# Starbucks Offer-Based Sales Predictions Capstone Project Final Report

## 1. DEFINITION

### 1.1 Project Overview

The project proposed is the Starbucks project, which is provided in the capstone course information. This project is aimed at analyzing promotional offers sent to Starbucks consumers, with the goal of improving offer planning, segmentation, and targeting decisions in the future. This project represents a classic business scenario where machine learning could be applied to increase business sales by optimizing marketing operations, and hence increase marketing return on investment.

In marketing, it is important to get a better picture and better insights about the customers at an individual level. More precisely, it is important to be able to answer questions such as: what are the best offers or offer characteristics that excite certain individuals? What offers are certain individuals receptive to, or repelled by? Machine learning has the power to unlock such deep insights into customer tastes and behaviors, since machine learning algorithms are capable of analyzing or synthesizing a huge number of possible relationships between data points, or features, that can affect a specific insight.

The dataset provided covers only one product (for simulation purposes), and consists of 3 tables/files:

- Portfolio - containing meta data about each offer (duration, type, etc.).
- Profile - demographic data for each customer.
- Transcript - records for transactions, offers received, offers viewed, and offers completed. Covers one-month worth of data.

### 1.2 Problem Statement

Starbucks customers receive offers periodically on various channels such as web, social media, and the Starbucks app. These offers usually involve one specific product and could include a reward that is claimable after spending a specific amount on this product. Each offer has some characteristics such as channels used for sending the offer, offer type, offer difficulty (how much spending is required to claim the offer), and reward amount (if there is a reward).

The problem is defined as follows: given a potential offer about a product with known characteristics and given an eligible customer base that could be targeted for this offer. The goal is to estimate how much each customer would spend on this one product, given they have received and viewed this offer. A potential solution would be a model that is used to make a batch prediction on how much each customer would spend, given they have viewed it. 2 potential benefits/uses of such prediction are:

- Allows for designing better offers that customers are more willing to spend on. Various offers with different combinations of characteristics could be run through the model, then the offer with highest predicted total spending would be sent.
- Before sending an offer, estimating how much customers would spend allows for better targeting decisions, for example by targeting the top 20% of predicted spending.

The solution proposed is a regression model, which predicts how much each customer would spend given an offer. The model is a multi-dimensional regression model. Its input features are customer demographic data (age, gender, income), and offer characteristics (offer type, difficulty, reward). This is achievable by training a linear learner. The model would be tested against a test split of the dataset until it meets evaluation criteria. Once deployed, it could receive one offer at a time as its input. It could be used to predict spending per customer, given this new offer.

## 1.3 Metrics

A good regression model is one where the difference between the actual or observed values and predicted values for the selected model is small and unbiased for train, validation and test data sets. For this problem, 2 metrics will be used: Root Mean Squared Error (RMSE), R-squared, and Adjusted R-squared. The model would be tuned to minimize RMSE for training, validation, and test data sets, and to maximize R-squared value. The R-squared value will be used as the main metric for evaluating model performance and comparing with the benchmark model.

# 2. ANALYSIS

## 2.1 Data Exploration

### 2.1.1 Data in provided tables

The first table in the dataset is “portfolio”, which contains metadata about each offer. Below are some statistics regarding the “portfolio” table:

- 10 records in total.
- 4 bogo offers, 4 discount offers, 2 informational offers.
- All offers were sent via email channel, as well as at least one other channel from (web, mobile, social).
- Bogo offers have higher reward and lower difficulty, and reward=difficulty.
- Total duration per offer type: {'Bogo': 24 days, 'Discount': 34 days, 'Informational': 7 days}.

The second table is “profile”. Below are some statistics regarding the “profile” table:

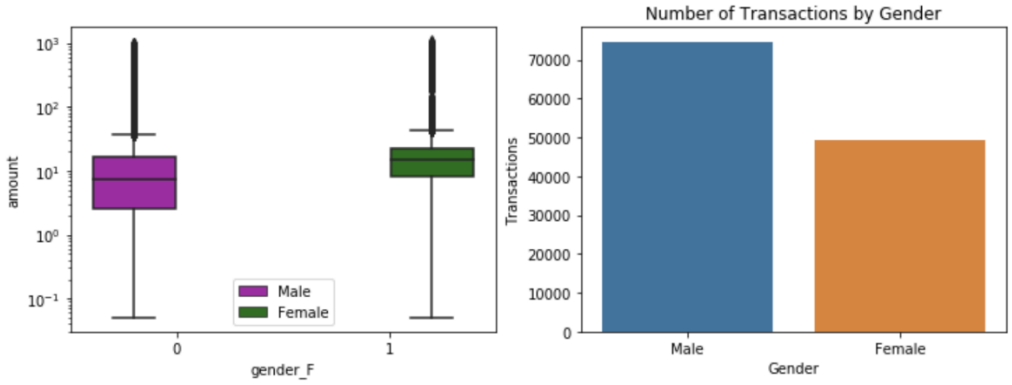
- 17000 profiles in total.
- Gender distribution: {'M': 8484, 'F': 6129, 'O': 212, 'None': 2175}.
- All profiles with Gender=’None’ have Income=’None’ as well. This means they can be omitted.
- Age:
    - Range: [18:118]
    - Mean: 62.5
    - Standard Deviation: 26.7
- Income:
    - Range: [30000:120000]
    - Mean: 65,405
    - Standard Deviation: 21,598
- Became Member On: earliest year is 2013, and latest is 2018.

The third table is “transcript”. Below are some statistics regarding the “transcript” table:

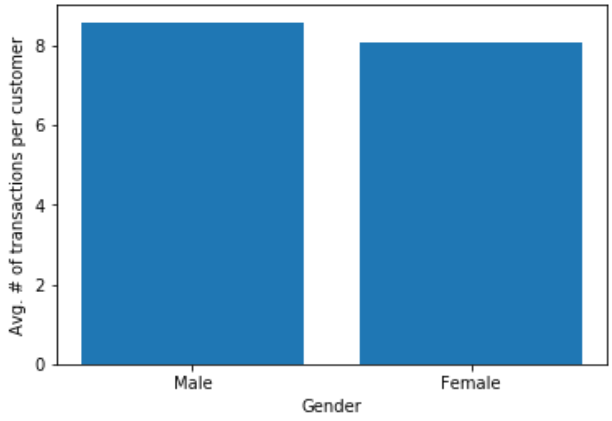
- Events distribution: {'transaction': 138953, 'offer received': 76277, 'offer viewed': 57725, 'offer completed': 33579}.
- Total sales (sum of all transactions): 1,774,898

### 2.1.2 Transactions Data vs Profiles

The transactions found in the transcript table can be broken down further to find some interesting insights regarding customer demographics and their generic purchasing behavior. First, here are some info when comparing genders, and their purchasing behavior (some of this info was extracted from the benchmark model page provided in the project proposal [1], and some was explored further):




[1]




The info above provides some insights such as:

- Male customers have more total transactions than female customers (expected since there are more male customers).
- Male customers have higher average number of transactions per customer: {'Male': 8.58, 'Female': 8.06}.
- Female customers on average spend more per transaction than male customers.

Below is a table with more details and a bit more analysis:

| | Male_amount | Female_amount | Male_age | Female_age | Male_income | Female_income |
|---|---|---|---|---|---|---|
| **count** | 72794.000000 | 49382.000000 | 72794.000000 | 49382.000000 | 72794.000000 | 49382.000000 |
| **mean** | 11.606600 | 17.490077 | 50.725650 | 55.637986 | 58491.317966 | 66813.960552 |
| **std** | 28.964542 | 35.342611 | 17.755604 | 17.776608 | 19048.170156 | 21965.016643 |
| **min** | 0.050000 | 0.050000 | 18.000000 | 18.000000 | 30000.000000 | 30000.000000 |
| **25%** | 2.570000 | 7.970000 | 37.000000 | 43.000000 | 43000.000000 | 51000.000000 |
| **50%** | 7.050000 | 15.000000 | 52.000000 | 56.000000 | 56000.000000 | 65000.000000 |
| **75%** | 16.220000 | 22.120000 | 64.000000 | 68.000000 | 70000.000000 | 82000.000000 |
| **max** | 977.780000 | 1062.280000 | 100.000000 | 101.000000 | 120000.000000 | 120000.000000 |

[1]

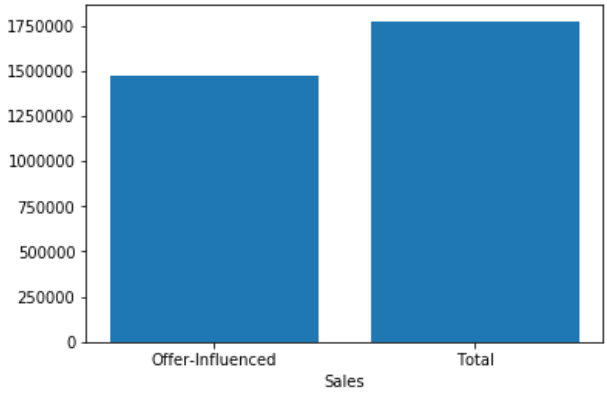
The table above provides additional insights such as:

- Female customers on average have a higher income than make customers.
- Female customers on average are older than male customers.
- Female customers spend more per transaction than male customers.

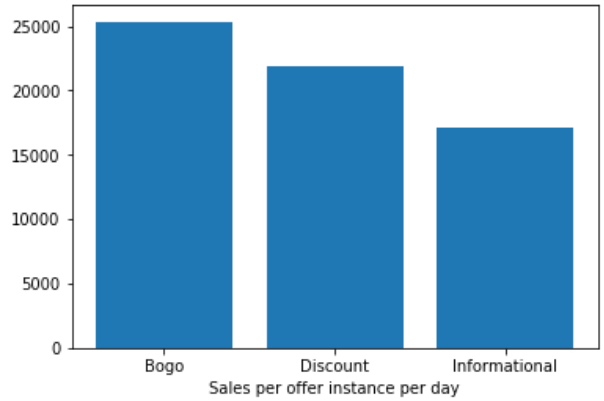
### 2.1.3 Transactions Data vs Offers

In order to prepare the data for modeling, many data processing steps was performed. The main goal behind these steps was to find all transactions that were influenced by each offer instance. An offer-influenced transaction is one where it happens within the duration of an offer going on, but only if such offer was received AND viewed by a customer. The exact data processing details will be provided later. For now, below are some relevant statistics regarding the dataset, after all processing was completed:

- Total Offer-Influenced Sales: 1,472,650
    - Offer-influenced sales / Total sales = 82.97%.




- Total Sales per offer instance per day: {'Bogo': 25346, 'Discount': 21903, 'Informational': 17093}




The first graph above shows that offer-influenced sales make up about 83% of the total sales recorded during that month for this one Starbucks product. While it is true that some customers will make a purchase regardless of the presence of an offer influencing their purchase, an assumption has to be made about the influence of an offer given that it has been viewed by the customer, and the transaction occurred within its duration.

The second graph above is calculated by dividing the total sales per offer type over the total number of days that each offer type covers. This graph is an interesting one, since it provides an overview of offer type as a standalone factor on sales made as a result of this offer. The data shown makes sense, since Bogo offers provide the highest reward/difficulty ratio (difficulty=reward), compared to discount offers (small reward) and informational offers (no reward).

However, this information alone is not enough. First, it does not justify using more Bogo offers, since the higher reward could more than offset the higher sales. Second, this graph does not take into account the effect of other features/factors, specifically demographic features of customers who made these purchases. This is where this project comes in. The goal is to synthesize all these features by creating a Machine Learning model that should help selecting the best customers to target for each type of offer, to ensure higher sales overall.

## 2. Algorithms and Techniques

Predicting offer-influenced sales values is a regression problem, so a linear regression model is appropriate. The algorithm used to build this model is the built-in AWS SageMaker algorithm "Linear Learner". This algorithm provides a solution for both classification and regression problems, allowing for exploring different training objectives simultaneously and choosing the best solution from a validation set. It allows the user to explore a large number of models and choose the best, which optimizes either continuous objectives such as mean square error, cross entropy loss, absolute error, etc., or discrete objectives suited for classification such as F1 measure, precision@recall, accuracy. The implementation provides a significant speedup over naive hyperparameter optimization techniques and an added convenience, when compared with solutions providing a solution only to continuous objectives [2].

## 3. Benchmark

As mentioned previously, the 2 metrics that will be used to determine the model performance are RMSE, and R-squared value. RMSE will be minimized during training, and R-squared will be used as the main performance-defining metric of the model. The benchmark model that was selected can be found here [1]. According to this model, the benchmark values of R-squared are summarized as follows:

- 0.0791369487837 with SVM Regressor
- -0.0823391370849 with Default Random Forest Regressor
- 0.069935792705576483 with Tuned Random Forest Regressor

Hence, the model to be trained must produce an R-squared value of at least 0.079 to successfully outperform the benchmark model.

# 3. METHODOLOGY

## 3.1 Data Preprocessing

### 3.1.1 Obtaining Input Dataset

For this project, the bulk of the work was data preprocessing and preparation. This is especially necessary for this dataset since offer and transaction data are combined into one table, with no indication of how they are related to each other. The main input of this project by design requires a dataset that displays for each person/offer combination, what is the total sales value/amount during the one-month period covered by the transcript table provided.

To achieve this output, the following preprocessing steps were performed:

- The transcript table was split into offer-related records, and transaction records.
- The offer-related records split was filtered to only contain “offer received” and “offer viewed” records.
- The offer-related records split was joined with the portfolio (offers) table, to obtain the offer duration field from that table. The duration was changed from days to hours.
- The end-date of each offer-related record (whether received or viewed) was updated with a new field “valid until”. This field is calculated by adding the “offer received” time with the duration field (in hours). This calculated field would be the same for that offer instance and all its related records (offer received, offer viewed).
- Now, select only the records where the offer was viewed, and merge (full-outer join) that with the transaction records, on the person field.
- Mark the records where the transaction was influenced by the offer viewed. Happens when “Valid until (offer) >= Time (transaction) >= Time (Offer viewed)”, then filter to keep only those records.
- Deduplicate this data on (person, offer, transaction) tuples. This step is required to avoid instances of the same transactions that belong to the same offer more than once. This situation could happen if the same offer was sent twice, with 2 overlapping time periods, and a transaction happens to belong to both instances of the offer. This ensures it is counted once.
- Finally, aggregate the dataset by applying a sum on the amount field, and grouping by (person, offer) combinations for each sum.
- Join this resulting table with the original portfolio (offers) and profile tables to obtain the (potential) model features in the same table. The dataset is now ready.

### 3.1.2 Processing Model Input Features

In order to prepare the model features to be input into a regression model, they need to be processed appropriately. Here are the features and how they were processed:

- Offer channels: there are a total of 4 channels. The email channel is always used, so it is irrelevant to the model. The channels column was split into 3 columns (web, mobile, social). Each column is a binary (1,0) value representing whether that channel was used or not.
- Offer type: there are 3 types of offers (Bogo, discount, informational). The offer type was replaced with 2 columns (Bogo, discount), which have a binary (1,0) value representing whether an offer is a Bogo or discount. If it is informational, both columns would have 0. This is basically Dummy Coding strategy.

- Gender: there are 3 genders in the dataset (M, F, O). The same Dummy Coding strategy was used, with 2 columns (M, F) replacing the Gender column.
- The "became member on" column was replaced with "membership years" column. The latest membership year was 2018, and that was used as the point of reference for this column data. Membership years contain an integer representing the number of years a customer has been a member for, where 2018 is 0 years, 2017 is 1 years, etc.

Finally, a Standard Scaler was applied to the entire dataset. It is worth noting that the model with Standard Scaler applied on its data ended up performing better than when a Normalizer was applied. This is expected for a regression problem with a relatively small dataset. The data was split into train-test splits (80/20). Then the train data was split into train-validation (80/20). The data files were saved locally and uploaded to S3.

The final dataset has the following number of records:

- Total: 32845 records
- Training: 21020 records
- Validation: 5256 records
- Test: 6569 records

## 3.2 Implementation

As mentioned, the model was created using the SageMaker Linear Learner model. This was done in a SageMaker notebook instance. First, the data is loaded into the modeling notebook. Then the Linear Learner model was initialized with the "regressor" model type. The optimizer function was "sgd" for Stochastic Gradient Descent. The loss function was initialized as "rmse" for Root Mean Squared Error. Training epochs was set at 100 (with a default early stopping of 3 epochs unchanged). The number of calibration samples (or the validation set samples) was set at 3000. It is worth noting that the validation set is not used when training a basic Linear Learner, and instead this "num_calibration_samples" input is specified instead. This means that the validation set was unnecessary, so it was concatenated with the training, forming a larger training set. The validation set is to be used in the refinement process though.

The model was trained (took around 3 minutes), and then deployed to make inferences (took around 7 to 8 minutes). The model was used to batch predict the test set. An evaluation function was defined to calculate the main metric of the model (R2 score). The R2 score would consistently be in the range (0.075 to 0.080). This is always the case with multiple train-test splits of the dataset.

## 3.3 Refinement

While the results achieved in the initial model was not very good when compared with the benchmark model, and it could definitely be improved. The first major refinement step was feature selection. It is important to eliminate features that do not have an impact on the model (redundant). It is even more important to eliminate features that adversely affect model performance. After some feature selection and iterative modeling, omitting some features improved performance significantly.

The features removed were:

- Channels (web, mobile, social): this makes sense because channel are usually not an important factor in encouraging purchases. The choice of channel(s) usually affects the likelihood that each customer would view an offer. The dataset assumes that all offers have been viewed already.
- Offer difficulty and reward: while the inclusion of these offers makes sense, performance was not affected at all when these features were removed.

Refining feature selection drastically improved the model's performance. The model's R2 score would be consistently in the range (0.088 to 0.095). This is compared to the benchmark model's (0.079).

The second refinement step tried was using SageMaker's "Hyper Parameter Tuner" to better tune the model's hyperparameters. The main hyperparameter to tune was "wd", which represents the L2 regularization bias term. This term was ranged between (1e-7 and 1e2). The tuner's objective parameter name was set at "validation:objective_loss". In regression, this represents the RMSE value of the validation dataset (dataset loaded via the validation channel). The objective metric is set to be minimized.

Unfortunately, using the Hyperparameter Tuner to improve the model did not yield better results. In fact, it yielded worse results, where the R2-score value was around (0.080). One possible reason for that is that in the Linear Learner's documentation, it is mentioned "The implementation provides a significant speedup over naive hyperparameter optimization techniques". This implies that tuning is performed as part of the modeling a Linear Learner. This means that for example, the Linear Learner automatically finds the best value of the L2 regularization bias term "wd" by utilizing the validation set's specified number of samples (via the input variable "num_calibration_samples").

It is still possible that the tuning process could be explored further in a way to eventually improve the model performance. This could be through better range tuning for the regularization term, and potentially tuning more model variable inputs. The model was tuned thoroughly; however, it was eventually considered acceptable to not succeed at improving model performance in the end. This issue was considered minor since the model trained without the Hyperparameter tuner already outperformed the benchmark model by a significant margin.

# 4. Results

## 4.1 Model Evaluation and Validation

The model and its results were detailed in the previous section. The Linear Learner model trained managed to achieve an R2-score of (0.088 to 0.095) when run on the testing dataset. To test model robustness, the model was run on both the training and testing datasets. Here are the R2-score results for one of the running instances (these results were consistently the same for multiple independent train-test splits):

- Training: 0.0934
- Testing: 0.0898

The performance drop for the test set compared to the training set is 3.9%. This means that the model dropped very little performance when predicting data points it never trained on. Thus, the model is considered robust enough and can make good predictions on new data points.

## 4.2 Justification

The benchmark model R2-score is 0.079. The trained model's R2-score range is (0.089 to 0.095). This model improved the R2-value by about 11.3% to 20.3%, on average by 15%. It could still be tuned to improve the model as well. From that perspective, the model and solution are considered significant enough. As for whether it is significant to adequately solve the problem, then it can be argued that it is not very good since the R2 score is still considered much closer to 0 than 1. However, it can still be used as a very good guideline for comparing spending amounts between customers, given an offer. The main use case proposed for this project is to compare customer spending given an offer, so that the top 20% could be targeted for example. For this specific use case, this model provides an adequate solution.

# REFERENCES

[1] https://medium.com/ml2vec/using-regression-modeling-to-predict-purchasing-habits-at-starbucks-8cf6007aec62

[2] https://sagemaker.readthedocs.io/en/stable/linear_learner.html