Starbuck's Capstone Challenge

Project Overview

This project is a real-life marketing strategy study based on simulated data set that mimics customer behavior on the Starbucks rewards mobile app. If we can gain useful insights from the data, Starbucks can better target offers towards customers with higher probability to use the offers.

At the same time, customers receive personalized and relevant offers on the mobile app and get better user experience. By doing these, Starbucks could potentially maximize revenue and save on marketing and promotional costs.

Problem Statement

In this project, I would like to combine transaction, demographic and offer data to answer the following two questions:

1. What are the main drivers of offer effectiveness?
2. Provided with offer characteristics and user demographics, can we predict whether a customer will respond to an offer?

The simulated data consists of 3 datasets:

- Offer Characteristics: portfolio
- Customer Demographics: profile
- Offer events and Transactions Records: transcript

I will construct a classification model to predict whether a customer will respond to an offer. The feature importance of the model will tell me a lot about the features that play big roles for prediction.

For the classification model,I will use both accuracy, F1-score for performance metrics. F1 Score is the weighted average of Precision and Recall. Therefore, this score takes both false positives and false negatives into account. F1 is usually more useful than accuracy, especially here we have an uneven class distribution. Accuracy works best if false positives and false negatives have similar cost. If the cost of false positives and false negatives are very different, it's better to look at both Precision and Recall.

In this project , we don't really mind false positive, that we are fine to send the offer to a person who is unlikely to respond to the offer, but we do want to reduce the false negative so we don't miss potential sales.

Data Exploration

1. Portfolio Offer data, schema:

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)

There are 10 unique offer ids, 4 bogo, 4 discount types and 2 informational types. I did one hot encoding on the 'channels' column.

	difficulty	duration	id	offer_type	reward	offer_id	channels_email	channels_mobile	channels_social	channels_web
0	10	7	ae264e3637204a6fb9bb56bc8210ddfd	bogo	10	B1	1	1	1	0
1	10	5	4d5c57ea9a6940dd891ad53e9dbe8da0	bogo	10	B2	1	1	1	1
2	0	4	3f207df678b143eea3cee63160fa8bed	informational	0	I1	1	1	0	1
3	5	7	9b98b8c7a33c4b65b9aebfe6a799e6d9	bogo	5	B3	1	1	0	1
4	20	10	0b1e1539f2cc45b7b9fa7c272da2e1d7	discount	5	D1	1	0	0	1

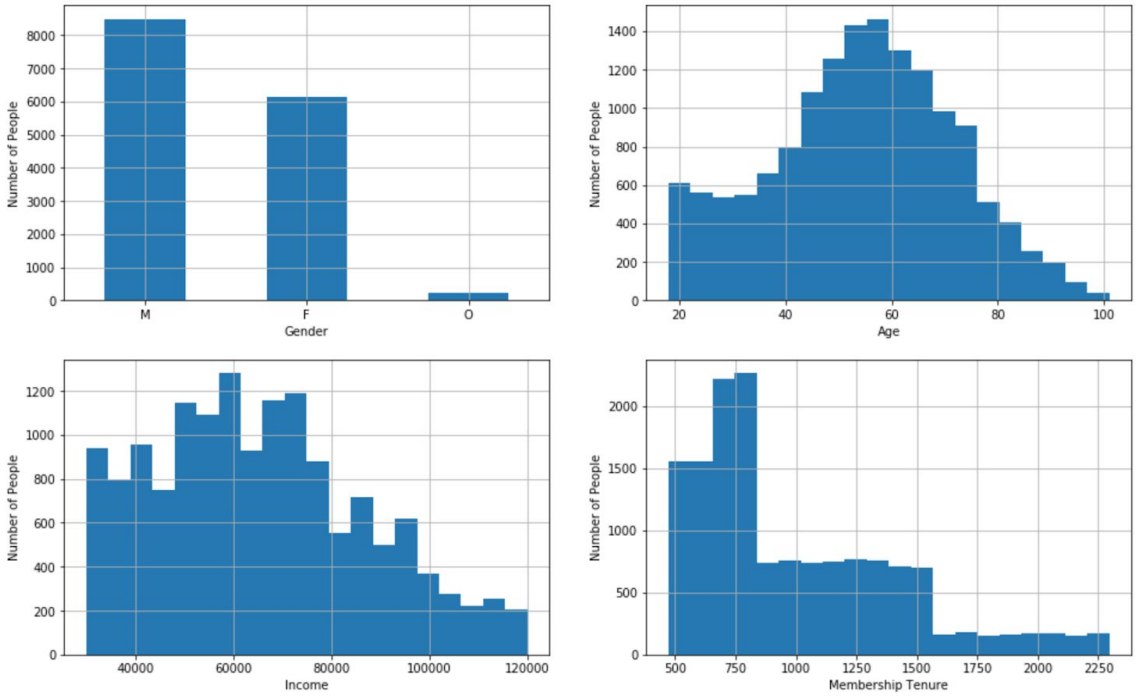
2. Customer Profile data

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

17,000 customers were contained in profile There are 2,175 person who are 118 years old without gender and income information. That is about 13% of the profile data. I also calculated that 11% of the transcript data are related to these profile. It's not significant amount and I will drop these data. I convert became_member_on column from a string of date to tenure of the membership in days.

I plotted population distributions by gender, income, age, membership tenure. There are significantly more males than females. Both the income and the age distribution seem to approach a normal distribution. There are significantly more members who joined in the recent two years.



3. Offer Events and Transaction Data

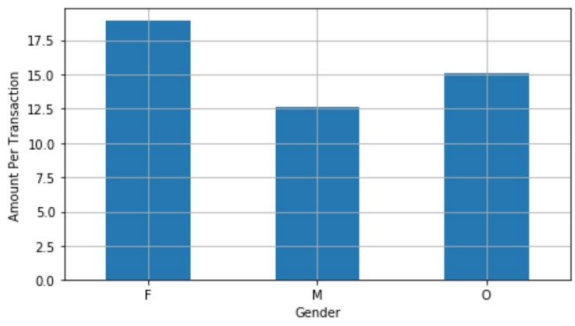
transcript.json

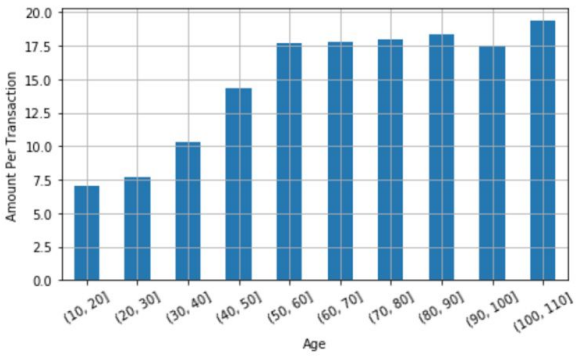
- event (str) - record description (ie transaction, offer received, offer viewed, etc.)
- person (str) - customer id
- time (int) - time in hours since start of test. The data begins at time t=0
- value - (dict of strings) - either an offer id or transaction amount depending on the record

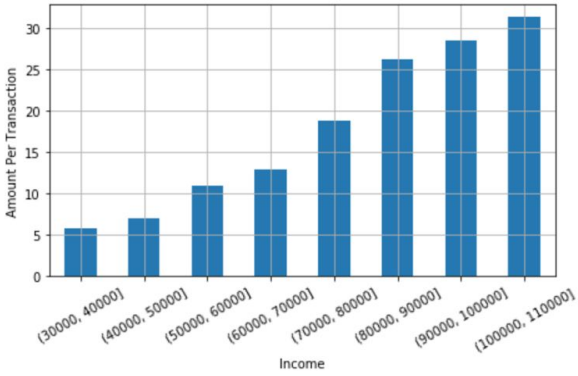
I expanded the value column to three column and merged two offer id columns into one.

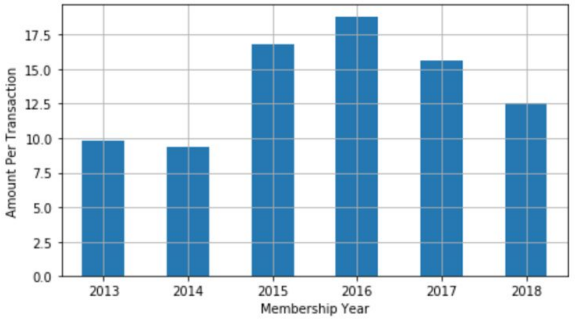
4. Merge three datasets

I merged portfolio, profile and transcript dataset and did some exploration about how the average amount customer spends on one transaction related to the demographics. By gender, women spend more money in general. Expenses and average transaction value grow as the income grows and age increases. People obtained membership in 2015-2017 spend the most.









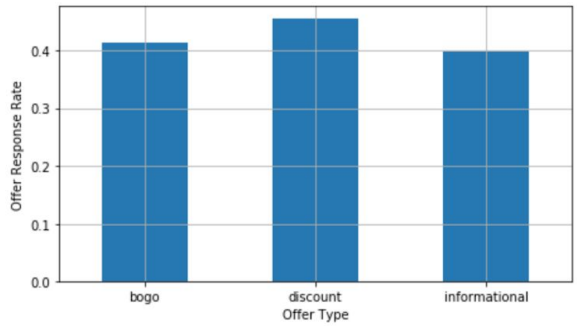
Data Preprocessing

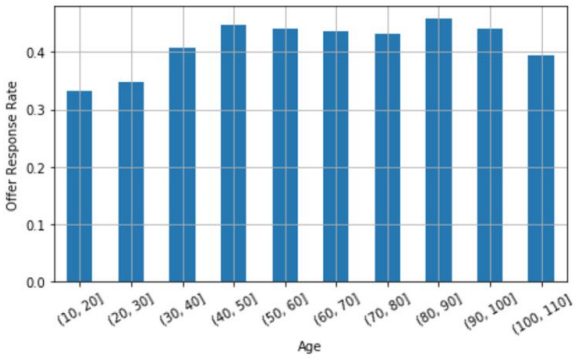
We would like to determine whether a person responds to an offer. Only a person who received, viewed and completed an offer in sequence is considered to be responded.

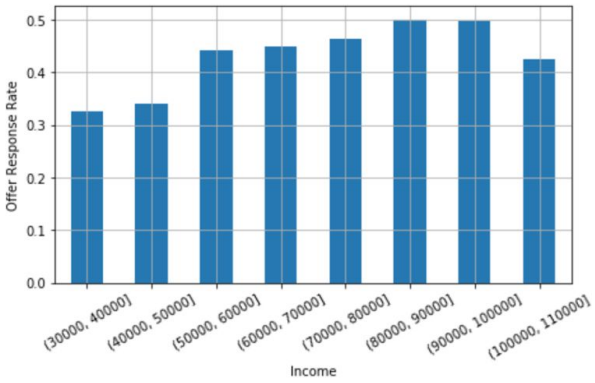
For each person in the profile, We loop through all his/her transactions to label whether that person is responsive to an offer. I also created two additional columns: received_counts and responsive_counts which provides the total number of received offers and total number of responded offers of that person. These two columns can generate the response rate for each person.

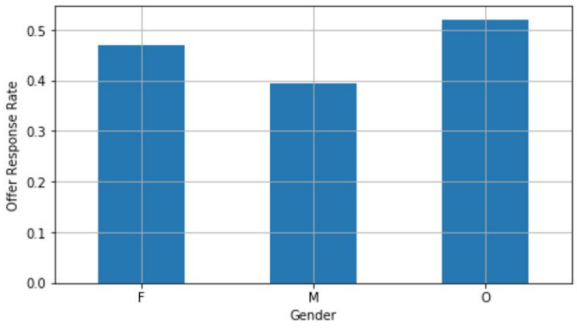
Function generate_person_offer_respond() will be called for each person, the preprocessing takes some time to run, I saved the processed data in file “data/aggregated_offer_respond.csv”.

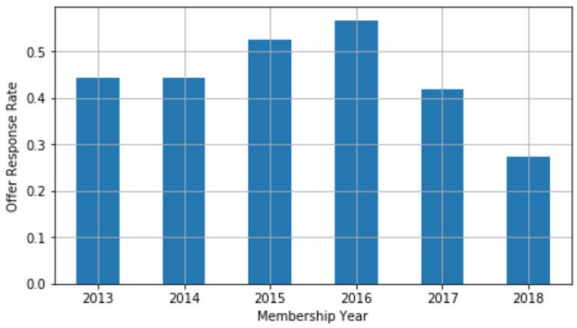
I plotted the response rate’s relationship with demographics. Discount type offer has the highest response rate. People who are over 30 have higher response rate than younger ones. People whose income are higher than 50,000 have higher response rate than lower income group. Female has higher response rate than maile. People who obtained membership in 2015 and 2016 have higher response rate.



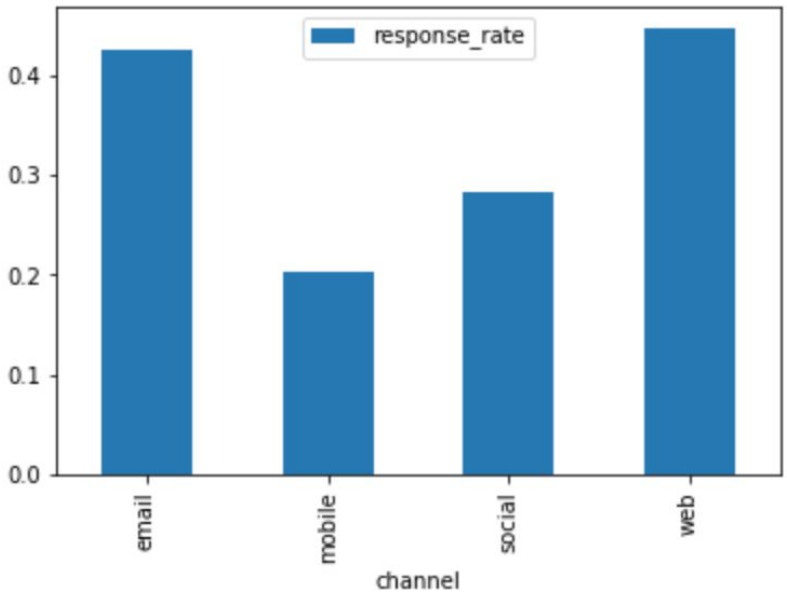






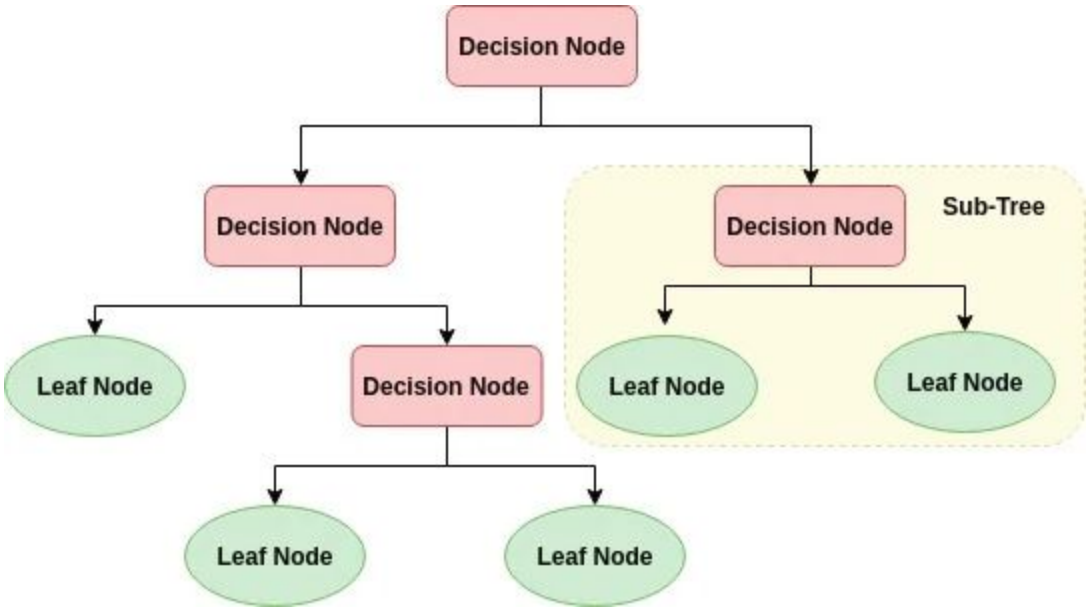


I also plotted response rate for different channels. Web and Email channels' response rate are much higher than mobile or social.



Algorithms

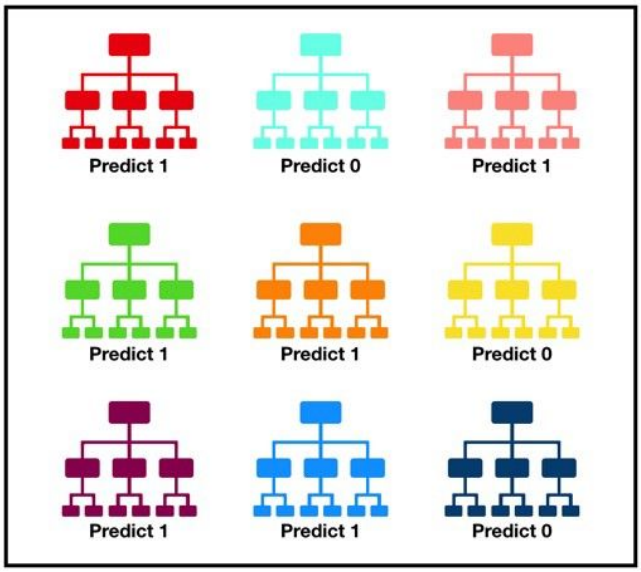
I selected decision tree as the baseline model. A decision tree is a flowchart-like tree structure where an internal node represents feature(or attribute), the branch represents a decision rule, and each leaf node represents the outcome. The topmost node in a decision tree is known as the root node. It learns to partition on the basis of the attribute value. It partitions the tree in recursive manner call recursive partitioning. The recursion is completed when the subset at a node all has the same value of the target variable, or when splitting no longer adds value to the predictions.



This flowchart-like structure helps us in decision making. That is why decision trees are easy to understand and interpret. Since I intend to analyse the feature importance to determine the drivers of an effective offer, a decision tree would provide good interpretability for me to analyse.

Meanwhile, I also selected random forest and Gradient Boosting models as improvements expecting them to perform better than the simple decision tree.

Random forest consists of a large number of individual decision trees that operate as an ensemble. Each individual tree in the random forest spits out a class prediction and the class with the most votes becomes our model's prediction (see figure below).



Tally: Six 1s and Three 0s
Prediction: 1

A large number of relatively uncorrelated models (trees) operating as a committee will outperform any of the individual constituent models.

Gradient Boosting Boosting is a method of converting weak learners into strong learners. In boosting, each new tree is a fit on a modified version of the original data set. The Algorithm begins by training a decision tree in which each observation is assigned an equal weight. After evaluating the first tree, we increase the weights of those observations that are difficult to classify and lower the weights for those that are easy to classify. The second tree is therefore grown on this weighted data. Here, the idea is to improve upon the predictions of the first tree.

Our new model is therefore Tree 1 + Tree 2. We then compute the classification error from this new 2-tree ensemble model and grow a third tree to predict the revised residuals. We repeat this process for a specified number of iterations. Subsequent trees help us to classify observations that are not well classified by the previous trees. Predictions of the final ensemble model is therefore the weighted sum of the predictions made by the previous tree models.

Benchmark and Metrics

The simple decision tree model will be used as benchmark. A RandomForestClassifier and a GradientBoostingClassifier's performance will be compared with the simple decision tree classifier. We usually use accuracy to measure classifier's performance. Here we will also use F1 score, since F1 score is a metric to balance recall & precision and to deal with imbalanced labels.

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

$$Precision = \frac{TruePositive}{TruePositive + FalsePositive}$$

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

$$Recall = \frac{TruePositive}{TruePositive + FalseNegative}$$

F1 score provides a better sense of model performance compared to purely accuracy as takes both false positives and false negatives in the calculation. With an uneven class distribution, F1 may usually be more useful than accuracy.

In this project , we don't really mind false positive, that we are fine to send the offer to a person who is unlikely to respond to the offer, but we do want to reduce the false negative so we don't want to miss potential sales. So in addition, we will look at F-beta score as we don't really care about precision, but care very much about recall.

$$F_\beta = (1 + \beta^2) \cdot \frac{\text{precision} \cdot \text{recall}}{(\beta^2 \cdot \text{precision}) + \text{recall}}$$

Model Implementation

I used 9 feature variables and target variable is responsive.

- Membership_days
- age
- income
- offer_reward
- Duration
- difficulty
- gender
- Membership_year
- Channels

I constructed a DecisionTreeClassifier, a RandomForestClassifier and a GradientBoostingClassifier. I plan to analyze feature importance to determine the drivers of an effective offer, a decision tree would provide good interpretability for analysis.

I performed the following steps for modeling:

- define target and feature variables

- split to train and test data
- apply feature scaler

Here's the performance metrics:

	model_name	training_accuracy	test_accuracy	F1_score	F-beta
0	DecisionTreeClassifier	0.9628	0.6307	0.5597	0.5539
1	RandomForestClassifier	0.9455	0.6585	0.5744	0.5534
2	GradientBoostingClassifier	0.6987	0.6943	0.5958	0.5532

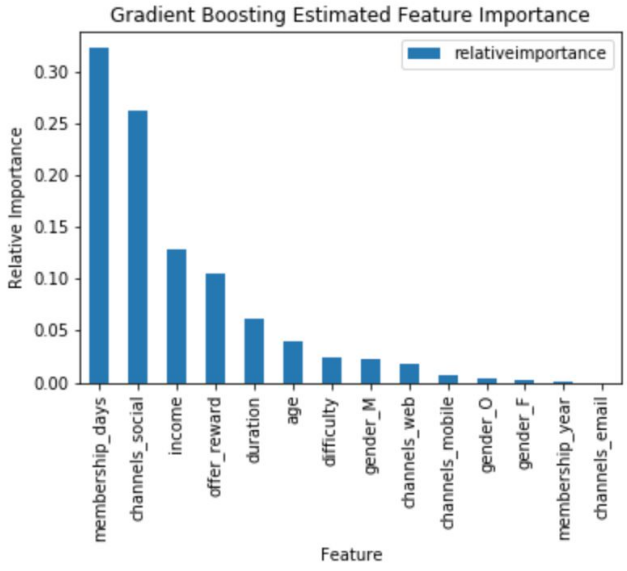
We can see that GradientBoostingClassifier has both higher accuracy and F1 score than RandomForestClassifier or the benchmark DecisionTreeClassifier.

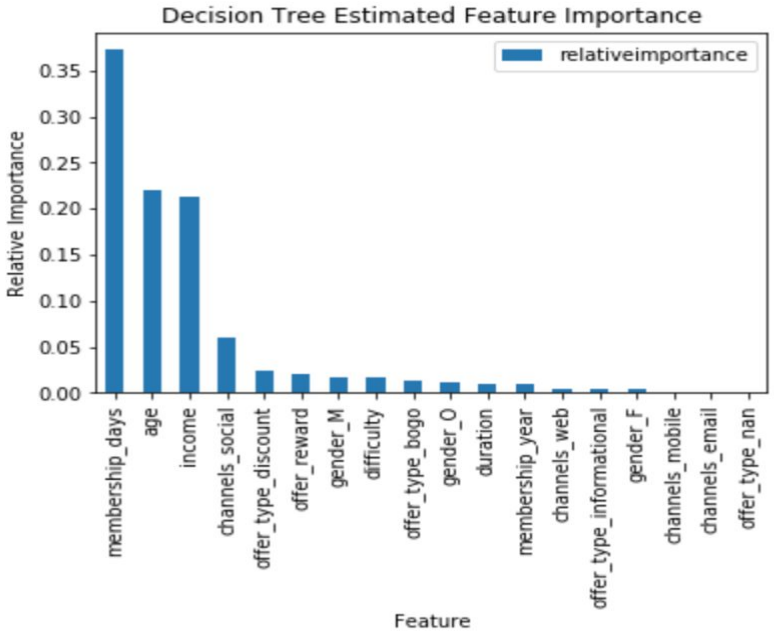
For all three models, the F1 scores are lower than the accuracy which may be due to the imbalance of classes. But the overall higher accuracy compared to F1 score indicates that the model is predicting the positive case more accurately compared to predicting the negative cases. Since I set beta to 2, I am weighing more on recall. F-beta score for the three models are very close.

Also DecisionTreeClassifier and RandomForestClassifier are overfitting. They have very high accuracy on the training data but poor generalization to test data. For GradientBoostingClassifier, train accuracy and test accuracy are very close. So it's very reasonable to choose GradientBoostingClassifier as the best classifier.

Feature Importance

One of the problems we are trying to solve is to discover the drivers of an effective offer. We can check the feature importances for the models.





Looking at the feature importance, we can see that the most important feature in both models are the tenure of membership. I am not surprised to see Income is the 3rd important feature. I don't quite understand why the channels_social is the second important feature for GradientBoostingClassifier. In DecisionTreeClassifier, channels_social is the fourth, not much influence.

Refinement

I first tried parameter tuning for the model, before experimenting with adding features to improve model performance. I conducted Randomized Search on the GradientBoostingClassifier to find the best parameters. Here are the hyperparameters settings:

- tree-specific parameters:
    - min_samples_split: [2, 5, 10]
    - min_samples_leaf : [1, 2, 4]
- Boosting parameters:
    - learning_rate: [0.1, 0.01, 0.001],
    - n_estimators: [10, 30, 50, 100, 150, 200, 250, 300],
- Miscellaneous parameters:
    - loss: ['deviance', 'exponential'],
    - random_state: 10

The optimal values are :
{'n_estimators': 300, 'min_samples_split': 2, 'min_samples_leaf': 4, 'loss': 'deviance', 'learning_rate': 0.1}

The accuracy for the best GradientBoostingClassifier increased slightly - from 69.43% to 69.93%, and the F1 score increased from 59.58% to 61.52%. F-beta score increased from 55.32% to 56.68%. There is not much performance increase by tuning the parameters.

Feature Engineering

When I preprocess the transaction data above, I have created a new response_rate column. It's actually the success rate, i.e. the total number of responded offers divided by the total number of received offers for each person. We will add this feature to the model and I believe it will largely increase the classifier's performance.

	model_name	training_accuracy	test_accuracy	F1_score	F-beta
0	DecisionTreeClassifier	0.9628	0.9103	0.8941	0.8883
1	RandomForestClassifier	0.9624	0.9120	0.8973	0.9004
2	GradientBoostingClassifier	0.9524	0.9428	0.9336	0.9423

Apparently that adding response rate feature drastically improves all models' performance. The accuracy for the GradientBoostingClassifier increased greatly - from 69.43% to 94.28%, and the F1 score increased from 59.58% to 93.36%. F-beta score increased from 56.68% to 94.23%. GradientBoostingClassifier still is the best performing model comparing with the benchmark DecisionTreeClassifier.

Conclusion

In this project, through data exploration and model construction, I was able to solve the two problems stated at the beginning of the report.

1. Discover the main drivers of offer effectiveness:

I found that the tenure of a member is the biggest predictor of the effectiveness of an offer. Income is the third one. I am surprised to see that social channel is on the top 5 even though Web and Email channels' response rate are much higher than social.

2. Explore if we can predict whether a customer will respond to an offer:

I constructed three models: DecisionTreeClassifier, RandomForstClassifier and GradientBoostClassifier to predict whether a customer will respond to an offer. GradientBoostClassifier performs the best. The accuracy is about 70%, F1-score of 60% and F-beta score of 67%. It’s a bit lower than my expectation. But I think it’s acceptable when only demographic data is provided.

I created a response_rate feature which greatly improves the classifier's performance. The accuracy is 94%, F1-score of 93% and F-beta score of 94%. But the response_rate feature may not be available for new customers who has only demographic information and doesn't have much offer related transaction data.

Future Enhancements:

1. I only constructed one model to predict whether a customer will respond to an offer. I could construct 3 different models for the 3 offer types: BOGO, Discount and Informational to get better results.

2. For convenience, I drop all the NaNs. In reality, there will be many missing data that we simply can not drop. In the future,I can use different techniques to fillna, such as using mean, median, etc.
3. Given more time and data, I could construct regression model to predict how much someone will spend based on demographics and offer type
4. A predictor can be build so that when being provided with the customer information, the prediction of response/transaction amount can be made. Also a web application can be built using this predictor.