Starbucks Capstone Project

1. Introduction

This Starbucks Capstone Project is Udacity DS Nanodegree. Udacity partnered with Starbucks to provide a real-world business problem and simulated data mimicking their customer behavior. The full source code can be found in my github.

2. Domain background

Starbucks Corporation is an American coffee company and operates over 30,000 locations worldwide. Once every few days, Starbucks sends out an offer to users of the mobile app. An offer can be merely an advertisement for a drink or an actual offer such as a discount or BOGO (buy one get one free). Some users might not receive any offer during certain weeks. The company would like to select customers with the highest probability of making a purchase, in contrast, want to know the customers who don't like to receive offers too. In other words, the right offer to the right customer. This project can be handled as a classifical supervising learning problem, so we have our features about the customers, the offers and the transcripts (as we will see in the dataset section) and our label will be if the offer was completed or not (binary classification). Maybe this label column needs to be created after the analysis of all data. My motivation for this project is because here in my city we don't have Starbucks and when I travel I need to go there since I really enjoy their products. So, it would be nice to work with a dataset's company that I really like and I don't have in my day-to-day.

3. Problem Statement

Building a model that predicts whether or not someone will complete an offer based on given demographics and offer type. Thus there will be two possible outputs from the model, making this problem as a supervising learning problem, more specifically, binary classification problem:

- The customer complete the offer (Yes / 1)
- The customer will not complete the offer (No / 0)

4. Dataset and Inputs

4.1 Dataset overview:

- The program used to create the data simulates how people make purchasing decisions and how those decisions are influenced by promotional offers.
- Each person in the simulation has some hidden traits that influence their purchasing patterns and are associated with their observable traits. People produce various events, including receiving offers, opening offers, and making purchases.
- As a simplification, there are no explicit products to track. Only the amounts of each transaction or offer are recorded.
- There are three types of offers that can be sent: buy-one-get-one (BOGO), discount, and informational. In a BOGO offer, a user needs to spend a certain amount to get a reward equal to that threshold amount. In a discount, a user gains a reward equal to a fraction of the amount spent. In an informational offer, there is no reward, but neither is there a requisite amount that the user is expected to spend. Offers can be delivered via multiple channels.

- The basic task is to use the data to identify which groups of people are most responsive to each type of offer, and how best to present each type of offer.

A customer can interact with an offer by receiving it, viewing it or completing it. It is possible for a customer to complete some offers without viewing them. After the preprocessing step with all the exploratory data analysis (EDA), I will combine all the data in an unique data cleaned to use for input to the models. To split into training and testing sets I will use a 70/30 split.

4.2 Data Dictionary:

Profile.json: Rewards program users (17000 users x 5 fields)

- gender: (categorical) M, F, O, or null
- age: (numeric) missing value encoded as 118
- id: (string/hash)
- became_member_on: (date) format YYYYMMDD
- income: (numeric)

```
Profile shape: 17000 rows 5 columns
```

	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
3	F	75	78afa995795e4d85b5d9ceeca43f5fef	20170509	100000.0
4	None	118	a03223e636434f42ac4c3df47e8bac43	20170804	NaN

Portfolio.json: Offers sent during 30-day test period (10 offers x 6 fields)

- reward: (numeric) money awarded for the amount spent
- channels: (list) web, email, mobile, social
- difficulty: (numeric) money required to be spent to receive reward
- duration: (numeric) time for offer to be open, in days
- offer_type: (string) bogo, discount, informational
- id: (string/hash)

Portfolio shape: 10 rows 6 columns

	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
3	5	[web, email, mobile]	5	7	bogo	9b98b8c7a33c4b65b9aebfe6a799e6d9
4	5	[web, email]	20	10	discount	0b1e1539f2cc45b7b9fa7c272da2e1d7
5	3	[web, email, mobile, social]	7	7	discount	2298d6c36e964ae4a3e7e9706d1fb8c2
6	2	[web, email, mobile, social]	10	10	discount	fafdcd668e3743c1bb461111dcafc2a4
7	0	[email, mobile, social]	0	3	informational	5a8bc65990b245e5a138643cd4eb9837
8	5	[web, email, mobile, social]	5	5	bogo	f19421c1d4aa40978ebb69ca19b0e20d
9	2	[web, email, mobile]	10	7	discount	2906b810c7d4411798c6938adc9daaa5

Transcript.json: Event log (306648 events x 4 fields)

- person: (string/hash)
- event: (string) offer received, offer viewed, transaction, offer completed
- value: (dictionary) different values depending on event type
    - offer id: (string/hash) not associated with any "transaction"
    - amount: (numeric) money spent in "transaction"
    - reward: (numeric) money gained from "offer completed"
- time: (numeric) hours after start of test

Transcript shape: 306534 rows 4 columns

	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
3	8ec6ce2a7e7949b1bf142def7d0e0586	offer received	{'offer id': 'fafdcd668e3743c1bb461111dcafc2a4'}	0
4	68617ca6246f4fbc85e91a2a49552598	offer received	{'offer id': '4d5c57ea9a6940dd891ad53e9dbe8da0'}	0

Initial Analysis

1. **Portfolio**

Since the portfolio dataset only has 10 rows, we don't need to analyse too much because we already see all the original data. We can already clean this dataframe. The dataframe can be visualized on the dataset section.

2. **Profile**

First, I check if the shape is the same as the description. Then, I check if the ID column has only unique values. All Ok with the data.

So, I check the null values. Remember that in the age column, the value of 118 represents the null value. Since all null values are in the same rows, I removed them.

```
Check NaN values in each column
gender              2175
age                    0
id                     0
became_member_on       0
income              2175
dtype: int64
--------------------------------------------------
Gender nulls values: 2175
Age nulls(118) values: 2175
Income nulls values: 2175
--------------------------------------------------
Shape before drop NaN values: (17000, 5)
Shape after drop NaN values: (14825, 5)
```

I created two new columns called ‘year’ and ‘age_ranges’ from the ‘became_member_on’ and ‘age’ columns.

	gender	age	id	became_member_on	income	year	age_range
1	F	55	0610b486422d4921ae7d2bf64640c50b	20170715	112000.0	2017	[50, 60)
3	F	75	78afa995795e4d85b5d9ceeca43f5fef	20170509	100000.0	2017	[70, 80)
5	M	68	e2127556f4f64592b11af22de27a7932	20180426	70000.0	2018	[60, 70)
8	M	65	389bc3fa690240e798340f5a15918d5c	20180209	53000.0	2018	[60, 70)
12	M	58	2eeac8d8feae4a8cad5a6af0499a211d	20171111	51000.0	2017	[50, 60)

3. Transcript

I check the unique values in the ‘event’ column and count them.

```
transaction        138953
offer received      76277
offer viewed        57725
offer completed     33579
```

Then, I checked the types of keys in the ‘value’ column and I got: 'amount', 'offer id', 'offer_id' and 'reward'. That analysis was enough to start the cleaning.

Data Cleaning and Feature Engineering

1. Portfolio clean

- Rename the column id to offer_id
- Rename the column duration to duration_days
- One hot encoding offer_type column
- One hot encoding channels column

```
Portfolio shape before preprocessing: 10 rows 6 columns
Portfolio shape after preprocessing: 10 rows 11 columns
```

	reward	difficulty	duration_days	offer_id	bogo	discount	informational	email	mobile	social	web
0	10	10	7	ae264e3637204a6fb9bb56bc8210ddfd	1	0	0	1	1	1	0
1	10	10	5	4d5c57ea9a6940dd891ad53e9dbe8da0	1	0	0	1	1	1	1
2	0	0	4	3f207df678b143eea3cee63160fa8bed	0	0	1	1	1	0	1
3	5	5	7	9b98b8c7a33c4b65b9aebfe6a799e6d9	1	0	0	1	1	0	1
4	5	20	10	0b1e1539f2cc45b7b9fa7c272da2e1d7	0	1	0	1	0	0	1

2. Profile clean

- Rename the column id to customer_id
- One hot encoding age_range column
- One hot encoding gender column
- One hot encoding year column
- Drop columns age_range, gender, become_member_on, age and year

```
Profile shape before preprocessing: 14825 rows 7 columns
Profile shape after preprocessing: 14825 rows 21 columns
```

	customer_id	income	2013	2014	2015	2016	2017	2018	F	M	...	[10, 20)	[100, 110)	[20, 30)	[30, 40)	[40, 50)	[50, 60)	[60, 70)	[70, 80)	[80, 90)	[90, 100)
1	0610b486422d4921ae7d2bf64640c50b	112000.0	0	0	0	0	1	0	1	0	...	0	0	0	0	0	1	0	0	0	0
3	78afa995795e4d85b5d9ceeca43f5fef	100000.0	0	0	0	0	1	0	1	0	...	0	0	0	0	0	0	0	1	0	0
5	e2127556f4f64592b11af22de27a7932	70000.0	0	0	0	0	0	1	0	1	...	0	0	0	0	0	0	1	0	0	0
8	389bc3fa690240e798340f5a15918d5c	53000.0	0	0	0	0	0	1	0	1	...	0	0	0	0	0	0	1	0	0	0
12	2eac8d8feae4a8cad5a6af0499a211d	51000.0	0	0	0	0	1	0	0	1	...	0	0	0	0	0	1	0	0	0	0

3. Transcript

- Rename the column person to customer_id
- One hot encoding event column
- Create new columns: amount and offer_id from value column
- Drop transactions rows whose customer_id is not in profile dataframe
- Convert time column to days
- Separate offers data and transactions data
- Drop duplicates rows

```
Transactions df shape: 123957 rows 3 columns
Offers df shape: 148431 rows 6 columns
```

Transactions df:

	customer_id	time	amount
11027	02c083884c7d45b39cc68e1314fec56c	0.00	0.83
11030	9fa9ae8f57894cc9a3b8a9bbe0fc1b2f	0.00	34.56
11032	54890f68699049c2a04d415abc25e717	0.00	13.23
11040	b2f1cd155b864803ad8334cdf13c4bd2	0.00	19.51
11041	fe97aa22dd3e48c8b143116a8403dd52	0.00	18.97
...	...	...	...
272382	24f56b5e1849462093931b164eb803b5	29.75	22.64
272384	b3a1272bc9904337b331bf348c3e8c17	29.75	1.59
272385	68213b08d99a4ae1b0dcb72aebd9aa35	29.75	9.53
272386	a00058cf10334a308c68e7631c529907	29.75	3.61
272387	76ddbd6576844afe811f1a3c0fbb5bec	29.75	3.53

123957 rows × 3 columns

Offers df:

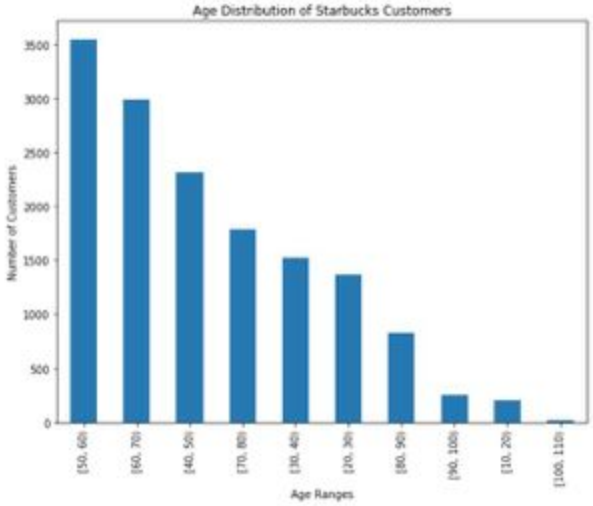
	customer_id	time	offer_id	offer_completed	offer_received	offer_viewed
0	78afa995795e4d85b5d9ceeca43f5fef	0.00	9b98b8c7a33c4b65b9aebfe6a799e6d9	0	1	0
1	e2127556f4f64592b11af22de27a7932	0.00	2906b810c7d4411798c6938adc9daaa5	0	1	0
2	389bc3fa690240e798340f5a15918d5c	0.00	f19421c1d4aa40978ebb69ca19b0e20d	0	1	0
3	2eeac8d8feae4a8cad5a6af0499a211d	0.00	3f207df678b143eea3cee63160fa8bed	0	1	0
4	aa4862eba776480b8bb9c68455b8c2e1	0.00	0b1e1539f2cc45b7b9fa7c272da2e1d7	0	1	0
...	...	...	...	...	...	...
272343	84fb57a7fe8045a8bff6236736ee73a0f	29.75	5a8bc65990b245e5a138643cd4eb9837	0	0	1
272350	abc4359eb34e4e2ca2349da2ddf771b6	29.75	3f207df678b143eea3cee63160fa8bed	0	0	1
272363	8dda575c2a1d44b9ac8e8b07b93d1f8e	29.75	0b1e1539f2cc45b7b9fa7c272da2e1d7	0	0	1
272365	8431c16f8e1d440880db371a68f82dd0	29.75	fafdcd668e3743c1bb461111dcafc2a4	1	0	0
272383	24f56b5e1849462093931b164eb803b5	29.75	fafdcd668e3743c1bb461111dcafc2a4	1	0	0

148431 rows × 6 columns

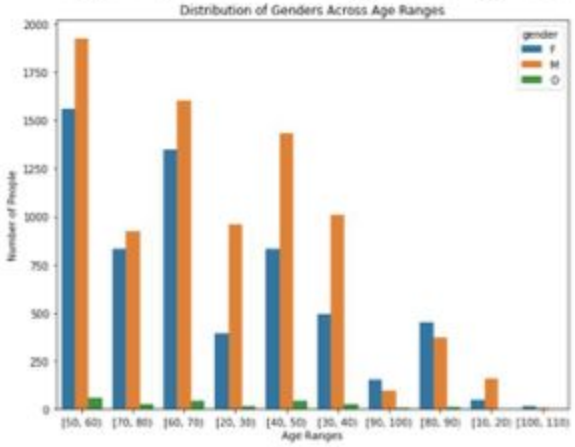
EDA

1. **Before Combining the Data**

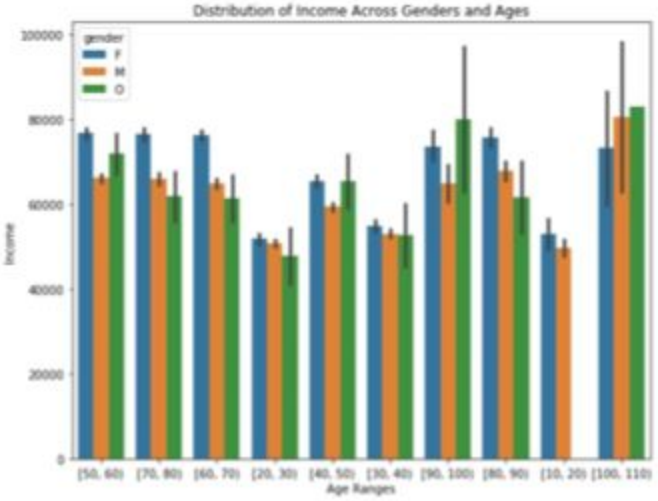
Here we can observe that most of the customers have the age between 50 and 60.



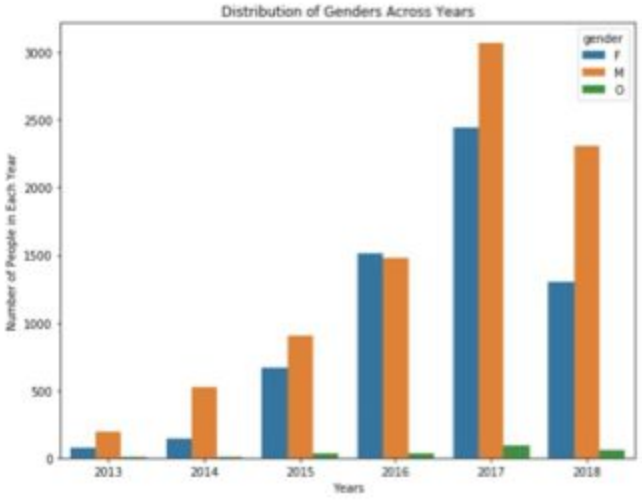
Here we observe that there are more males than any other gender category and are more customers in the age range of 50-60 in all gender categories.



Here we can observe that females generally tend to have a higher income than men, even though most customers are men.

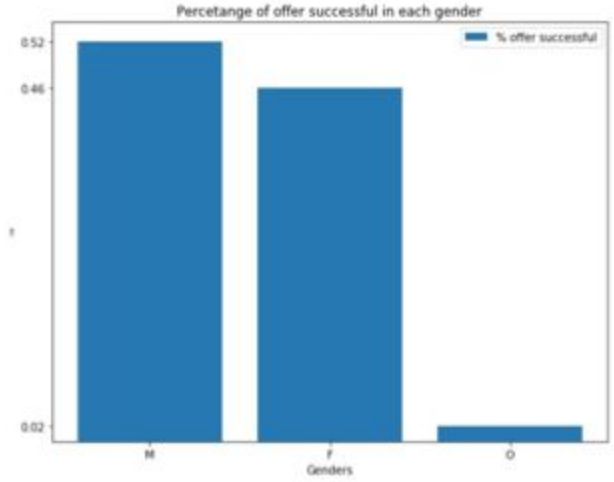


The results show that there is an increasing trend in the number of customers over the years. It had its peak in 2017, however, in 2018 this trend started to decrease.

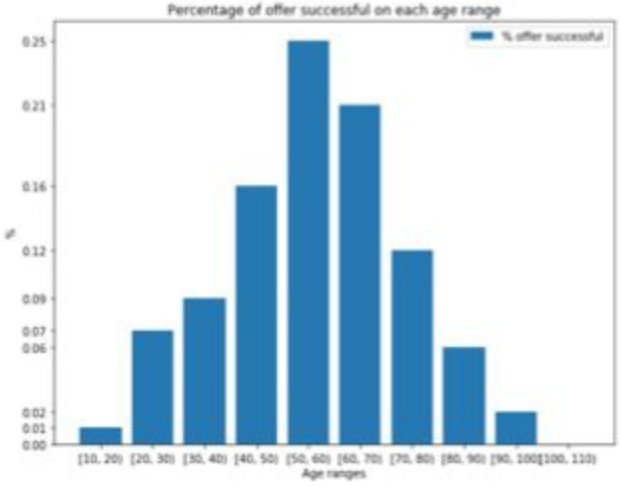


2. After Combining the Data

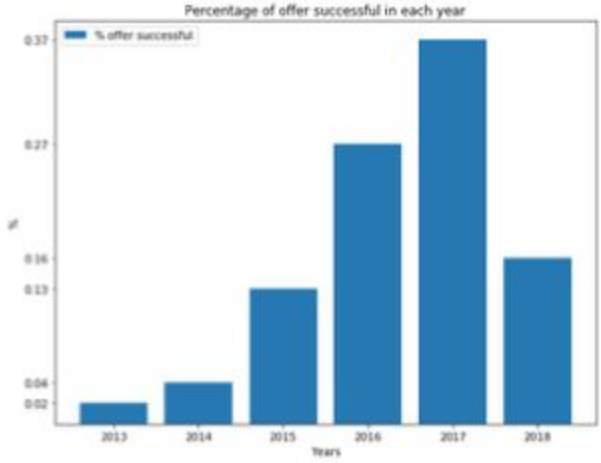
Here we can see that the most successful offers were offered for males.



Here we can see that the most successful offers were offered for the age ranges 40 - 70 years old customers.



Here we can see that the most successful offers were offered in the year 2017, followed by 2016 and decrease in the year 2018. That was expected since the number of customers that start to be a member follow the same distribution.



Preparing the Data

Combining all data

Offer portfolio, customer profile and transaction data are combined to form a single dataset and train models. In the combined dataset, each row will describe an offer's attributes, customer demographic data, and whether the offer was successful and through which channels the offer was advertised. An offer is said to be successful only if it is viewed and completed within a given duration.

	offer_id	customer_id	time	total_amount	offer_successful	reward	difficulty	duration_days	bogo	discount	...
0	9b98b8c7a33c4b65b9aebfe6a799e6d9	78afa995795e4d85b5d9ceeca43f5fef	0.0	37.67	1	5	5	7	1	0	...
1	5a8bc65990b245e5a138643cd4eb9837	78afa995795e4d85b5d9ceeca43f5fef	7.0	49.39	0	0	0	3	0	0	...
2	ae264e3637204a6fb9bb56bc8210ddfd	78afa995795e4d85b5d9ceeca43f5fef	17.0	48.28	1	10	10	7	1	0	...
3	f19421c1d4aa40978ebb69ca19b0e20d	78afa995795e4d85b5d9ceeca43f5fef	21.0	48.28	1	5	5	5	1	0	...
4	2906b810c7d4411798c6938adc9daaa5	e2127556f4f64592b11af22de27a7932	0.0	0.00	0	2	10	7	0	1	...
...	...	...	...	...	...	...	...	...	...	...	...
66496	2906b810c7d4411798c6938adc9daaa5	3cc622f76e4464bc1809a71e671579d6c	24.0	34.04	0	2	10	7	0	1	...
66497	5a8bc65990b245e5a138643cd4eb9837	01fe5ec6682416086eb2f7ec374cb1b7	24.0	19.92	0	0	0	3	0	0	...
66498	9b98b8c7a33c4b65b9aebfe6a799e6d9	d825b707198046e39baa49dcfa11d621	24.0	28.42	0	5	5	7	1	0	...
66499	2298d6c36e964ae4a3e7e9706d1fb8c2	c207a8a8ef5c47ee970ca458859e07eb	24.0	75.84	1	3	7	7	0	1	...
66500	2298d6c36e964ae4a3e7e9706d1fb8c2	9a3f45cf29ef428b932492c7a5d6ac52	24.0	14.25	1	3	7	7	0	1	...

66501 rows × 35 columns

Columns: ['time', 'offer_id', 'customer_id', 'total_amount', 'offer_successful', 'difficulty', 'duration_days', 'reward', 'bogo', 'discount', 'informational', 'email', 'mobile', 'social', 'web', 'income', 2013, 2014, 2015, 2016, 2017, 2018, 'F', 'M', 'O', '[10, 20)', '[20, 30)', '[30, 40)', '[40, 50)','[50, 60)', '[60, 70)', '[70, 80)', '[80, 90)', '[90, 100)', '[100, 110)']

Split the data in train/test sets with 70% and 30% respectively.
Train shape: (46550, 28)
Test shape: (19951, 28)

Evaluation Metrics

To evaluate the performance of the models I will use the accuracy and f1-score metrics, since in my opinion both recall and precision are important here. Finally, I will compute and plot the confusion matrix and the auc roc curve.

To understand the metrics I will use in this project, we first need to understand the confusion matrix.

	Predicted class		
Actual Class		Class = Yes	Class = No
	Class = Yes	True Positive	False Negative
	Class = No	False Positive	True Negative

In our context:

- **True Positive (TP)**: Send an offer and the user will likely use it.
- **False Positive (FP)**: Send an offer but the user doesn't want it or doesn't use it.
- **True Negative (TN)**: Do not send an offer and the user doesn't want it or doesn't use it.
- **False Negative (FN)**: Do not send an offer and the user would have likely used it if we sent it.

Accuracy = TP+TN/TP+FP+FN+TN
Precision = TP/TP+FP
Recall = TP/TP+FN
F1 Score = 2*(Recall * Precision) / (Recall + Precision)

AUC ROC Curve it tells how much a model is capable of distinguishing between classes.

Models Used

First, I constructed a naive predictor model as a baseline for the other models. A Naive predictor assumes every offer being sent is successful irrespective of any

features and this kind of sets a baseline to evaluate the performance of the other two models that I build.

Naive predictor performance

```
naive_predictor_accuracy = accuracy_score(y_train, np.ones(len(y_train)))
naive_predictor_f1score = f1_score(y_train, np.ones(len(y_train)))

print("Naive predictor accuracy: %.3f" % (naive_predictor_accuracy))
print("Naive predictor f1-score: %.3f" % (naive_predictor_f1score))
```

```
Naive predictor accuracy: 0.471
Naive predictor f1-score: 0.640
```

My second model was a logistic regression from scikit-learn library to use as a solid baseline model against the random forest. To set the first hyperparameters I used random_state=42 and solver='liblinear'.
To performance the randomized search cv, I constructed the follow parameters dictionary:

- Penalty: l1, l2
- C: 1.0, 0.1, 0.01

The results suggest that a logistic regression model's accuracy and f1-score on the test data is better than the naive predictor

- Accuracy
    - Naive predictor: 0.471
    - Logistic regression: 0.722
- F1-score
    - Naive predictor: 0.640
    - Logistic regression: 0.717

My last model was a random forest classifier from scikit-learn library. To set the first hyperparameters I used just the random_state=42.

To performance the randomized search cv, I constructed the follow parameters dictionary:

- n_estimators: [10, 50, 100, 150, 200, 250, 300]
- max_features: ['auto', 'sqrt']
- max_depth: [int(x) for x in np.arange(3, 11)] + [None]
- min_samples_split: [2, 5, 10]
- min_samples_leaf: [1, 2, 4]

The results suggest that a random forest model's accuracy and f1-score on the test data is better than the naive predictor and logistic regression

- Accuracy
    - Naive predictor: 0.471
    - Logistic regression: 0.722
    - Random forest 0.731
- F1-score
    - Naive predictor: 0.640
    - Logistic regression: 0.717
    - Random forest 0.724

Comparing the Models

Below is the dataframe showing the metrics for each model:

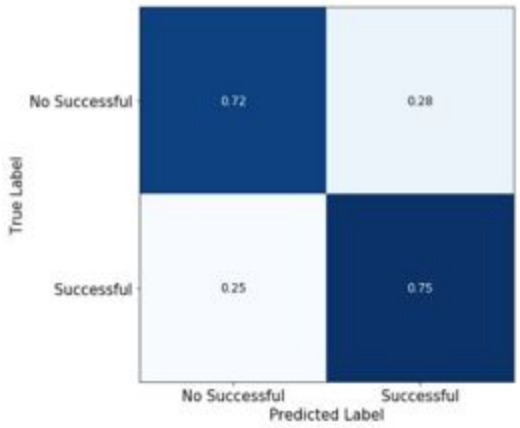
	classifiertype	accuracy	f1score
0	randomforest	0.731041	0.724312
1	logisticregression	0.722270	0.716993
2	naivepredictor	0.470763	0.640161

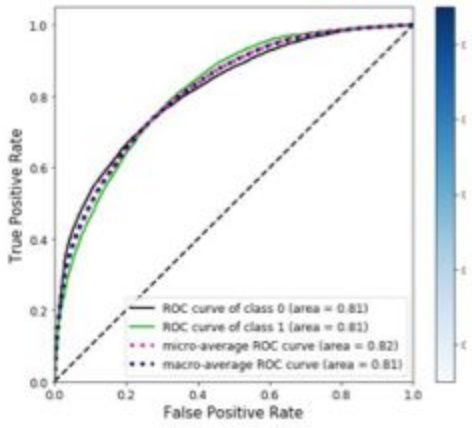
The results show the random forest achieved the best performance after tuning. The difference is very small but between the top two models and their performance was much better than the benchmark.

Here we can see the best parameters for the best model:

```
RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                       criterion='gini', max_depth=10, max_features='auto',
                       max_leaf_nodes=None, max_samples=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=300,
                       n_jobs=None, oob_score=False, random_state=42, verbose=0,
                       warm_start=False)
```

Here we can observe the normalized confusion matrix and the roc auc curve of the best model:





Conclusion

The problem that I chose to solve was to build a model that predicts whether a customer will respond to an offer. My strategy for solving this problem has four steps. First, I combined offer portfolio, customer profile, and transaction data. Second, I assessed the accuracy and F1-score of a naive model that assumed all customer offers were successful. Third, I compared the performance of logistic regression, random forest, and gradient boosting models. This analysis suggests that a random forest model has the best training data accuracy and F1-score. Fourth, I refined random forest model hyperparameters using a grid search. My analysis suggests that the resulting random forest model has on test data accuracy of 0.731 and an F1-score of 0.724 suggests that the random forest model I constructed did not overfit the training data.

Improvements

The performance of a random forest model can be still improved by analysing features which impacts an offer's success rate as a function of offer difficulty, duration, and reward. These additional features should provide a random forest classifier with the opportunity to construct a better decision boundary that separates successful and unsuccessful customer offers.