UDACITY

Machine Learning Engineer Nanodegree

Capstone Project Report

Optimizing App Offers with Starbucks

I. PROJECT DEFINITION

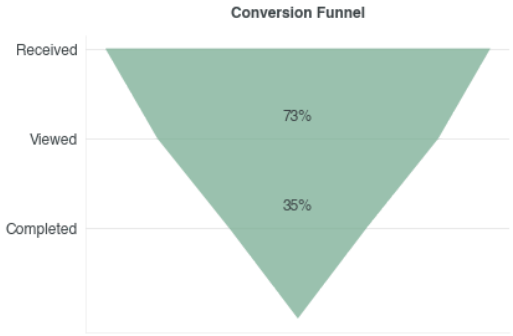
Project Overview

Starbucks is a multinational chain with over 32,000 coffeehouses worldwide and 20 million members in its loyalty program. With over 10 million downloads and 600,000 ratings on the Android Google Play store (3.8 million ratings on the Apple store), the Starbucks mobile app makes it so convenient for customers to frequently order their favorites or try the newest seasonals.

To increase its profits, the marketing and data science teams would leverage data about its customers, such as their transactions and demographics, to create personalized and timed promotions that would incentivize customers to spend more or visit more often.

Problem Statement

Using data from the company's digital channels, the objective is to use historical data to determine the best offer to send to the customer. Based on a 30-day pilot, the current conversion rate for a customer to view and complete an offer is 35%. Of all the customers who viewed the offer, 48% completed it. With more accurate targeting, this conversion rate should increase.



The other 65% that received but did not view and complete the offers could include customers who did not find the offer incentivizing enough (viewed but not completed, 38% of the total), who did not see the offer but spent enough (completed but not viewed), or who never saw the offer nor spent enough (not viewed nor completed).

Metrics

To determine which underlying model performs the best, we will evaluate based on the standard performance metrics:

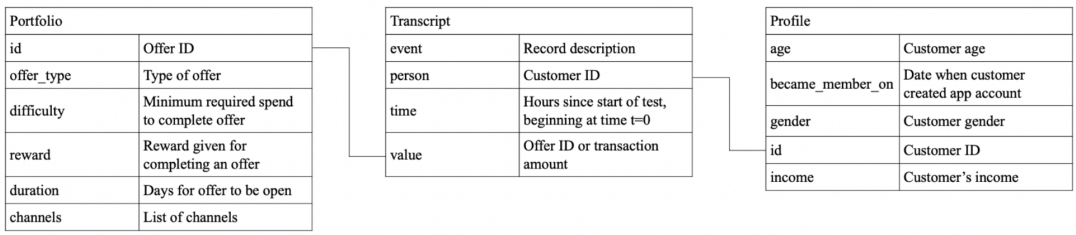
- **Accuracy** is the percentage of the dataset that the model predicted correctly
- **Precision** is the percentage of correctly predicted positive observations to the total predicted positive observations. Since positive means that the customer will respond, a high precision means that most customers will respond if we send an offer. Low precision means few customers will respond if we send an offer.
- **Recall** is the percentage of correctly predicted positive observations to the actual class. High recall means little opportunity loss, as most customers who would have responded received an offer. Low recall means we did not send offers to the majority of customers who would have responded.
- **F1** - weighted average of precision and recall, which helps prevent high imbalance between the two error types

II. ANALYSIS

Datasets and Inputs

The data contains simulated data that mimics customer behavior on the Starbucks rewards mobile app. The typical flow of a transaction through the rewards app would be receiving the offer, viewing the offer, making a transaction, and finally completing the offer. Customers can make transactions without seeing or using the offers. However, in order for a promotion to have a positive ROI, the customer must have seen the offer first. This assumes that the customer was influenced to make a purchase.

The simulated data contains 3 JSON files on offers, customer demographics, and transaction history that can be transformed into the ERD below.



For simplicity, there are only 10 offers in the portfolio, but in reality, there are hundreds over the years. The data is simulated for 30 days, leading to 306,534 events captured with 17,000 customers. Sample data is below.

`portfolio: 10 records`

	reward	channels	difficulty	duration	offer_type	id
0	10	[email, mobile, social]	10	7	bogo	ae264e3637204a6fb9bb56bc8210ddfd
1	10	[web, email, mobile, social]	10	5	bogo	4d5c57ea9a6940dd891ad53e9dbe8da0
2	0	[web, email, mobile]	0	4	informational	3f207df678b143eea3cee63160fa8bed

`profile: 17000 records`

	gender	age	id	became_member_on	income
0	None	118	68be06ca386d4c31939f3a4f0e3dd783	20170212	NaN
1	F	55	0610b486422d4921ae7d2bf64640c50b	20170715	112000.0
2	None	118	38fe809add3b4fcf9315a9694bb96ff5	20180712	NaN

`transcript: 306534 records`

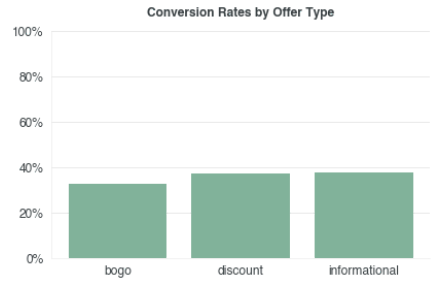
	person	event	value	time
0	78afa995795e4d85b5d9ceeca43f5fef	offer received	{'offer id': '9b98b8c7a33c4b65b9aebfe6a799e6d9'}	0
1	a03223e636434f42ac4c3df47e8bac43	offer received	{'offer id': '0b1e1539f2cc45b7b9fa7c272da2e1d7'}	0
2	e2127556f4f64592b11af22de27a7932	offer received	{'offer id': '2906b810c7d4411798c6938adc9daaa5'}	0

The datasets were flattened such that the granularity of the final table was by customer, offer ID, and the time at which the offer was received. Transactions were tied to the offer time frames and could have overlapped with other simultaneous offers for the customer.

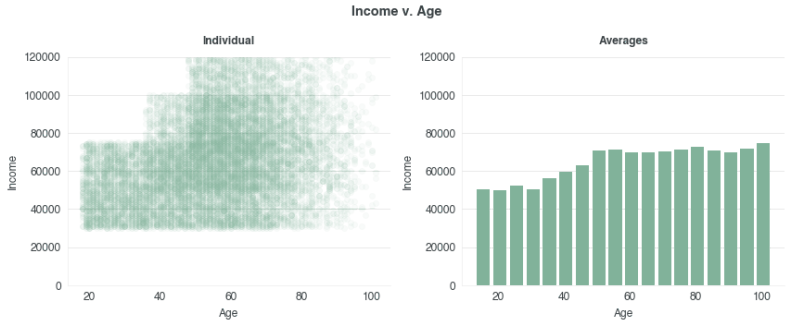
Data Exploration and Visualization

In the dataset, there were 10 total offers that were either informational, buy-one-get-one promotions, or discounts. All offers were delivered via email and had varying rewards, difficulty, and duration. Discounts had longer duration and lower rewards (not as high as BOGO's 50% off).

The breakdown of conversion rates by offer type is shown in the chart below. Discounts have a higher conversion rate than BOGOs. Informational offers had no requirement nor reward, but any transaction made during its campaign duration was counted as an "offer completion" event. Thus, the conversion rate for informational offers is likely reported to be higher than it actually is because some of those customers could have made any single purchase anyways. I chose to overreport informational offers' conversion rate instead of leaving it as 0% because this can at least be used as labels to train the model.



In addition to knowing what the offers are, it might be more important to know the types of customers. In the dataset, most individuals were between 20 and 100 years old with a few outliers older than 100. Unrealistically, there are clear distinctions with income ranges by age, but in general, income increases as age increases. I also split the dataset by gender, and females tend to fall in the older age groups and higher income groups.



Customer loyalty is an important feature because older customers tend to order more frequently and consistently than new customers. Also, Starbucks has a fast-growing membership pool.

III. METHODOLOGY

Data Preprocessing

The following steps were done with the flattened data:

- Encode categorical features such as gender and offer_type. This converts string types into numerical types for the model. For example, the 4 options for gender were turned into the 3 following columns: M, F, and O. Missing values was the 4th option but was dropped because it was recognized when M, F, and O were all equal to 0.
- Impute missing values with the mean for income, which is a continuous variable. Missing income made up at least 10% of the overall dataset, so it would have been better to give an approximate income than drop the data with missing values.
- Normalize and standardize features using Scikei-Learn's StandardScaler class. This removes the mean and scales the data by the unit variance such that most values would be between -1 and 1 (if within 1 standard deviation). Scaling all the features helps the model produce more accurate outcomes. If the data were not scaled, then the income values would be between 35,000 to 120,000 which would overshadow all other small features when calculating distances between data points for the loss function.
- Split the train and test data by randomly shuffling the observations and making a 70/30 split.

Model Selection

I used the scikit-learn library to see how different models would initially perform. The first model was the KNN benchmark model (first row in table below), but the train scores were significantly higher than the test scores which implied overfitting. To decrease the variance, I added cross-validation in subsequent models when evaluating the score against the train dataset. This corrected the benchmark model to have 67% accuracy (second row in table).

The other models I tried included support vector machine (Linear and RBF kernels), logistic regression, decision tree, random forest, and gradient boosting. The last 2 ensemble models and the support vector (RBF) performed the best, while the others were worse than the benchmark model.

	Model	Train Accuracy	Train F1	Train Precision	Train Recall	Test Accuracy	Test F1	Test Precision	Test Recall
0	KNN (no cross validation)	82.100000	66.200000	100.000000	49.500000	66.9	36.7	56.9	27.1
1	KNN	0.673029	0.375939	0.581462	0.277836	66.9	36.7	56.9	27.1
2	Linear SVC	0.666574	0.355790	0.565735	0.259840	66.5	34.8	56.2	25.2
3	SVC (RGB)	0.713606	0.488128	0.666302	0.385182	71.3	49.0	66.4	38.8
4	Logistic Regression	0.665156	0.366474	0.557353	0.273311	66.4	36.1	55.5	26.7
5	Decission Tree	0.633067	0.489620	0.482989	0.496474	64.5	50.3	50.0	50.7
6	Random Forest	0.697431	0.527090	0.591099	0.475637	70.5	54.0	60.5	48.8
7	Gradient Boosting	0.717916	0.512961	0.661523	0.419071	72.1	52.1	66.7	42.8

I chose to experiment with tuning for the support vector machine and gradient boosting models. The reason I did not choose random forest, which had a higher F1 score than the support vector machine, was that I already had gradient boosting as my tree-based, ensemble model. F1 score was a better measure than accuracy because recall would have been undervalued otherwise (the benchmark's recall was half of the precision score).

Hyperparameter Tuning

I tuned the models based on a set of hyperparameters and chose the values that generated the best F1 score so that it balances precision and recall evenly.

For the support vector classifier, the 2 hyperparameters I experimented with were C and gamma. Because the model was prone to overfitting, I tested with C values equal to or less than 1, and a lower C acts like a stronger regularization parameter by allowing a larger soft margin (more errors) when training the support vector classifier. I also tested with gamma, which describes the radius of influence each point has.

For the gradient boosting classifier, the 2 hyperparameters I experimented with were max_depth and min_samples_split. Intuitively max_depth should not be equal to the number of features and min_samples_split should not be a small number due to overfitting. The best result was based on max_depth=12 and min_samples_split=50. The table below ranks the performance of each tuned model.

The gradient boosting model outperformed the support vector classifier, likely because it combines multiple models with weak learners to produce strong learners. The trade-off of getting better accuracy is that we lose visibility to the exact levers, making it harder to interpret.

params	split0_test_score	split1_test_score	split2_test_score	split3_test_score	split4_test_score	mean_test_score	std_test_score	rank_test_score
{'max_depth': 12, 'min_samples_split': 50}	0.539417	0.541325	0.550625	0.530375	0.543723	0.541093	0.006565	1
{'max_depth': 12, 'min_samples_split': 30}	0.539528	0.543681	0.548712	0.528447	0.537174	0.539508	0.006781	2
{'max_depth': 12, 'min_samples_split': 10}	0.538451	0.542441	0.545428	0.520714	0.544696	0.538346	0.009145	3
{'max_depth': 15, 'min_samples_split': 50}	0.534907	0.540728	0.541039	0.524633	0.532894	0.534840	0.006018	4
{'max_depth': 15, 'min_samples_split': 30}	0.534475	0.534327	0.534734	0.522423	0.530971	0.531386	0.004689	5
{'max_depth': 5, 'min_samples_split': 30}	0.530619	0.533042	0.542109	0.523373	0.524524	0.530733	0.006745	6
{'max_depth': 15, 'min_samples_split': 10}	0.533775	0.531764	0.535107	0.514649	0.527420	0.528543	0.007418	7
{'max_depth': 5, 'min_samples_split': 50}	0.529854	0.533727	0.537772	0.515480	0.522632	0.527893	0.007965	8
{'max_depth': 5, 'min_samples_split': 10}	0.529801	0.530312	0.539344	0.518061	0.520075	0.527519	0.007716	9

IV. RESULTS

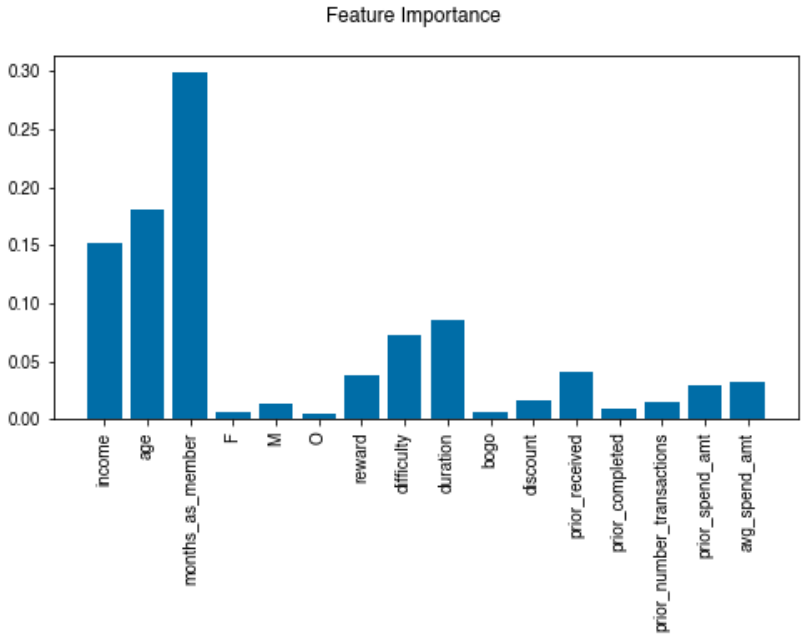
Model Evaluation and Validation

When running the tuned gradient boosting classifier against the test dataset, the accuracy was 72.5%, which is better than the benchmark model's accuracy of 66.9%. The F1 score was significantly better due to having a more balanced recall. Therefore, the proposed model would improve targeting, which increases the overall conversion rate.

Benchmark Model	Proposed Model
Accuracy: 66.9%	Accuracy: 72.5%
F1: 36.7%	F1: 56.5%
Precision: 56.9%	Precision: 64.4%
Recall: 27.1%	Recall: 50.2%

Justification

Looking at the feature importance of the model, we can see that membership age had the largest weight in the decision-making. This is aligned with the idea that customer loyalty and retention are key for a successful business. In fact, the customer profile (income, age, and months as member) had the most telling features of how they would respond to an offer. Gender and offer type had lesser importance than the offer duration and difficulty.



V. CONCLUSION

Reflection

In the capstone project, we cleaned and analyzed customer, offer, and transaction data from Starbucks and came up with a solution for improving the way we target customers with different offers. We experimented with different types of models, including support vector machines and gradient boosting, conducted hyperparameter tuning, and improved the model's accuracy and recall scores.

Personally, I found that defining the objective and setting up the problem to be high impact were the most difficult. Since the prompt was very open-ended, it was risky to propose an idea only to realize that the data might not be sufficient for the issue I wanted to solve (which was originally tying propensity scores to the company's margins and doing a one-vs-all classification with all 10 offers/classes). If I had chosen a different objective, such as identifying what **type** of offer (instead of which offer) to give, then the model and its performance would have been different.

Improvement

From a business standpoint, a model is trusted and successful if it actually generates more profits, so an improvement would be tying the customers' propensity scores to the incremental revenue for each offer. If a customer is very likely to take an offer that generates $2.00 of incremental revenue but most likely to take an offer that generates only $0.20 of incremental revenue, then the model should actually recommend the first option because it has a higher expected value.

If the model's inputs included all 10 offers for each customer, then the model should output the predicted probability (instead of predicted class) that a customer would take each offer. The final step of the calculation would be a function that chooses the offer with the highest expected value (which is the predicted probability times revenue).

If deploying this application, the model can be written using Amazon SageMaker's SKLearn module and deployed directly from Sagemaker. An AWS Lambda function can use the model's outputs to calculate the expected values and determine the offer with the maximum expected value. It's very likely that in a marketing campaign context that these decisions are not made on-demand, so we can schedule a weekly job using an orchestrator like Airflow which will call a SageMaker Operator to determine the best offers to the newest batch of eligible customers.