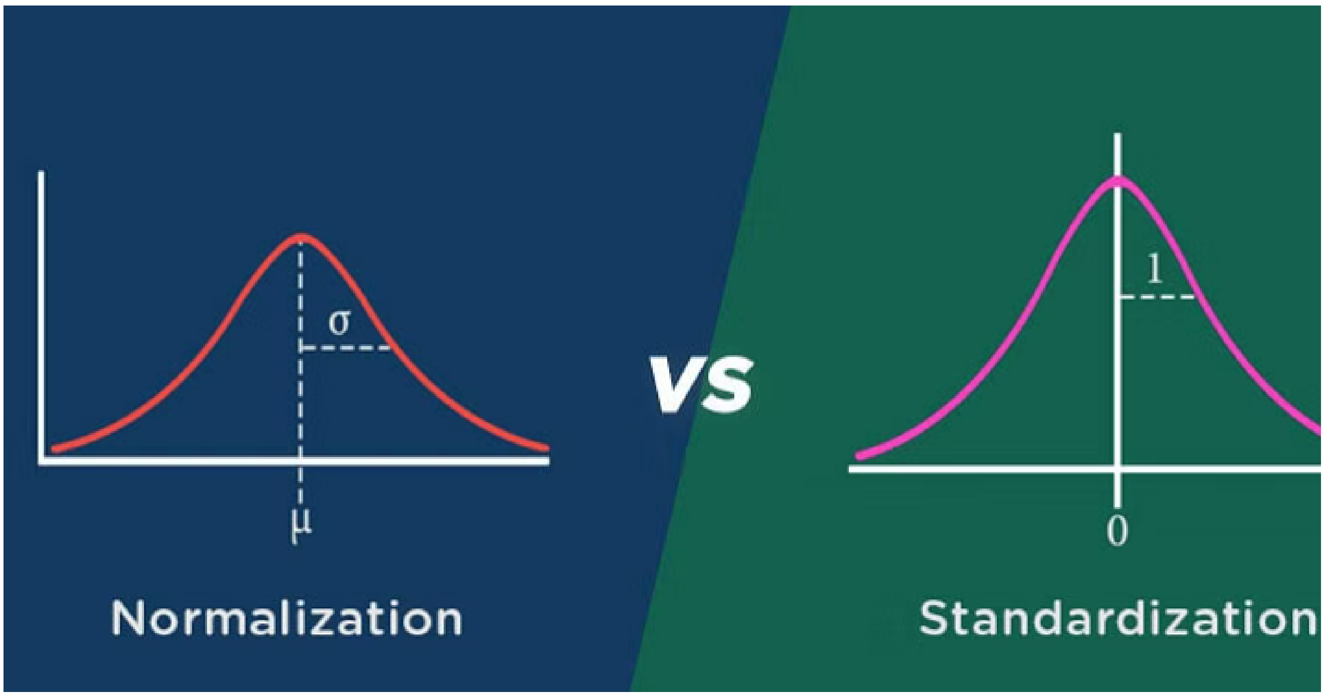
2. We split the dataset using train_test_split function, that we import:
   ```
   from sklearn.model_selection import train_test_split
   ```
   We split training and testing.
   We use the training for build the model and we use the testing to evaluate the performance of the model.

3. Scaling: normalization and standardization:
    a. Standarization: The values are centered around the mean with a unit standard deviation.

$$z = \frac{x_i - \mu}{\sigma}$$

    b. Normalization: The values are shifted and rescaled.

$$x_{\text{norm}} = \frac{x - \min(x)}{\max(x) - \min(x)}$$



4. We need to convert the pandas dataframe to np array.

Implementation

To build the model we use Sequential().

```
Layer (type)                 Output Shape              Param #

=================================================================

dense_7 (Dense)              (None, 32)                256


_________________________________________________________________

dense_8 (Dense)              (None, 15)                495


_________________________________________________________________

dense_9 (Dense)              (None, 10)                160


_________________________________________________________________

dense_10 (Dense)             (None, 6)                 66


_________________________________________________________________

dense_11 (Dense)             (None, 4)                 28


_________________________________________________________________

dense_12 (Dense)             (None, 6)                 30


_________________________________________________________________

dense_13 (Dense)             (None, 6)                 42


_________________________________________________________________

dense_14 (Dense)             (None, 4)                 28

=================================================================

Total params: 1,105

Trainable params: 1,105

Non-trainable params: 0
```

The optimizer that we use is Adam.

The **Adam optimization algorithm** is an extension to stochastic gradient descent that has recently seen broader adoption for deep learning applications in computer vision and natural language processing.

Adam is an optimization algorithm that can be used instead of the classical stochastic gradient descent procedure to update network weights iterative based in training data.

The loss that we use sparse categorical cross entropy:

$$\text{Loss} = -\sum_{j=1}^{K} y_j \log(\hat{y}_j)$$

where k is number of classes in the data

Refinement

The model suffers from underfitting. So, to overcome underfitting:

1. New dataframe with highly recommended features and dependent features:
    a. more layers and hidden units
2. Train the model longer
3. Advanced optimization algorithm

Section 4: Results

Model Evaluation and Validation

To evaluate the model we use a validation set:

```
Train on 190933 samples, validate on 81829 samples
Epoch 1/15
 - 15s - loss: nan - acc: 0.2433 - val_loss: nan - val_acc: 0.2448
Epoch 2/15
 - 15s - loss: nan - acc: 0.2434 - val_loss: nan - val_acc: 0.2448
Epoch 3/15
 - 15s - loss: nan - acc: 0.2434 - val_loss: nan - val_acc: 0.2448
Epoch 4/15
 - 11s - loss: nan - acc: 0.2434 - val_loss: nan - val_acc: 0.2448
Epoch 5/15
 - 15s - loss: nan - acc: 0.2434 - val_loss: nan - val_acc: 0.2448
Epoch 6/15
 - 14s - loss: nan - acc: 0.2434 - val_loss: nan - val_acc: 0.2448
Epoch 7/15
 - 14s - loss: nan - acc: 0.2434 - val_loss: nan - val_acc: 0.2448
Epoch 8/15
 - 10s - loss: nan - acc: 0.2434 - val_loss: nan - val_acc: 0.2448
Epoch 9/15
 - 10s - loss: nan - acc: 0.2434 - val_loss: nan - val_acc: 0.2448
Epoch 10/15
 - 15s - loss: nan - acc: 0.2434 - val_loss: nan - val_acc: 0.2448
Epoch 11/15
 - 15s - loss: nan - acc: 0.2434 - val_loss: nan - val_acc: 0.2448
Epoch 12/15
 - 16s - loss: nan - acc: 0.2434 - val_loss: nan - val_acc: 0.2448
Epoch 13/15
 - 13s - loss: nan - acc: 0.2434 - val_loss: nan - val_acc: 0.2448
Epoch 14/15
 - 14s - loss: nan - acc: 0.2434 - val_loss: nan - val_acc: 0.2448
Epoch 15/15
 - 16s - loss: nan - acc: 0.2434 - val_loss: nan - val_acc: 0.2448
```

```
model.evaluate(X_test , y_test)
```

```
81829/81829 [==============================] - 7s 86us/step

[nan, 0.2448154077419889]
```

Justification

Underfitting techniques:

- Kernel Initializer: “normal”: To start with some weights and then update to better values
- Increase number of hidden layer and units

Section 5: Conclusion

Reflection

The most occuring event is 'offer_received', so I tried to do a model with that, but the results of the model seems like not so good.

There is no change in rate of accuracy it keep constant.

Major classes perform well but not minorities classes. This is because an imbalance dataset.

Improvement

Potential improvement: Design, analysis and build deep learning model