Data Science Nanodegree Capstone
Udacity

Jeremy Harris
Dec 23rd, 2018

Rewards App Optimization

Project Overview

Customer rewards programs are meant to increase customer spending and thus increase revenue. Bringing a customer into a business increases the likelihood of them spending more than the initial reward. Some customers only spend enough to get the reward and nothing more. Other customers spend regardless of a reward and may not even know there was a reward in the first place. If the customer didn't know there was a reward and they complete it then the reward is not the driving factor behind the revenue and could be considered a loss, because it didn't initiate any behavior.

The data set provided is fake generated data, but it helps to simulation real-world scenarios. This sample data can be used to build prediction models around which customer should not be offered a reward because it was not a behavior cause.

The sample data consists of customer profiles, e.g. income, gender, enrollment date; offers, e.g. discount/informational, duration, delivery method, etc.; and transactions, e.g. purchases, offers received/viewed/completed, rewards earned.

Problem Statement

A method to reduce reward offers when unnecessary is needed. Unnecessary offers reduce company revenue which is not the intent. An offer should increase revenue by driving additional sales.

To answer the problem statement, I need to know, of the existing customers, which ones shouldn't be sent which offers. At its simplest, I need to find which customers were given offers and didn't view them but still completed the offer. This defines a set of customers that were going to make purchases regardless of an offer being extended. This is a logistic regression problem, because result is binary, yes/no.

I will explore both Logistic Regression and Classification models and see how well they perform. All models will be scored using cleaned data both with and without scaling.

Metrics

The output of the analysis is a binary answer to the question: Should this customer be sent a reward offer? The model results will be graded using both a confusion matrix, a ROC curve and a classification report.

The number of customers that shouldn't be given offers is very low compared to the number that should. This means that simply sending all customers an offer would result in a high accuracy score but does not solve the problem statement. This requires many different metrics to analyze various models for feasibility.

Confusion Matrix

The confusion matrix shows the number of true positives, false positives, true negatives and false negatives. This chart is a good visual representation of what the model outputs.

Recall (Sensitivity)

This value shows the true positive rate of the model. I want a model that has the best recall rate which should minimize false negatives.

Analysis

Data Preprocessing

The sample data contained three data sets: portfolio, profile, and transcript. Each data set went through a cleansing stage before being used in the analysis.

Portfolio

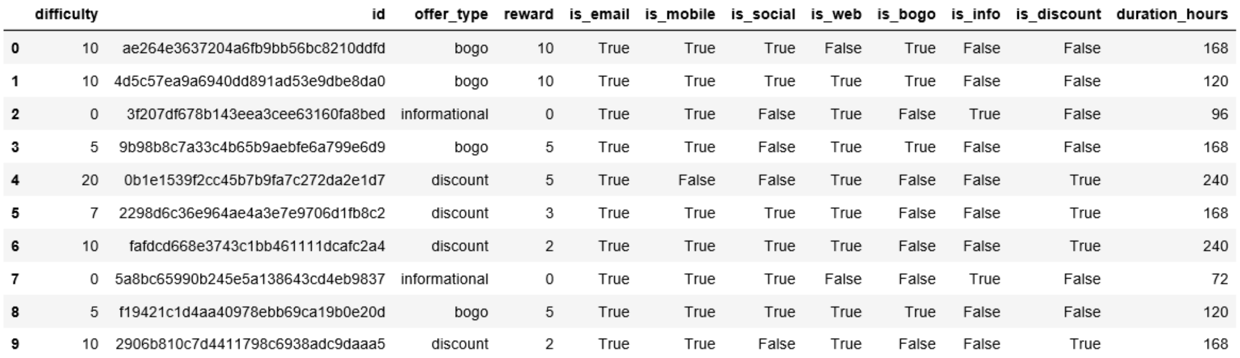
This contains the offers that were made to customers, totaling 10 offers. An offer can be informational, a discount, or buy-one-get-one (BOGO). Each offer has a duration and reward amount that would be given to a customer should it be completed within the timeframe.

As provided in the instruction set, informational offers do not have a reward but should be viewed as the impact of reading the information could impact purchase decisions for the duration indicated.

The data was enhanced as follows:

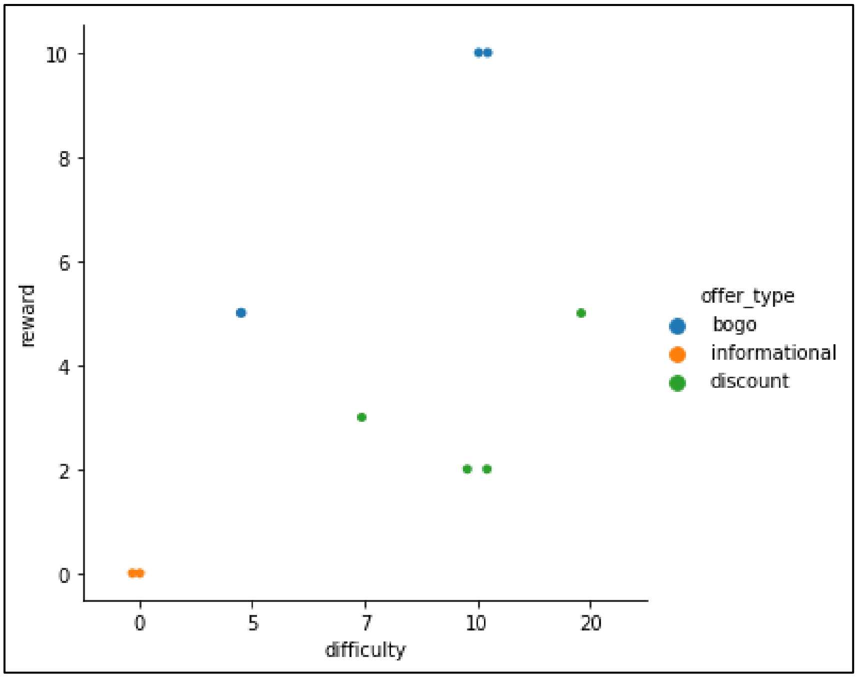
- Parse the delivery method (channel) into individual values. Raw data was in list format.
  - Email, mobile, social, web
- Add Boolean indicators for each offer type.
  - BOGO, informational, discount
- Convert duration from days to hours

While the delivery method (channel) of the offers could have valuable information, none of the offers used only a single method. Without individual delivery methods I was not able to decide if one method was better than another for certain customers.



	difficulty	id	offer_type	reward	is_email	is_mobile	is_social	is_web	is_bogo	is_info	is_discount	duration_hours
0	10	ae264e3637204a6fb9bb56bc8210ddfd	bogo	10	True	True	True	False	True	False	False	168
1	10	4d5c57ea9a6940dd891ad53e9dbe8da0	bogo	10	True	True	True	True	True	False	False	120
2	0	3f207df678b143eea3cee63160fa8bed	informational	0	True	True	False	True	False	True	False	96
3	5	9b98b8c7a33c4b65b9aebfe6a799e6d9	bogo	5	True	True	False	True	True	False	False	168
4	20	0b1e1539f2cc45b7b9fa7c272da2e1d7	discount	5	True	False	False	True	False	False	True	240
5	7	2298d6c36e964ae4a3e7e9706d1fb8c2	discount	3	True	True	True	True	False	False	True	168
6	10	fafdcd668e3743c1bb461111dcafc2a4	discount	2	True	True	True	True	False	False	True	240
7	0	5a8bc65990b245e5a138643cd4eb9837	informational	0	True	True	True	False	False	True	False	72
8	5	f19421c1d4aa40978ebb69ca19b0e20d	bogo	5	True	True	True	True	True	False	False	120
9	10	2906b810c7d4411798c6938adc9daaa5	discount	2	True	True	False	True	False	False	True	168

Generally, as the difficulty increases so does the reward, which would be expected. As a customer I would be discouraged to complete an offer if the challenge was greater than the reward.



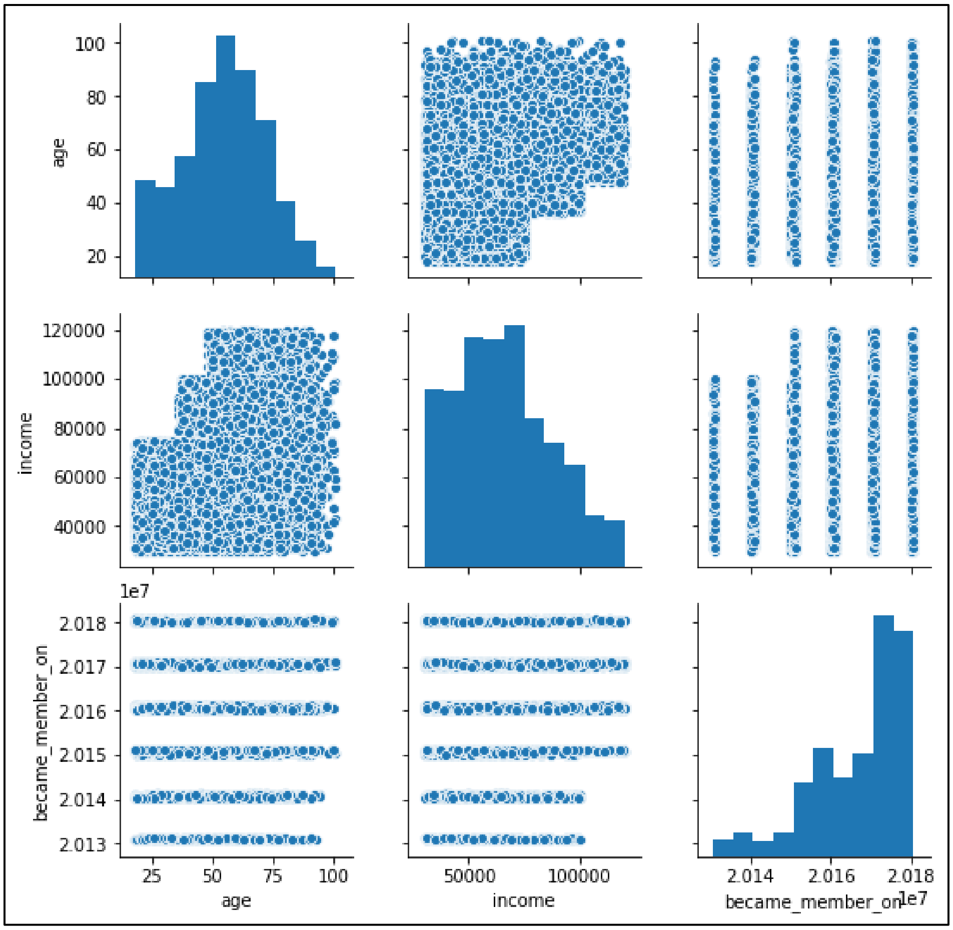
Profile

This is the customer information; age, income, gender and join date. If this were actual customer data, it is likely provided by the customer and thus cannot be considered completely accurate. A customer can list any income and age or choose to provide nothing at all.

Of the 17,000 records there were no duplicate customers, using the id provided, in the data. This meant no de-duplication was necessary.

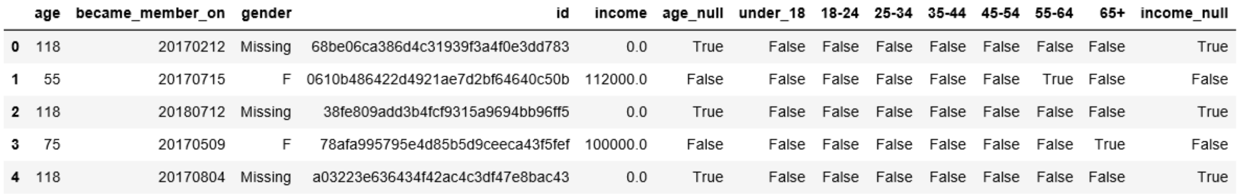
If age was null than gender and income were also null. There were no cases where just one field was null. An age of 118 means no value was provided, i.e. null.

Below is the pair distribution of the raw data integer-based data. Notice that income increases with age



Cleaning the profile data was a bit more involved. I decided on these steps:

1. Add a new field called age_null that would be set to true if age = 118
2. Create age brackets
   a. Under 18, 18-24, 25-34, 35-44, 45-54, 55-64, 65+
3. Add income_null set to True when income was null and set income to 0.
4. Set null gender values to 'Missing'.



	age	became_member_on	gender	id	income	age_null	under_18	18-24	25-34	35-44	45-54	55-64	65+	income_null
0	118	20170212	Missing	68be06ca386d4c31939f3a4f0e3dd783	0.0	True	False	False	False	False	False	False	False	True
1	55	20170715	F	0610b486422d4921ae7d2bf64640c50b	112000.0	False	False	False	False	False	False	True	False	False
2	118	20180712	Missing	38fe809add3b4fcf9315a9694bb96ff5	0.0	True	False	False	False	False	False	False	False	True
3	75	20170509	F	78afa995795e4d85b5d9ceeca43f5fef	100000.0	False	False	False	False	False	False	False	True	False
4	118	20170804	Missing	a03223e636434f42ac4c3df47e8bac43	0.0	True	False	False	False	False	False	False	False	True

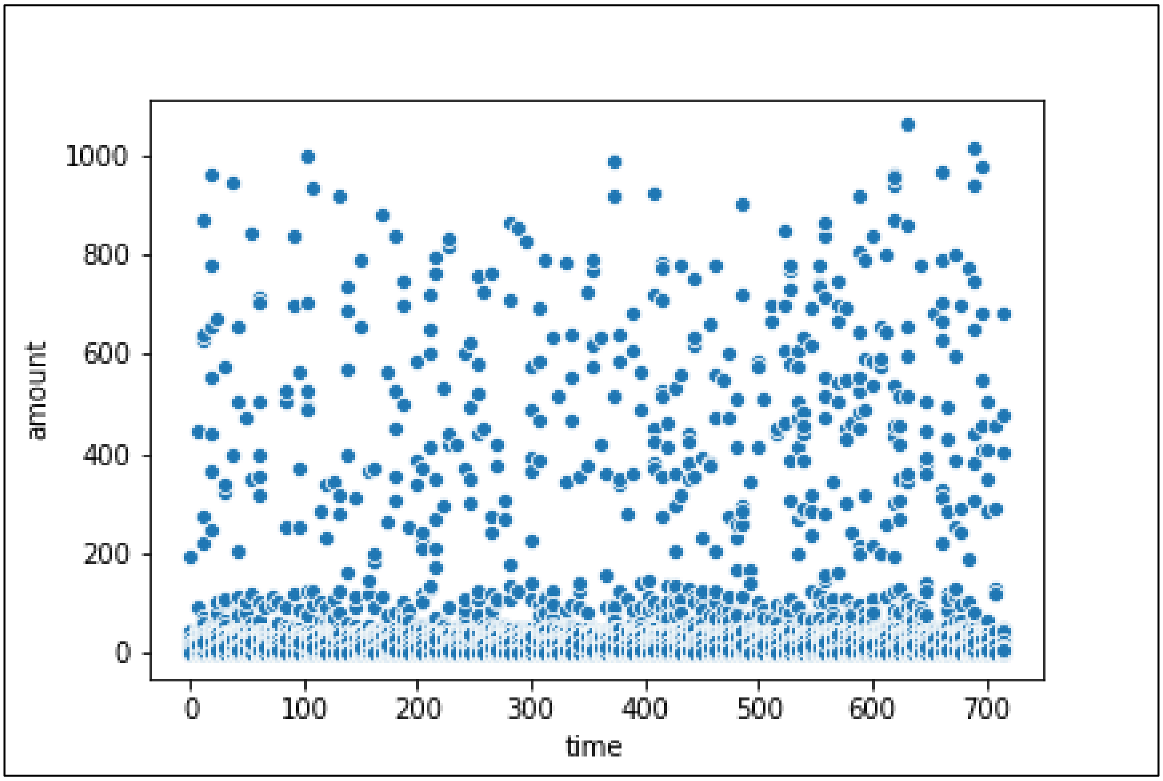
Transcript

This data set contains all the transactions a customer made and when, totaling 306,534 transactions. When they received offers, when they were viewed, when they were completed, and what rewards were earned. The events could contain multiple pieces of information. Specifically, if an offer was completed it's a single event but it contains two values, which offer_id and how much reward. I parsed this data to have one column per event metric, which splits offer_id and reward into their own columns. This makes the data easier to work with but does include columns that may not apply to an event.

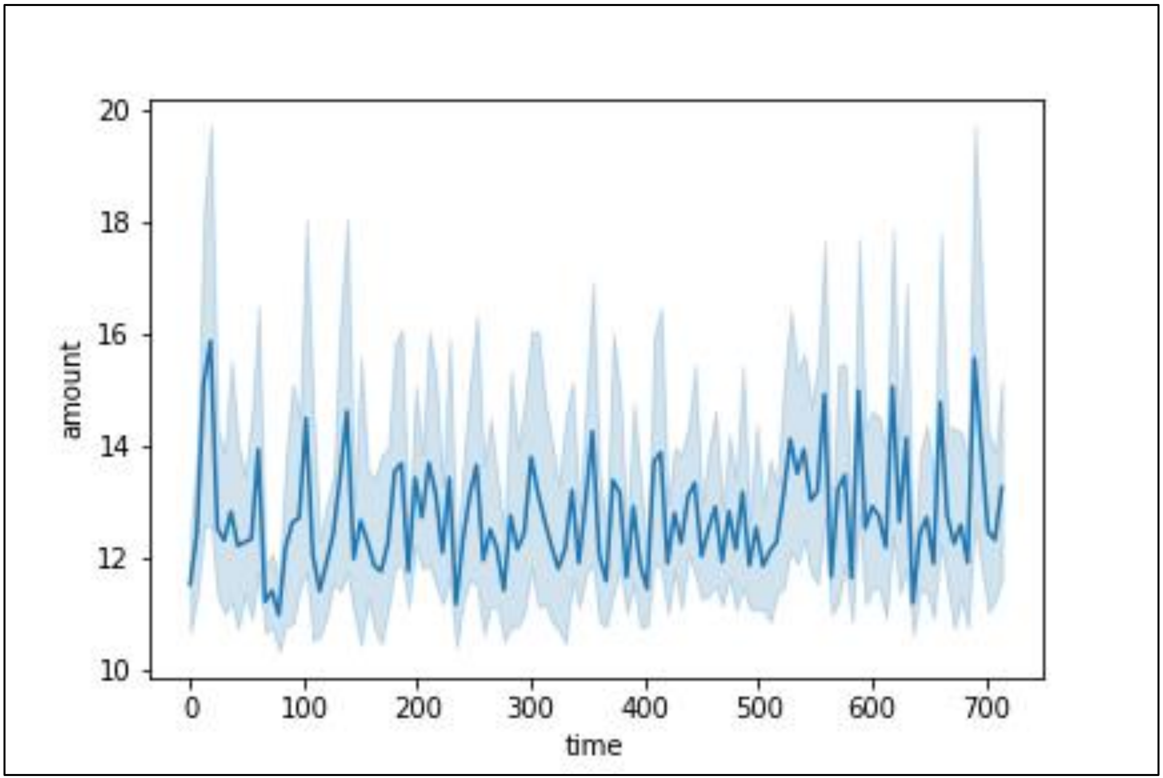
For example, when an offer is received, there is no reward, so it would always be null. Another example is that transactions are unrelated to offers, so they would always have a null value for offer_id and reward. Just something that needs to be kept in mind when working with the data.

I also standardized the event names since some had underscores, and others had spaces. This is the largest dataset but required minimal transformation.

The amount of each transaction seems consistent over the dataset, which span 714 hours. As you can see, most of the transactions are less than 100 but there are some that exceed 1000.



Looking at this another way we can see the transaction amounts aggregated by time. The dark line shows the central tendency and the lighter shading is the confidence interval.



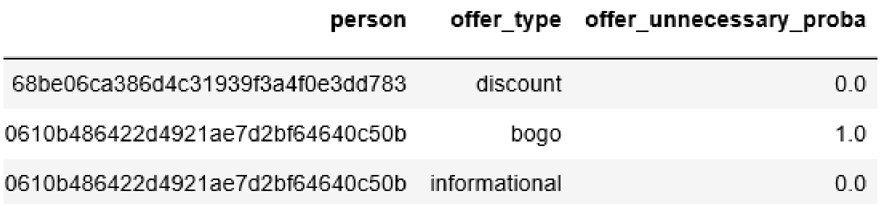
Data Exploration

The challenge is that I want this to be extensible to new customers, the cold start problem. New customers don't have any purchase history to build metrics with. This means that the input to a logistic regression model should only require the basic information that we get from the customer when they sign up for the program. As they make purchases and engage with the rewards program their behavior will change the models and determine whether they start receiving specific offers to drive behavior.

Using the historical transactions for each customer I can build this dataset. For all transactions I can aggregate the data to find offer_recieved, offer_viewed, and offer_completed for every offer made to each customer. Using this base data, I can add a new column for offer_unnecessary when a customer completes but does not view the offer. Now I know whether each offer a customer received was necessary.

A customer can also receive the same type of offer multiple times, i.e. customer receives more than one discount offer. This results in further aggregation by offer type which transforms the offer_unnecessary from a Boolean to a probability. If the customer was given 2 offers and 1 of them was unnecessary then the offer_unnecessary_proba value is .5 or 50%. Notice that this does not apply to informational. These offers will always show 0% because they cannot be completed. Analysis on customer behavior is outside the scope of this effort.

If the customer didn't receive any offers, they are dropped from the final data set. The final data set consists of 36,300 records and looks like this, for example:

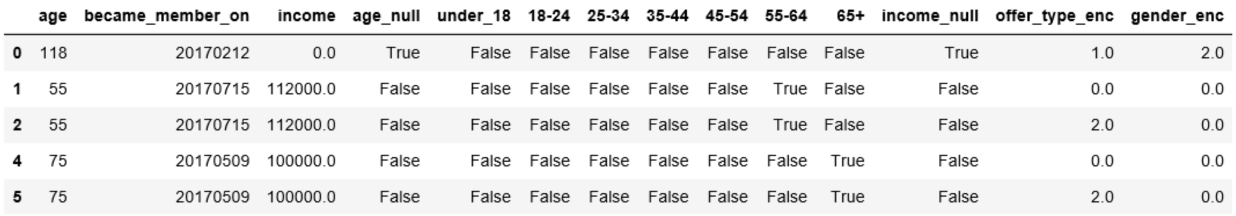


person	offer_type	offer_unnecessary_proba
68be06ca386d4c31939f3a4f0e3dd783	discount	0.0
0610b486422d4921ae7d2bf64640c50b	bogo	1.0
0610b486422d4921ae7d2bf64640c50b	informational	0.0

At this point a decision needs to be made. What is the cutoff for when an offer is deemed unnecessary? Is it 50%, 80% or something else? For this analysis I chose 80%.

For input into the models I encoded offer_type and gender using ordinal encoding. Then excluded the original offer_type and gender fields.

My model input dataset (X) is as follows:



	age	became_member_on	income	age_null	under_18	18-24	25-34	35-44	45-54	55-64	65+	income_null	offer_type_enc	gender_enc
0	118	20170212	0.0	True	False	False	False	False	False	False	False	True	1.0	2.0
1	55	20170715	112000.0	False	False	False	False	False	False	True	False	False	0.0	0.0
2	55	20170715	112000.0	False	False	False	False	False	False	True	False	False	2.0	0.0
4	75	20170509	100000.0	False	False	False	False	False	False	False	True	False	0.0	0.0
5	75	20170509	100000.0	False	False	False	False	False	False	False	True	False	2.0	0.0

Results

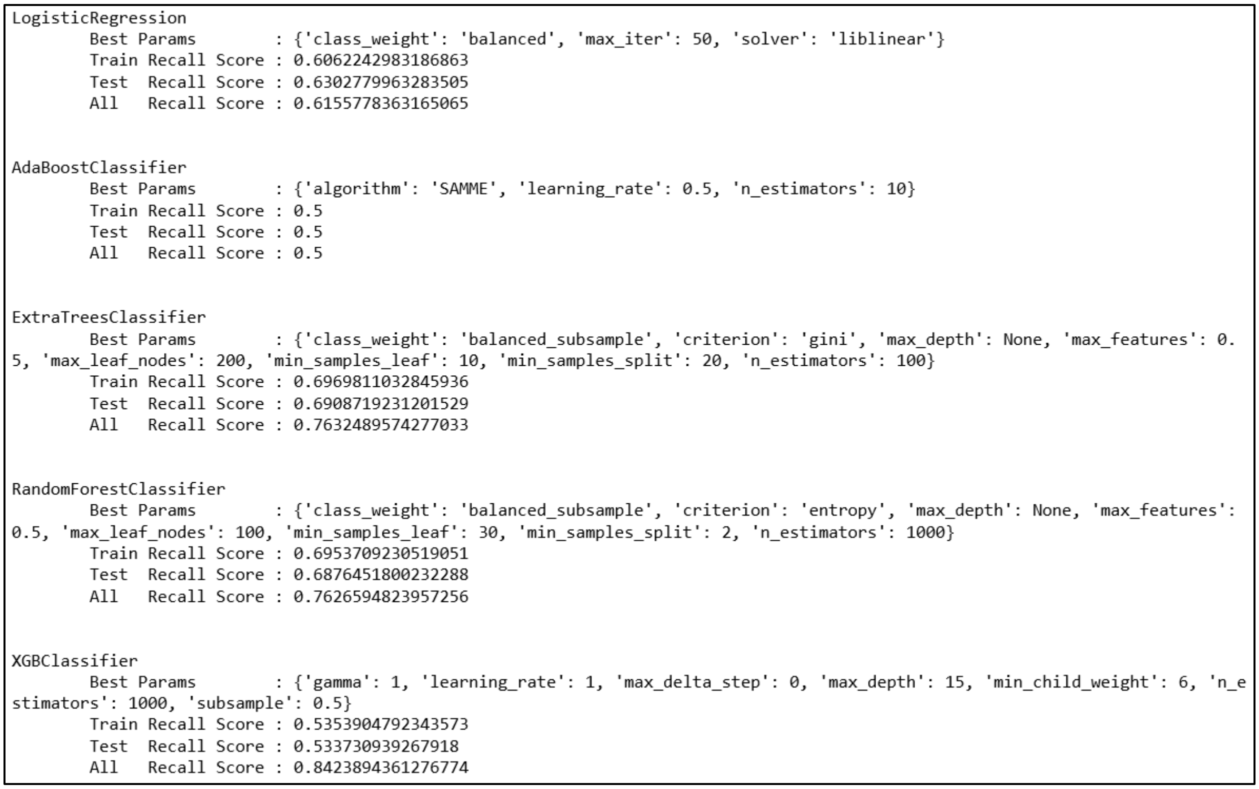
Model Evaluation and Validation

I split the above dataset into training and testing, 80% and 20%, respectively. The testing data will be held out to use on the final models.

I ran the training data through 5 different models using GridSearchCV to tune each one. I set GridSearchCV to use a custom scorer based on the recall score. This will rank the models by the recall score instead of the default score mechanism. I also used 5-fold cross-validation on the training dataset.

I initially included GradientBoostingClassifier but the training time was extremely long. If the model takes too long to train in a test environment, then it would probably not be a good fit for production if the model is retraining itself automatically. Chosen models must be performant.

Below is the outcome of the model training. Even though XGBClassifier has a higher overall score, I would choose RandomForestClassifier because it had the best score against the test data.

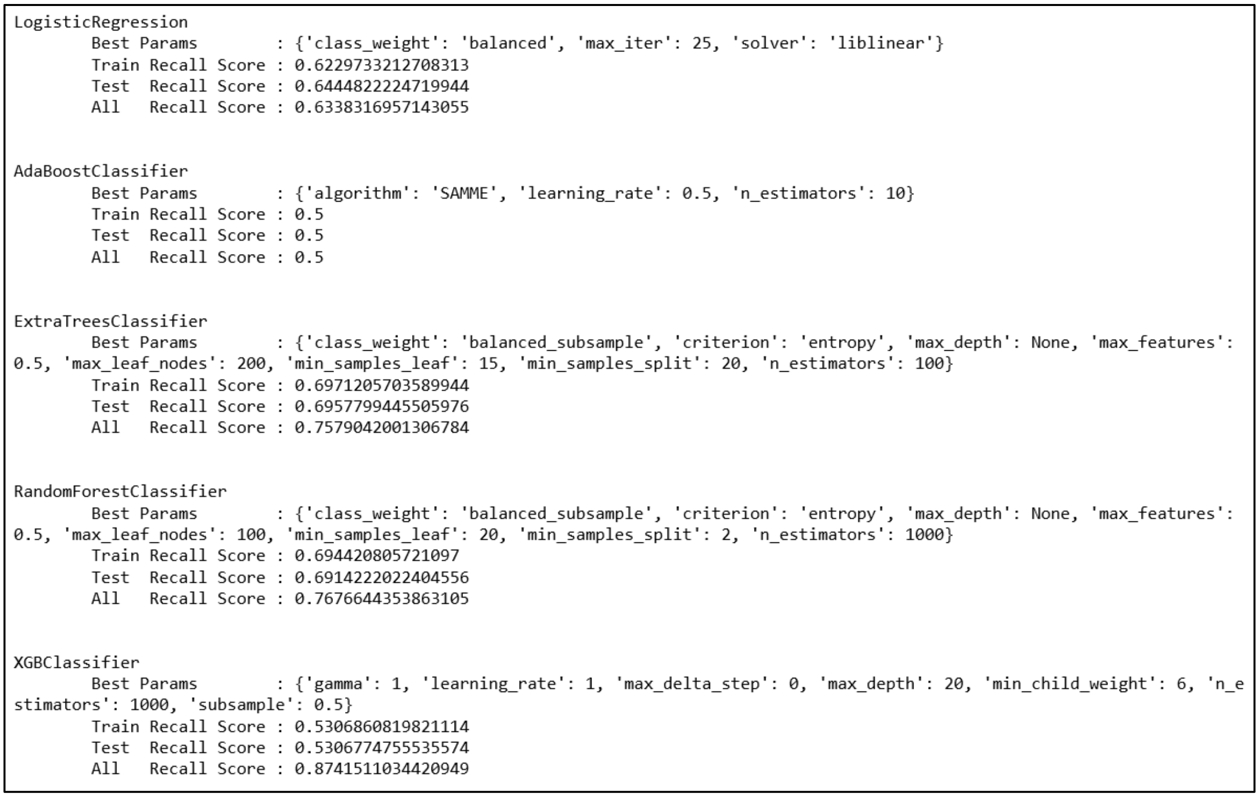


```
LogisticRegression
        Best Params       : {'class_weight': 'balanced', 'max_iter': 50, 'solver': 'liblinear'}
        Train Recall Score : 0.6062242983186863
        Test  Recall Score : 0.6302779963283505
        All   Recall Score : 0.6155778363165065


AdaBoostClassifier
        Best Params       : {'algorithm': 'SAMME', 'learning_rate': 0.5, 'n_estimators': 10}
        Train Recall Score : 0.5
        Test  Recall Score : 0.5
        All   Recall Score : 0.5


ExtraTreesClassifier
        Best Params       : {'class_weight': 'balanced_subsample', 'criterion': 'gini', 'max_depth': None, 'max_features': 0.
5, 'max_leaf_nodes': 200, 'min_samples_leaf': 10, 'min_samples_split': 20, 'n_estimators': 100}
        Train Recall Score : 0.6969811032845936
        Test  Recall Score : 0.6908719231201529
        All   Recall Score : 0.7632489574277033


RandomForestClassifier
        Best Params       : {'class_weight': 'balanced_subsample', 'criterion': 'entropy', 'max_depth': None, 'max_features':
0.5, 'max_leaf_nodes': 100, 'min_samples_leaf': 30, 'min_samples_split': 2, 'n_estimators': 1000}
        Train Recall Score : 0.6953709230519051
        Test  Recall Score : 0.6876451800232288
        All   Recall Score : 0.7626594823957256


XGBClassifier
        Best Params       : {'gamma': 1, 'learning_rate': 1, 'max_delta_step': 0, 'max_depth': 15, 'min_child_weight': 6, 'n_e
stimators': 1000, 'subsample': 0.5}
        Train Recall Score : 0.5353904792343573
        Test  Recall Score : 0.533730939267918
        All   Recall Score : 0.8423894361276774
```

Expanding on the above model I also ran the data through a scaler to see if it would provide any improvements. Specifically, I chose MinMaxScaler, which translates the data to values between 0 and 1 based on the minimum and maximum value of each feature. When the data isn't on the same scale it can degrade predictive performance.

As you can see, there was an improvement in most of the models. You'll also see that by scaling the data the parameters used in the best model may have changed, e.g. LogisticRegression changed the max_iter parameter from 50 to 25.

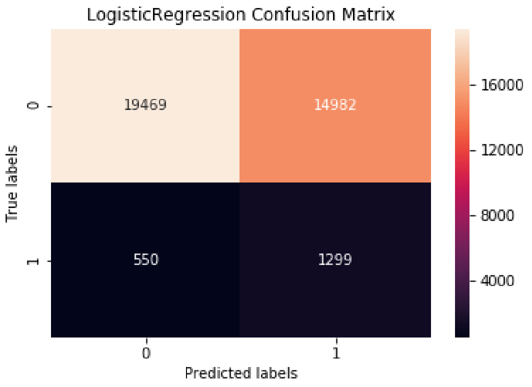
From these results I would change to the ExtraTreesClassifier model, since it has the best recall score on the test data, albeit only slightly.

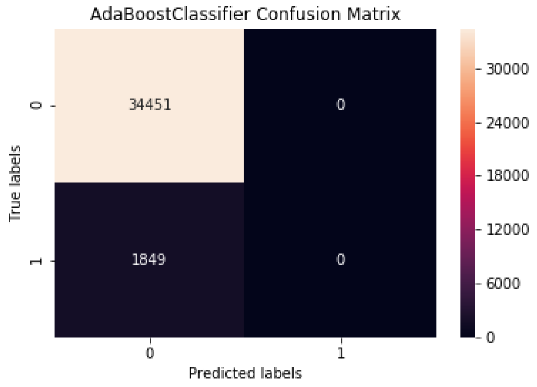


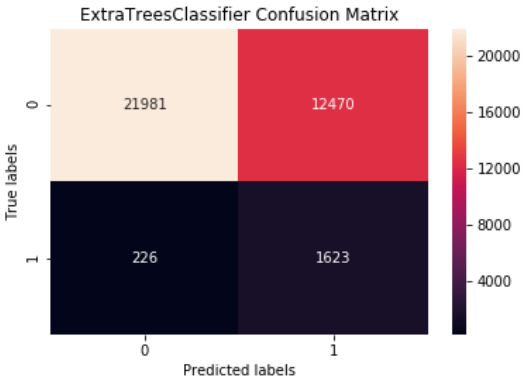
```
LogisticRegression
        Best Params       : {'class_weight': 'balanced', 'max_iter': 25, 'solver': 'liblinear'}
        Train Recall Score : 0.6229733212708313
        Test  Recall Score : 0.6444822224719944
        All   Recall Score : 0.6338316957143055


AdaBoostClassifier
        Best Params       : {'algorithm': 'SAMME', 'learning_rate': 0.5, 'n_estimators': 10}
        Train Recall Score : 0.5
        Test  Recall Score : 0.5
        All   Recall Score : 0.5


ExtraTreesClassifier
        Best Params       : {'class_weight': 'balanced_subsample', 'criterion': 'entropy', 'max_depth': None, 'max_features':
0.5, 'max_leaf_nodes': 200, 'min_samples_leaf': 15, 'min_samples_split': 20, 'n_estimators': 100}
        Train Recall Score : 0.6971205703589944
        Test  Recall Score : 0.6957799445505976
        All   Recall Score : 0.7579042001306784


RandomForestClassifier
        Best Params       : {'class_weight': 'balanced_subsample', 'criterion': 'entropy', 'max_depth': None, 'max_features':
0.5, 'max_leaf_nodes': 100, 'min_samples_leaf': 20, 'min_samples_split': 2, 'n_estimators': 1000}
        Train Recall Score : 0.694420805721097
        Test  Recall Score : 0.6914222022404556
        All   Recall Score : 0.7676644353863105


XGBClassifier
        Best Params       : {'gamma': 1, 'learning_rate': 1, 'max_delta_step': 0, 'max_depth': 20, 'min_child_weight': 6, 'n_e
stimators': 1000, 'subsample': 0.5}
        Train Recall Score : 0.5306860819821114
        Test  Recall Score : 0.5306774755535574
        All   Recall Score : 0.8741511034420949
```

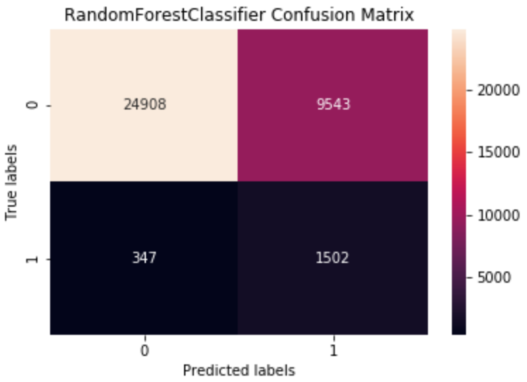
Looking at these models another way, using the scaled model results, I can use a confusion matrix to visually see the ratio of predictions, i.e. true positive, false positive, true negative, and false negative. These are based on predictions across all the data for each model.

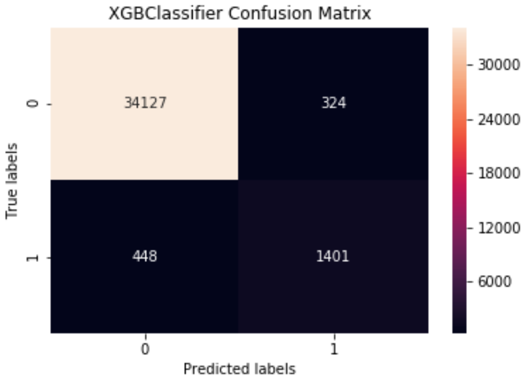
If I were choosing a model just based on these metrics, I would be inclined to pick XGBClassifier since it has the lowest false-positives and false-negatives.







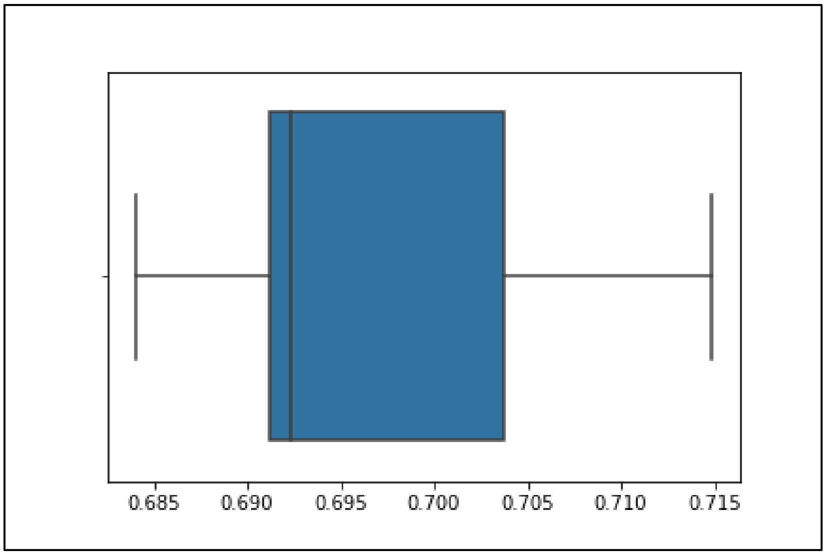




Looking at three different metrics I came away with 3 different models that I would have used. Overall, I would tend to stick with my first finding and use RandomForestClassifier. This model had good performance without scaling which means one less step in the final pipeline.

Using the best parameters from RandomForestClassifier I ran it though 10 iterations of 5-fold cross validation, with a different random seed for each iteration. Because I'm using a binary classifier the model uses StratifiedKFold by default.

Using the recall score from all 50 cross validated iterations I can see if the model maintains the same score I found in the initial training/testing. This is best shown using a boxplot, below. The median across all 50 iterations is 0.691, which is right in between the test and training recall scores.



Conclusion

Reflection

Overall, using predictive modeling I was able to correctly identify whether a customer should be sent an offer based only on their initial profile about 70% of the time. This method would require retraining as more customers sign-up for the program and make transactions.

I started down the path of ROI based on false positives and false negatives but decided it was better having an initial starting point, as this analysis discusses. I found it very fascinating that with just a relatively few features I was able to expand out into all sorts of insights, most of which aren't included here for brevity.

Improvement

There are several areas of improvement that can be made to this analysis.

First, additional models could be added to the analysis, including chaining multiple models together. Try using a deep learning model that can retrain itself as more data comes into the system.

Second, more model parameters could be evaluated. I was hardware limited in this analysis which meant a lot of time was spent waiting for the model tuning to finish.

Third, add metrics that show the monetary cost of incorrect predictions. It's obvious that sending out unnecessary offers costs the company money. But subtler, is the change to revenue if the customer would have never received an offer but should have. Offers tend to increase customer spending, which means if they didn't get an offer they should have received, its lost revenue. A model chosen by this method would heavily penalize incorrect predictions, which could change the model used.