Starbucks Challenge: Predicting effective offers using app user data

Michael Bong
Sep 20 · 16 min read

In this project, we are aiming to predict effective offers to be provided to app users.

Depending on the characteristics of an app user, they will be provided with a discount, or BOGO, or informational offer.

Ideally, we want to provide app users with the type of offer they are most likely to act on/respond to — ie. an 'effective offer'.

Photo by quan le on Unsplash

. . .

Business questions

In order to guide the project, we focus on the following business questions:

1. For each type of offer, what features (offer characteristics, user characteristics) increases its effectiveness?
2. For each type of offer, can we predict the likelihood of an app user responding to the offer?

. . .

Skeleton of the plan

A quick roadmap for the project:

- Import packages
- Explore the 3 datasets (portfolio, profile, transcript)
- Preprocess the datasets
- Engineer useful features
- Combine the datasets
- Implement models
- Choose best model
- Highlight important features

- Reflect on findings and potential future improvements

. . .

Photo by Aurélien - Wild Spot on Unsplash

Import packages

We start by importing all the packages required:

```
import pandas as pd
import numpy as np
import math
import json
import seaborn as sns
from datetime import datetime

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import mean_squared_error
from sklearn.metrics import classification_report
```

```
from sklearn.metrics import classification_report
from time import time
from sklearn.model_selection import GridSearchCV
import matplotlib.pyplot as plt
from sklearn.linear_model import Ridge
from sklearn.tree import DecisionTreeRegressor

pd.set_option('display.max_columns', None)
% matplotlib inline
```

. . .

Photo by Franki Chamaki on Unsplash

Import and explore datasets

We will be exploring all 3 datasets:

- portfolio
- profile

- transcript

Key steps taken here:

- Importing the datasets

```
# Importing the datasets
# read in the json files
portfolio = pd.read_json('data/portfolio.json', orient='records', lines=True)
profile = pd.read_json('data/profile.json', orient='records', lines=True)
transcript = pd.read_json('data/transcript.json', orient='records', lines=True)
```

- Create a generic meta data report for each dataset

```
def explore_data(df, n_unique_value=20):
    '''
    input:
    - df: data frame to explore
    - n_unique_value: number of unique values to present in report

    outputs:
    - report containing meta data on input data frame
    '''
```

- Explore certain fields in more detail
- Clean each dataset
- Generate new fields from existing fields

Dataset 01: Portfolio data

Source:

- portfolio.json

Contains:

- offer ids and meta data about each offer (duration, type, etc.)

Fields:

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

- BOGO offer: Buy One, Get One free
- Discount offer: Discount with purchase
- Informational offer: Provides product information, without price discount or free products

- 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)

. . .

Key callouts and actions from report:

- There are no null values
- Rename ‘difficulty’ to ‘offer_difficulty’
- Rename ‘duration’ to ‘offer_duration’
- Rename ‘reward’ to ‘offer_reward’
- ‘channels’ column should be broken into ‘channel_email’, ‘channel_mobile’, ‘channel_social’, ‘channel_web’

Create expanded columns Drop original column

- ‘offer_type’ column should be broken into ‘offer_bogo’, ‘offer_informational’, ‘offer_discount’

Create expanded columns Drop original column

- ‘portfolio.id’ is equivalent to ‘transcript.value[offer_id]’

Rename 'portfolio.id' to 'portfolio.offer_id'

- A quick view of the updated dataset:

offer_difficulty	offer_duration	offer_id	offer_type	offer_reward	channel_email	channel_mobile	channel_social	channel_web
10	7	ae264e3637204a6fb9bb56bc8210ddfd	bogo	10	1	1	1	0
10	5	4d5c57ea9a6940dd891ad53e9dbe8da0	bogo	10	1	1	1	1
0	4	3f207df678b143eea3cee63160fa8bed	informational	0	1	1	0	1
5	7	9b98b8c7a33c4b65b9aebfe6a799e6d9	bogo	5	1	1	0	1
20	10	0b1e1539f2cc45b7b9fa7c272da2e1d7	discount	5	1	0	0	1
7	7	2298d6c36e964ae4a3e7e9706d1fb8c2	discount	3	1	1	1	1
10	10	fafdcd668e3743c1bb461111dcafc2a4	discount	2	1	1	1	1
0	3	5a8bc65990b245e5a138643cd4eb9837	informational	0	1	1	1	0
5	5	f19421c1d4aa40978ebb69ca19b0e20d	bogo	5	1	1	1	1
10	7	2906b810c7d4411798c6938adc9daaa5	discount	2	1	1	0	1

Dataset 02: Profile data

Source:

- profile.json

Contains:

- demographic data for each customer

Fields:

- 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

. . .

Key actions and callouts from report:

- Deep dive into ‘age’, ‘gender’, ‘income’
- There are 2175 null values in ‘gender’, ‘income’ columns

Clean the null values

- Clean the null values

- Create an alternative view of ‘became_member_on’ field

  - Calculate the number of days each person has been a member ('days_as_members')
  - Deep dive into 'days_as_members'

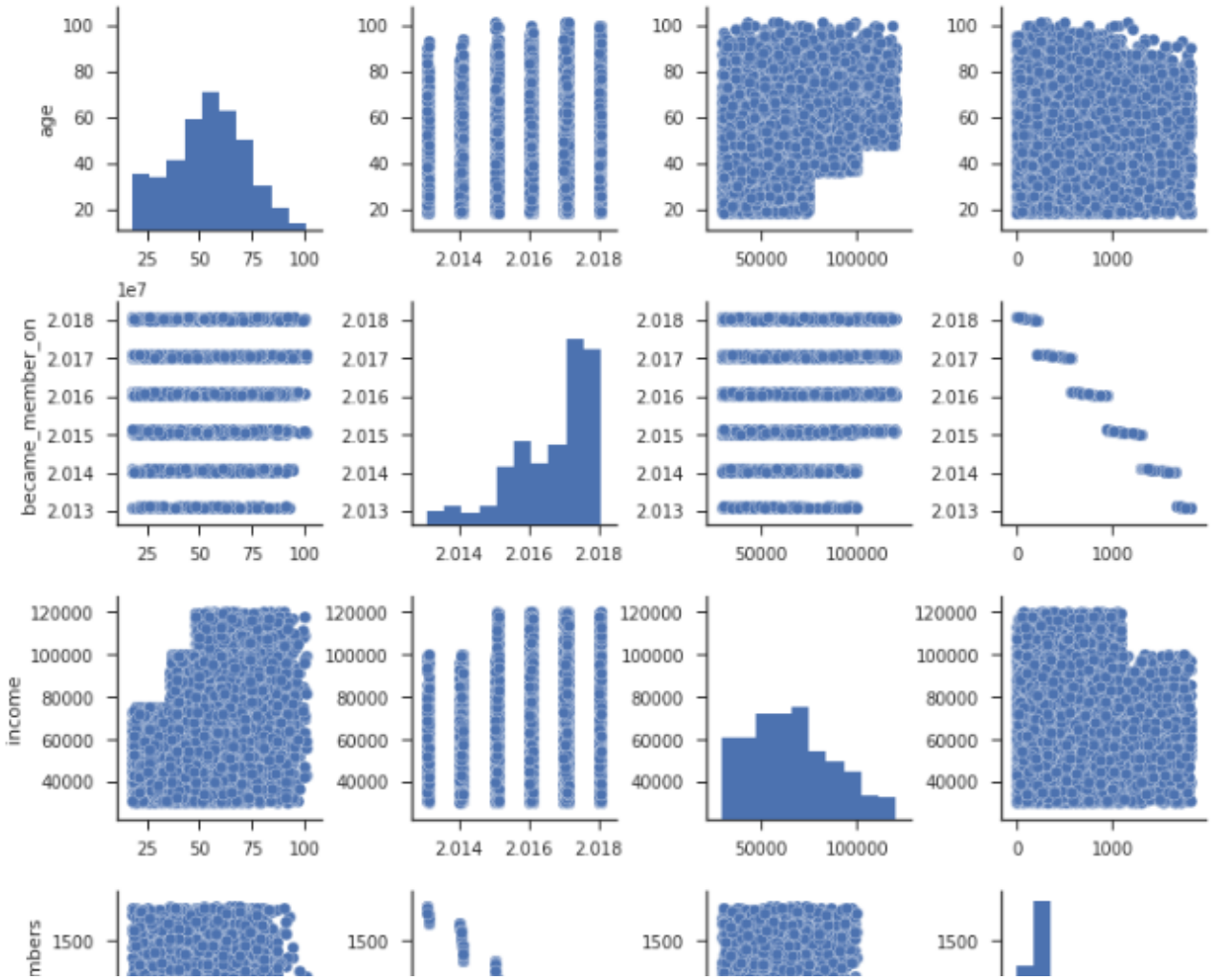
- Quick visualisations of all fields relation to one another

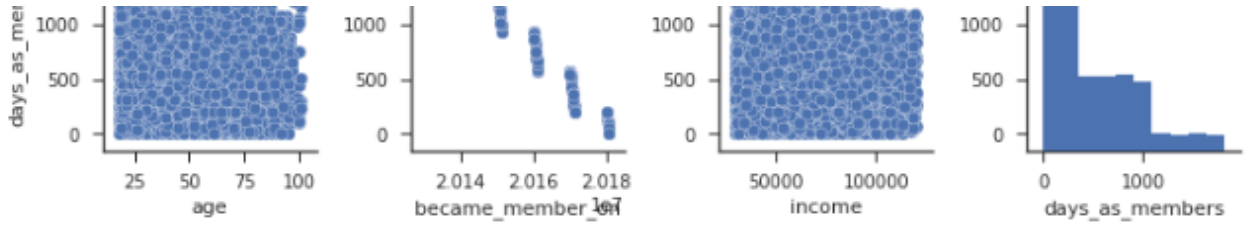
  - Correlation plot
  - Pair plot

- ‘profile.id’ is a primary key, and equivalent to ‘transcript.person’

  - Rename 'profile.id' to 'profile.cust_id'

- A pairplot of the relationship between fields:





- A quick view of the updated dataset

	age	became_member_on	gender	cust_id	income	became_member_on_date	days_as_members
1	55	20170715	F	0610b486422d4921ae7d2bf64640c50b	112000.0	2017-07-15	376
3	75	20170509	F	78afa995795e4d85b5d9ceeca43f5fef	100000.0	2017-05-09	443
5	68	20180426	M	e2127556f4f64592b11af22de27a7932	70000.0	2018-04-26	91
8	65	20180209	M	389bc3fa690240e798340f5a15918d5c	53000.0	2018-02-09	167
12	58	20171111	M	2eeac8d8feae4a8cad5a6af0499a211d	51000.0	2017-11-11	257

Dataset 03: Transcript data

Source:

- transcript.json

Contains:

- records for transactions, offers received, offers viewed, and offers completed

Fields:

- 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

. . .

Key actions and callouts from report:

- There are no null values
- ‘value’ column should be broken into

- 'offer_id'
- 'amount'
- 'reward'

- ‘transcript.person’ is a key, and equivalent to ‘profile.id’

  - rename to 'cust_id' for consistency

- ‘transcript.value[offer_id]’ is equivalent to ‘portfolio.id’

  - rename to 'offer_id' for consistency

- A quick view of the cleaned data set:

	event	cust_id	time	amount	reward	offer_id
0	1. offer received	78afa995795e4d85b5d9ceeca43f5fef	0	NaN	NaN	9b98b8c7a33c4b65b9aebfe6a799e6d9
1	1. offer received	a03223e636434f42ac4c3df47e8bac43	0	NaN	NaN	0b1e1539f2cc45b7b9fa7c272da2e1d7
2	1. offer received	e2127556f4f64592b11af22de27a7932	0	NaN	NaN	2906b810c7d4411798c6938adc9daaa5
3	1. offer received	8ec6ce2a7e7949b1bf142def7d0e0586	0	NaN	NaN	fafdcd668e3743c1bb461111dcafc2a4
4	1. offer received	68617ca6246f4fbc85e91a2a49552598	0	NaN	NaN	4d5c57ea9a6940dd891ad53e9dbe8da0

. . .

Photo by Shane Aldendorff on Unsplash

Data pre-processing

We will take the following steps to prepare the data for next steps:

- Process 01: Assign offer_id to each transaction (where applicable)
- Process 02: Catering for 'none' offer transactions
- Process 03: Merging the cleaned 'Portfolio', 'Profile', 'Transcript' data sets
- Process 04: Determine the distinct types of transactions, whether it is driven by the offer and categorize each transaction
- Process 05: Finalise creation of merged data set for use in models and analysis — overall and by offer type

. . .

Process 01: Assigning offer_id to each transaction — where applicable

We start by merging the transcript and portfolio tables. This provides the base for us to determine how many instances of each event happens for each offer type.

A few key points that we can see below:

- for 'bogo' and 'discount' offers
  - there are events - offer 'received', 'viewed', 'completed'
  - the number of event instances decreases as it moves from offer 'received' -> 'viewed' -> 'completed', suggesting that not all received offers are viewed, and not all viewed offers are completed
- for 'informational' offers
  - there are events - offer 'received', 'viewed', but none for 'completed'
  - the number of event instances decreases as it moves from offer 'received' -> 'viewed'
- for 'all' offers
  - there are no 'transaction' events

From these points we can formulate some guidelines and actions:

1. we need to associate ‘transaction’ events to an ‘offer_id’
2. a complete event journey for

- 'bogo' and 'discount' offers = 'offer received' -> 'offer viewed' -> 'transaction' -> 'offer completed'
- 'informational' offers = 'offer received' -> 'offer viewed' -> 'transaction'

We then take the following steps:

Steps we will take are as follows:

1. Explore the data
2. Identify ‘offer_start_time’ for each ‘offer viewed’ based on associated ‘offer received’
3. Identify ‘offer_start_time’ for each ‘offer completed’ based on associated ‘offer received’
4. Re-assign ‘offer_id’ for each ‘transaction’ event that is followed by an ‘offer_completed’ event
5. Assign an ‘offer_id’, ‘offer_start_time’ to each ‘transaction’ event if there a preceding ‘offer received’ event

A quick view of the updated transcript dataset:

cust_id	time	event	offer_id	offer_start_time	amount	reward	offer_difficulty	offer_duration
0009655768c64bdeb2e877511632db8f	168	1. offer received	5a8bc65990b245e5a138643cd4eb9837	168.0	NaN	NaN	0.0	3.0
0009655768c64bdeb2e877511632db8f	192	2. offer viewed	5a8bc65990b245e5a138643cd4eb9837	168.0	NaN	NaN	0.0	3.0
0009655768c64bdeb2e877511632db8f	228	3. transaction	5a8bc65990b245e5a138643cd4eb9837	168.0	22.16	NaN	0.0	3.0
0009655768c64bdeb2e877511632db8f	336	1. offer received	3f207df678b143eea3cee63160fa8bed	336.0	NaN	NaN	0.0	4.0
0009655768c64bdeb2e877511632db8f	372	2. offer viewed	3f207df678b143eea3cee63160fa8bed	336.0	NaN	NaN	0.0	4.0
0009655768c64bdeb2e877511632db8f	408	1. offer received	f19421c1d4aa40978ebb69ca19b0e20d	408.0	NaN	NaN	5.0	5.0
0009655768c64bdeb2e877511632db8f	414	3. transaction	f19421c1d4aa40978ebb69ca19b0e20d	408.0	8.57	NaN	5.0	5.0
0009655768c64bdeb2e877511632db8f	414	4. offer completed	f19421c1d4aa40978ebb69ca19b0e20d	408.0	NaN	5.0	5.0	5.0
0009655768c64bdeb2e877511632db8f	456	2. offer viewed	f19421c1d4aa40978ebb69ca19b0e20d	408.0	NaN	NaN	5.0	5.0
0009655768c64bdeb2e877511632db8f	504	1. offer received	fafdcd668e3743c1bb461111dcafc2a4	504.0	NaN	NaN	10.0	10.0
0009655768c64bdeb2e877511632db8f	528	3. transaction	fafdcd668e3743c1bb461111dcafc2a4	504.0	14.11	NaN	10.0	10.0
0009655768c64bdeb2e877511632db8f	528	4. offer	fafdcd668e3743c1bb461111dcafc2a4	504.0	NaN	2.0	10.0	10.0

		completed						
0009655768c64bdeb2e877511632db8f	540	2. offer viewed	fafdcd668e3743c1bb461111dcafc2a4	504.0	NaN	NaN	10.0	10.0

Process 02: Catering for 'none' offer transactions

Here, we ensure that all the datasets are modified to handle 'none' offer transactions. These are transactions that are genuinely not associated to any offers.

Key steps taken are as follows:

1. Modify the offer duration (days) in the 'portfolio' data set to (hours)
2. Modify the 'portfolio' data set to cater for independent transactions
3. Replace the offer id of independent transactions in the 'Transcript' data set with '0000'
4. Replace the Null offer start time of independent transactions in the 'Transcript' data set with event time

Process 03: Merging the cleaned 'Portfolio', 'Profile', 'Transcript' data sets

Next, we merge the 3 cleaned datasets and remove Nulls from the merged set. The following steps are taken:

1. Reset the 'transcript' data set to include only required fields
2. Reset the 'portfolio' data set to include only required fields
3. Reset the 'profile' data set to include only required fields
4. Merge the 3 data sets
5. Remove records with Null values in merged set

A quick view of the cleaned, merged data set as follows:

	cust_id	time	event	offer_id	offer_start_time	amount	offer_type
0009655768c64bdeb2e877511632db8f	168	1. offer received	5a8bc65990b245e5a138643cd4eb9837	168.0	0.00	informational
0009655768c64bdeb2e877511632db8f	192	2. offer viewed	5a8bc65990b245e5a138643cd4eb9837	168.0	0.00	informational
0009655768c64bdeb2e877511632db8f	228	3. transaction	5a8bc65990b245e5a138643cd4eb9837	168.0	22.16	informational
0009655768c64bdeb2e877511632db8f	336	1. offer received	3f207df678b143eea3cee63160fa8bed	336.0	0.00	informational

0009655768c64bdeb2e877511632db8f	372	2. offer viewed	3f207df678b143eea3cee63160fa8bed	336.0	0.00	informational	

offer_duration_hours	offer_difficulty	offer_reward	channel_email	channel_mobile	channel_social	channel_web	age	gender	income	days_as_members
72	0	0	1	1	1	0	33.0	M	72000.0	461.0
72	0	0	1	1	1	0	33.0	M	72000.0	461.0
72	0	0	1	1	1	0	33.0	M	72000.0	461.0
96	0	0	1	1	0	1	33.0	M	72000.0	461.0
96	0	0	1	1	0	1	33.0	M	72000.0	461.0

Process 04: Determine the distinct types of transactions, whether it is driven by the offer and categorize each transaction

Here, we take these steps to determine derive what is an effective offer:

1. Calculate offer end time and flag if events ocurred within each offer time frame
2. Determine the events for each customer, offer — to see if it the offer influenced the events
3. Categorize each customer, offer interaction into — effective offer, ineffective offer

This is a key part of project as we are generating the ‘effective offer’ field. This is the field that we are aiming to try and predict from the other fields.

We initially flag ‘effective offer’ at the customer id, offer id, offer start time level. The criteria for an effective offer differs by offer type as follows:

- **Effective offer criteria for offer type = informational**

1. sequence of events: viewed time < transacted time (offer was viewed and then transacted)
2. valid events: offer received - viewed - transacted
3. valid amount/spend: > 0

- **Effective offer criteria for offer type = bogo**

1. sequence of events: viewed time < completed time (offer was viewed and then completed)
2. valid events: offer received - viewed - transacted - completed
3. valid amount/spend: > 0

- **Effective offer criteria for offer type = discount**

1. sequence of events: viewed time < completed time (offer was viewed and then completed)
2. valid events: offer received - viewed - transacted - completed
3. valid amount/spend: > 0

Next, we consolidate the 'effective offer' flag at the the customer id, offer id level. The following steps are followed to generate the consolidated flag — For each cust id, offer id:

- Calculate the percentage of effective offer instances / total offer instances
- For the 'lenient' consolidated percentage — If the percentage is >= 0.0, then mark as effective offer = 1
- For the 'rigid' consolidated percentage — If the percentage is >= 0.5, then mark as effective offer = 1
- **As there are immaterial differences between the instance counts of the 'lenient' and 'rigid' consolidated percentages, the decision was made to use the 'lenient' consolidated flag as proxy for effective offer at the cust id, offer id level.**

Percentage of customers with effective offers - lenient:

	effective_offer_lenient	cust_id
0	0	0.592504
1	1	0.407496

Additionally, we also categorised a customer-offer interaction into categories:

- **Categ_01: Effective offers — customers influenced by offers successfully**
- **Categ_02: Ineffective offers — customers influenced by offers unsuccessfully**

  - received -> viewed -> do nothing

- **Categ_03: Ineffective offers — customers not influenced by offers (did act)**

  - no offers - transaction
  - has offers - transacted/completed offer before viewing offer

- **Categ_04: Ineffective offers — customers not influenced by offers (did not act)**

  - received -> do nothing

We will be utilising this in future improvements in the project. A quick distribution by categories and offer types can be seen below:

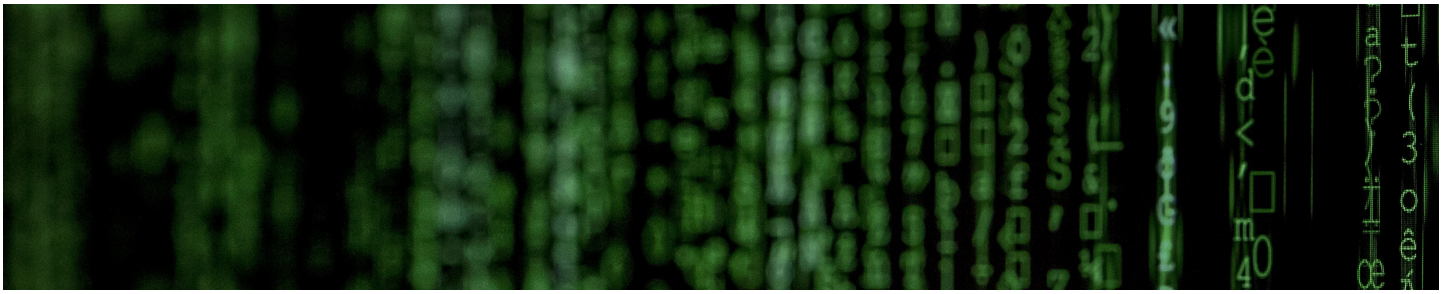
	offer_type	effective_offer	cust_categ	count
0	bogo	0	2	3063
1	bogo	0	3	8611
2	bogo	0	4	1099
3	bogo	1	1	9289
4	discount	0	2	2022
5	discount	0	3	8106
6	discount	0	4	1746
7	discount	1	1	10245
8	informational	0	2	2384
9	informational	0	3	2934
10	informational	0	4	1817
11	informational	1	1	3906
12	none	0	3	2300

Process 05: Finalise creation of merged data set for use in models and analysis — overall and by offer type

Here, we will create the final model-ready set for each offer type and overall:

```
Informational offer final set has: 11041 rows, 40 columns
Bogo offer final set has: 22062 rows, 40 columns
Discount offer final set has: 22119 rows, 40 columns
None offer final set has: 2300 rows, 40 columns
-------------------------------------------------
Overall final set has: 57522 rows, 44 columns
```

. . .



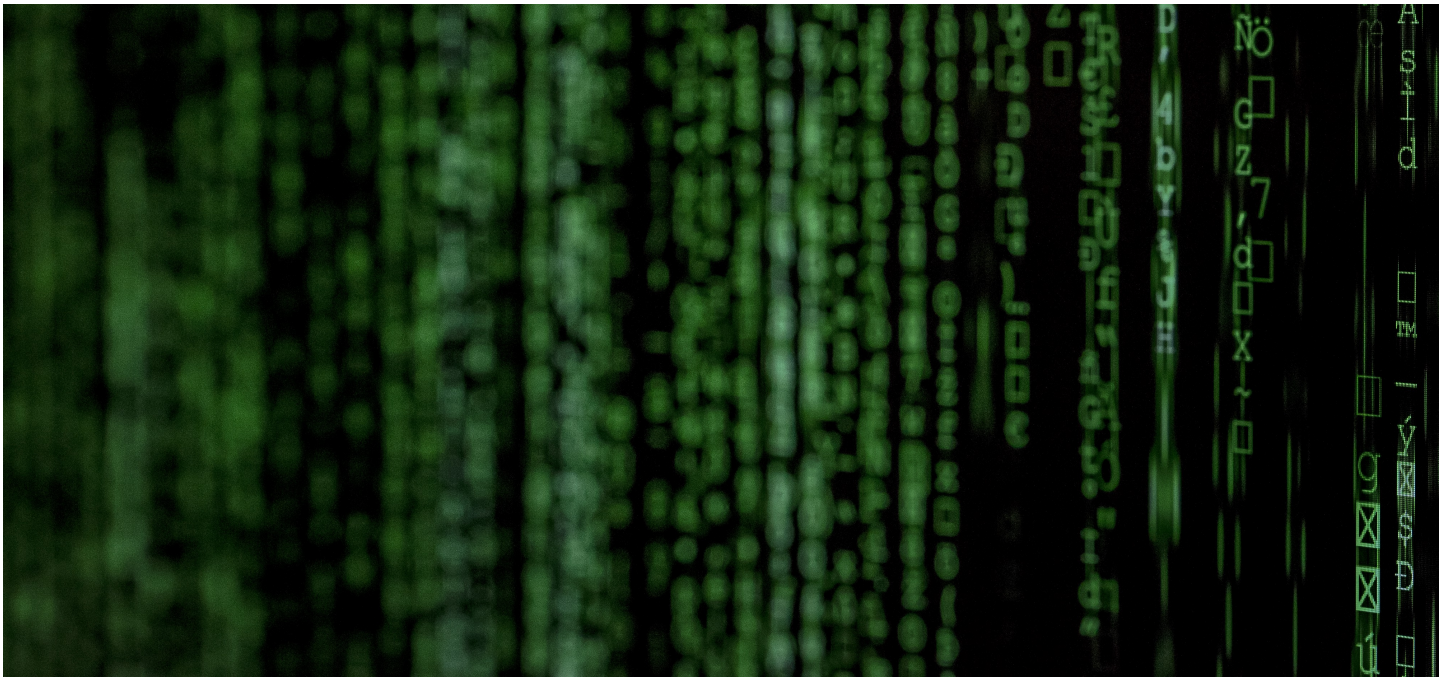


Photo by Markus Spiske on Unsplash

Model implementation

Here comes the fun part — model implementation! We will perform these key steps:

1. Implementation 01: Create functions to prepare the model implementation pipeline
2. Implementation 02: Run 3 models (Baseline, Initial, Optimised) for each offer types ('Informational', 'Bogo', 'Discount', 'Overall')
3. Implementation 03: Assess the accuracy of each model and choose the best model
4. Implementation 04: Determine feature importance

Implementation 01: Create functions to prepare the model implementation pipeline

Key functions to be prepared:

Function 1: Data prep — breaks input dataframe into 'target', 'feature'

```
# function to: prepare the modelling dataset to only contain fields to be considered for prediction
def data_prep(df,drop_cols_list=[]):
    '''
    inputs:
    - df: prepared dataframe for modeling
    - drop_cols_list: list of column names to be dropped, if no drop list is provided - pass an empty list
    outputs:
    - Returns 2 dataframes - features and target dataframes
    '''
```

Function 2: Model pipeline — splits the input dataframe into 'train', 'test', perform scaling

```
def model_pipeline(features,target):
    '''
    inputs:
    - features & target dataframe (created from data_prep function)

    outputs:
    - Splits features and target dataframe to train and test sets, performs feature scaling on both datasets.
    - Outputs X_train, X_test, y_train and y_test dataframes
    '''
```

Function 3: Train model

```
def train_predict(learner, X_train, y_train, X_test, y_test):
    '''
    inputs:
       - learner: the learning algorithm to be trained and predicted on
       - sample_size: the size of samples (number) to be drawn from training set
       - X_train: features training set
       - y_train: review_scores_rating training set
       - X_test: features testing set
       - y_test: review_scores_rating testing set

    outputs:
       - results - accuracy, classification report
    '''
```

Function 4: Run model

```
def run_model(clf, X_train, y_train, X_test, y_test, name):
    '''
    inputs:
    - clf: classifier model
    - X_train: train set
    - y_train: train set
    - X_test: test set
    - y_test: test set
    - name: name of model

    outputs:
    - dataframe of results from model training and prediction
    '''
```

Function 5: Grid search to find optimum parameters for model

```
def rand_forest_param_selection(X,y):
    '''
    input:
    - X,y: training datasets for X and y (X_train, y_train from data_prep() function)

    output:
    - dictionary with best parameters for random forest model
    '''
```

Function 6: Repeatable model run and optimisation

```
def run_optimise_model(offer_type_model, model_base_set_df, drop_cols_list, baseline_model_name, model_name,
                       optimised_model_name, use_preoptimised_param='Y'):
    '''
    input:
    - offer_type_model: string name of offer type
    - model_base_set_df: data set to be used for modelling
    - drop_cols_list: list of columns to be dropped from modelling data set
    - baseline_model_name: string name of baseline model
    - model_name: string name of initial model
    - optimised_model_name: string name of optimised model
    - use_preoptimised_param: 'Y' to use previously derived optimum parameters for the model,
                              'N' to re-derive optimum parameters

    output:
    - data frame of optimised model results
    - optimised model
    '''
```

We will use random state = 23.

Please note the functions here are heavily borrowed from previous Udacity projects that I have done in prior terms.

Implementation 02: Run 3 models (Baseline, Initial, Optimised) for each of the 5 offer types ('Informational', 'Bogo', 'Discount', 'None', 'Overall')

A few key details here:

- 5 offer types are investigated here: 'informational', 'bogo', 'discount', 'none', 'overall'
- Baseline model used for comparison is a Decision Tree Classifier
- We have chosen the Random Forest Classifier for all offer types

Steps taken:

1. Baseline model is run for each offer type
2. Initial, unoptimised model is then run
3. A grid search is performed to determine the optimum parameters for each model
4. Final, optimised model is run

For the 'Informational' offer model:

```
----------------------------------------------------------------------------------------------------
Model creation for "informational" offer type
----------------------------------------------------------------------------------------------------

----------------------------------------------------------------------------------------------------
Initial, pre-optimisation run results: Baseline model (baseline)
```

```
----------------------------------------------------------------------------------------------------
##################################################
DecisionTreeClassifier trained on 8832 samples.
##################################################
MSE_train: 0.1835
MSE_test: 0.2187
##################################################
Training accuracy: 0.8165
Test accuracy: 0.7813
##################################################
Prediction time: 0.0013539791107177734
##################################################
             precision    recall  f1-score   support

          0     0.8960    0.7545    0.8192      1450
          1     0.6397    0.8327    0.7235       759

avg / total     0.8079    0.7813    0.7863      2209

----------------------------------------------------------------------------------------------------
Initial, pre-optimisation run results: Production model (info_1)
----------------------------------------------------------------------------------------------------
##################################################
RandomForestClassifier trained on 8832 samples.
##################################################
MSE_train: 0.1646
MSE_test: 0.2082
##################################################
Training accuracy: 0.8354
Test accuracy: 0.7918
##################################################
Prediction time: 0.026103496551513672
##################################################
             precision    recall  f1-score   support

          0     0.8678    0.8055    0.8355      1450
          1     0.6732    0.7655    0.7164       759

avg / total     0.8009    0.7918    0.7946      2209



----------------------------------------------------------------------------------------------------
---Using previously identified optimum parameters for each model---
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
---Previous optimum parameters loaded---
--------------------------------------------------
Optimum parameters as follows:
{'max_depth': 10, 'max_features': 'auto', 'min_samples_leaf': 2, 'min_samples_split': 10, 'n_estimators': 20}
----------------------------------------------------------------------------------------------------


----------------------------------------------------------------------------------------------------
Final, post-optimisation run results: Optimised Production model (info_1_optimised)
----------------------------------------------------------------------------------------------------
##################################################
RandomForestClassifier trained on 8832 samples.
##################################################
MSE_train: 0.1452
MSE_test: 0.2078
##################################################
Training accuracy: 0.8548
Test accuracy: 0.7922
##################################################
Prediction time: 0.026236534118652344
##################################################
             precision    recall  f1-score   support

          0     0.8880    0.7821    0.8317      1450
          1     0.6609    0.8116    0.7286       759

avg / total     0.8100    0.7922    0.7963      2209
```

For the 'Bogo' offer model:

```
-----------------------------------------------------------------------------------------------------
Model creation for "bogo" offer type
-----------------------------------------------------------------------------------------------------

-----------------------------------------------------------------------------------------------------
Initial, pre-optimisation run results: Baseline model (baseline)
-----------------------------------------------------------------------------------------------------
##################################################
DecisionTreeClassifier trained on 17649 samples.
##################################################
MSE_train: 0.1888
MSE_test: 0.1924
##################################################
Training accuracy: 0.8112
Test accuracy: 0.8076
##################################################
Prediction time: 0.002546072006225586
##################################################
             precision    recall  f1-score   support

          0     0.9134    0.7382    0.8165      2559
          1     0.7143    0.9035    0.7978      1854

avg / total     0.8298    0.8076    0.8087      4413

-----------------------------------------------------------------------------------------------------
Initial, pre-optimisation run results: Production model (bogo_1)
-----------------------------------------------------------------------------------------------------
##################################################
RandomForestClassifier trained on 17649 samples.
##################################################
MSE_train: 0.1553
MSE_test: 0.1867
##################################################
Training accuracy: 0.8447
Test accuracy: 0.8133
##################################################
Prediction time: 0.056472063064575195
##################################################
             precision    recall  f1-score   support

          0     0.8741    0.7921    0.8311      2559
          1     0.7459    0.8425    0.7913      1854

avg / total     0.8202    0.8133    0.8144      4413
```

```
-----------------------------------------------------------------------------------------------------
---Using previously identified optimum parameters for each model---
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
---Previous optimum parameters loaded---
--------------------------------------------------
Optimum parameters as follows:
{'max_depth': 10, 'max_features': 'auto', 'min_samples_leaf': 10, 'min_samples_split': 2, 'n_estimators': 30}
-----------------------------------------------------------------------------------------------------

-----------------------------------------------------------------------------------------------------
Final, post-optimisation run results: Optimised Production model (bogo_1_optimised)
-----------------------------------------------------------------------------------------------------
##################################################
RandomForestClassifier trained on 17649 samples.
##################################################
MSE_train: 0.1488
MSE_test: 0.1790
##################################################
Training accuracy: 0.8512
Test accuracy: 0.8210
##################################################
```

```
Prediction time: 0.08208060264587402
##################################################
             precision    recall  f1-score   support

          0     0.8929    0.7855    0.8358      2559
          1     0.7461    0.8700    0.8033      1854

avg / total     0.8312    0.8210    0.8221      4413
```

For the 'Discount' offer model:

```
----------------------------------------------------------------------------------------------------
Model creation for "discount" offer type
----------------------------------------------------------------------------------------------------

----------------------------------------------------------------------------------------------------
Initial, pre-optimisation run results: Baseline model (baseline)
----------------------------------------------------------------------------------------------------
##################################################
DecisionTreeClassifier trained on 17695 samples.
##################################################
MSE_train: 0.1655
MSE_test: 0.1648
##################################################
Training accuracy: 0.8345
Test accuracy: 0.8352
##################################################
Prediction time: 0.0024750232696533203
##################################################
             precision    recall  f1-score   support

          0     0.8426    0.8549    0.8487      2392
          1     0.8262    0.8120    0.8191      2032

avg / total     0.8351    0.8352    0.8351      4424

----------------------------------------------------------------------------------------------------
Initial, pre-optimisation run results: Production model (disc_1)
----------------------------------------------------------------------------------------------------
##################################################
RandomForestClassifier trained on 17695 samples.
##################################################
MSE_train: 0.1431
MSE_test: 0.1600
##################################################
Training accuracy: 0.8569
Test accuracy: 0.8400
##################################################
Prediction time: 0.056638240814208984
##################################################
             precision    recall  f1-score   support

          0     0.8574    0.8445    0.8509      2392
          1     0.8201    0.8346    0.8273      2032

avg / total     0.8403    0.8400    0.8401      4424
```

```
----------------------------------------------------------------------------------------------------
---Using previously identified optimum parameters for each model---
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
---Previous optimum parameters loaded---
--------------------------------------------------
Optimum parameters as follows:
{'max_depth': 15, 'max_features': 'auto', 'min_samples_leaf': 2, 'min_samples_split': 2, 'n_estimators': 30}
----------------------------------------------------------------------------------------------------

----------------------------------------------------------------------------------------------------
```

```
Final, post-optimisation run results: Optimised Production model (disc_1_optimised)
------------------------------------------------------------------------------------------------------------
################################################
RandomForestClassifier trained on 17695 samples.
################################################
MSE_train: 0.0792
MSE_test: 0.1600
################################################
Training accuracy: 0.9208
Test accuracy: 0.8400
################################################
Prediction time: 0.10805654525756836
################################################
             precision    recall  f1-score   support

          0     0.8824    0.8123    0.8459      2392
          1     0.7979    0.8725    0.8336      2032

avg / total     0.8436    0.8400    0.8402      4424
```

For the 'None' offer model:

```
------------------------------------------------------------------------------------------------------------
Model creation for "none" offer type
------------------------------------------------------------------------------------------------------------

------------------------------------------------------------------------------------------------------------
Initial, pre-optimisation run results: Baseline model (baseline)
------------------------------------------------------------------------------------------------------------
################################################
DecisionTreeClassifier trained on 1840 samples.
################################################
MSE_train: 0.0000
MSE_test: 0.0000
################################################
Training accuracy: 1.0000
Test accuracy: 1.0000
################################################
Prediction time: 0.00022149085998535156
################################################
             precision    recall  f1-score   support

          0     1.0000    1.0000    1.0000       460

avg / total     1.0000    1.0000    1.0000       460

------------------------------------------------------------------------------------------------------------
Initial, pre-optimisation run results: Production model (none_1)
------------------------------------------------------------------------------------------------------------
################################################
RandomForestClassifier trained on 1840 samples.
################################################
MSE_train: 0.0000
MSE_test: 0.0000
################################################
Training accuracy: 1.0000
Test accuracy: 1.0000
################################################
Prediction time: 0.0028541088104248047
################################################
             precision    recall  f1-score   support

          0     1.0000    1.0000    1.0000       460

avg / total     1.0000    1.0000    1.0000       460




------------------------------------------------------------------------------------------------------------
---Using previously identified optimum parameters for each model---
```

```
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
---Previous optimum parameters loaded---
--------------------------------------------------
Optimum parameters as follows:
{'max_depth': 5, 'max_features': 'auto', 'min_samples_leaf': 2, 'min_samples_split': 2, 'n_estimators': 10}
----------------------------------------------------------------------------------------------------


----------------------------------------------------------------------------------------------------
Final, post-optimisation run results: Optimised Production model (none_1_optimised)
----------------------------------------------------------------------------------------------------
##################################################
RandomForestClassifier trained on 1840 samples.
##################################################
MSE_train: 0.0000
MSE_test: 0.0000
##################################################
Training accuracy: 1.0000
Test accuracy: 1.0000
##################################################
Prediction time: 0.0017580986022949219
##################################################
             precision    recall  f1-score   support

          0     1.0000    1.0000    1.0000       460

avg / total     1.0000    1.0000    1.0000       460
```

For the 'Overall' offer model:

```
----------------------------------------------------------------------------------------------------
Model creation for "all" offer type
----------------------------------------------------------------------------------------------------

----------------------------------------------------------------------------------------------------
Initial, pre-optimisation run results: Baseline model (baseline)
----------------------------------------------------------------------------------------------------
##################################################
DecisionTreeClassifier trained on 46017 samples.
##################################################
MSE_train: 0.1817
MSE_test: 0.1782
##################################################
Training accuracy: 0.8183
Test accuracy: 0.8218
##################################################
Prediction time: 0.0063669681549072266
##################################################
             precision    recall  f1-score   support

          0     0.8485    0.8513    0.8499      6818
          1     0.7826    0.7790    0.7808      4687

avg / total     0.8217    0.8218    0.8218     11505

----------------------------------------------------------------------------------------------------
Initial, pre-optimisation run results: Production model (all_1)
----------------------------------------------------------------------------------------------------
##################################################
RandomForestClassifier trained on 46017 samples.
##################################################
MSE_train: 0.1561
MSE_test: 0.1633
##################################################
Training accuracy: 0.8439
Test accuracy: 0.8367
##################################################
Prediction time: 0.15441250801086426
##################################################
             precision    recall  f1-score   support
```

```
              0     0.8962    0.8193    0.8560      6818
              1     0.7663    0.8620    0.8113      4687

avg / total     0.8433    0.8367    0.8378     11505
```


Implementation 03: Assess the accuracy of each model and choose the best model

In order to determine how good the models are, we consider 4 key metrics:

- **Testing accuracy**
- **Recall**: Ability of model to find all positive samples. (True positives/(True positives + False negatives))
- **Precision**: Ability of model to not label a negative sample as positive. (True positives/(True positives + False positives))
- **F1 score**: Harmonic mean of recall and precision. (2PrecisionRecall/(Recall + Precision)) For all metrics, the closer the score is to 1, the better.

Analysis of the performance of models as follows:

- For all offer types, the optimised Random Forest model for all offer types outperformed the baseline Decision Tree model on all 4 metrics.

- The key exception here is the model for 'none' offer type. The accuracy is 1.00 as in the entire dataset for 'none' offers, there is only ineffective offers. The results for 'none' offers should be ignored.
- From a testing accuracy perspective, the 'overall' and 'discount' offer type models have the highest accuracy at 0.84, followed by 'bogo' with 0.82 and 'informational' with 0.79.
- From a recall perspective, the scores range from 0.79 to 0.84.
- From a precision perspective, the scores range from 0.81 to 0.85.
- From a F1 score perspective, the scores range from 0.79 to 0.84.
- Therefore, all the optimised Random Forest models for each offer type has accuracy scores of ~0.79 and above, which is relatively good.

Given that all 4 models are quite on par from a testing accuracy standpoint, we have the option of using the models for individual offer types or using the overall offer type model.

Implementation 04: Determine feature importance for each optimised model

We will look at the optimum model for each offer type (informational, bogo, discount, overall) and analyse the top features that drive the predictive power of each model.

Feature importance — 'informational' offer model

- The prominent feature is ‘amount_valid’ which is the amount spent by customers within the offer period. This makes sense as the more a customer spends, the more likely they will complete the offer
- Customers with higher ‘income’, has higher ‘days_as_members’, and ‘age’ are more likely to result in an effective offer. This could be as they have greater disposable income to spend on a non-price-reducing offer and perhaps being older means they have the means to do so. Having been members of Starbucks for longer also shows loyalty, which in turn mean they are more likely to act on the offer.
- The longer the ‘offer_duration_hours’ the better, as it gives customers more chance to act effectively on the offers
- It seems best to focus on ‘web’ and ‘social’ channels to deliver this offer type
- It also seems that sending customers more offers of different types (higher ‘offer_frequency_per_cust’) and of the same type (higher ‘dist_offer_id_type’) helps with generating effective offers

Feature importance — ’bogo’ offer model

Observations on features that help predict an 'effective offer' (an offer that is received -> viewed -> transacted -> completed by customer):

- The prominent feature is 'amount_valid' which is the amount spent by customers within the offer period. This makes sense as the more a customer spends, the more likely they will complete the offer. This is the main feature for all offer models.
- Customers with higher 'income', has higher 'days_as_members', and 'age' are more likely to result in an effective offer. This could be as they have greater disposable income to spend on a non-price-reducing offer and perhaps being older means they have the means to do so. Having been members of Starbucks for longer also shows loyalty, which in turn mean they are more likely to act on the offer.
- The longer the 'offer_duration_hours' the better, as it gives customers more chance to act effectively on the offers
- It seems best to focus on the 'social' channel to deliver this offer type
- It also seems that sending customers more offers of different types (higher 'offer_frequency_per_cust') only helps with generating effective offers
- 'offer_difficulty' and 'offer_reward' also play a part here

Feature importance — 'discount' offer model

Observations on features that help predict an 'effective offer' (an offer that is received -> viewed -> transacted -> completed by customer):

- The prominent feature is 'amount_valid' which is the amount spent by customers within the offer period. This makes sense as the more a customer spends, the more likely they will complete the offer. This is the main feature for all offer models.
- It seems best to focus on the 'social' channel to deliver this offer type, followed by the 'mobile' channel
- Customers with higher 'income', has higher 'days_as_members', and 'age' are more likely to result in an effective offer. This could be as they have greater disposable income to spend on a non-price-reducing offer and perhaps being older means they have the means to do so. Having been members of Starbucks for longer also shows loyalty, which in turn mean they are more likely to act on the offer.
- 'offer_difficulty' and 'offer_reward' also play a part here
- It also seems that sending customers more offers of the same type (higher 'dist_offer_id_type') only helps with generating effective offers
- The longer the 'offer_duration_hours' the better, as it gives customers more chance to act effectively on the offers

Feature importance — 'overall' offer model

Observations on features that help predict an 'effective offer' (an offer that is received -> viewed -> transacted -> completed by customer):

- The prominent feature is 'amount_valid' which is the amount spent by customers within the offer period. This makes sense as the more a customer spends, the more likely they will complete the offer. This is the main feature for all offer models.
- It seems best to focus on the 'social' channel to deliver this offer type, followed by the 'mobile' and 'email' channels
- Customers with higher 'income', has higher 'days_as_members', and 'age' are more likely to result in an effective offer. This could be as they have greater disposable income to spend on a non-price-reducing offer and perhaps being older means they have the means to do so. Having been members of Starbucks for longer also shows loyalty, which in turn mean they are more likely to act on the offer.
- The longer the 'offer_duration_hours' the better, as it gives customers more chance to act effectively on the offers
- 'offer_reward' also play a part here

. . .

Photo by Anton Shuvalov on Unsplash

Conclusion

Overall, this was a great project which provides plenty of opportunity to apply key skills such as data and feature engineering, and modelling.

Let's circle back to the business questions that were formed to guide our project and see how well we have covered them.

Then, let us look at potential future improvements.

Reflection — Answering the business questions

Let's see if we have been able to answer the business questions.

Question 1: For each type of offer, what features (offer characteristics, user characteristics) increases its effectiveness?

Each type of offer has varying features that drive the effectiveness of an offer.

In general, the key predictive features are 'amount_valid' which is the amount spent by customers within the offer period. This makes sense as the more a customer spends, the more likely they will complete the offer. This is the main feature for all offer models.

Other predictive features are:

- Customer features where those with higher 'income', has higher 'days_as_members', and 'age' are more likely to result in an effective offer. This could be as they have greater disposable income to spend on a non-price-reducing offer and perhaps being older means they have the means to do so. Having been members of Starbucks for longer also shows loyalty, which in turn mean they are more likely to act on the offer.

- Offer features such as 'offer_difficulty', 'offer_reward'. The longer the 'offer_duration_hours' the better, as it gives customers more chance to act effectively on the offers. It also seems that sending customers more offers of different types (higher 'offer_frequency_per_cust') and of the same type (higher 'dist_offer_id_type') helps with generating effective offers.

- Channel features where effective channels are 'social', 'mobile', 'email' — in order of effectiveness.

Question 2: For each type of offer, can we predict the likelihood of an app user responding to the offer?

We have been able to create optimised models that will enable us to target users with an offer they are most likely to respond to. Separate models have been created for each offer type.

- The 'overall' and 'discount' offer type models have the highest accuracy at ~84%, followed by 'bogo' with ~82% and 'informational' with ~79%.

- These are relatively high accuracy scores and should enable us to confidently target customers with offer types that they are most likely to respond positively to.

- Given that all 4 models are quite on par from a testing accuracy standpoint, we have the option of using the models for individual offer types or using the overall offer type model.

Potential future improvements

There are multiple things that we can explore further in the future as follows:

- Performing a deep dive on the customer groupings that were created to better understand the demographics of customers we are sending offers to.

- Further optimising the offer models and using more bespoke models.

. . .

Thanks for reading and hope you find this helpful! The notebook for this project can be accessed via this link: https://github.com/MikeBong/udacity_datascience_nd/tree/master/project_capstone_starbucks_challenge

Machine Learning