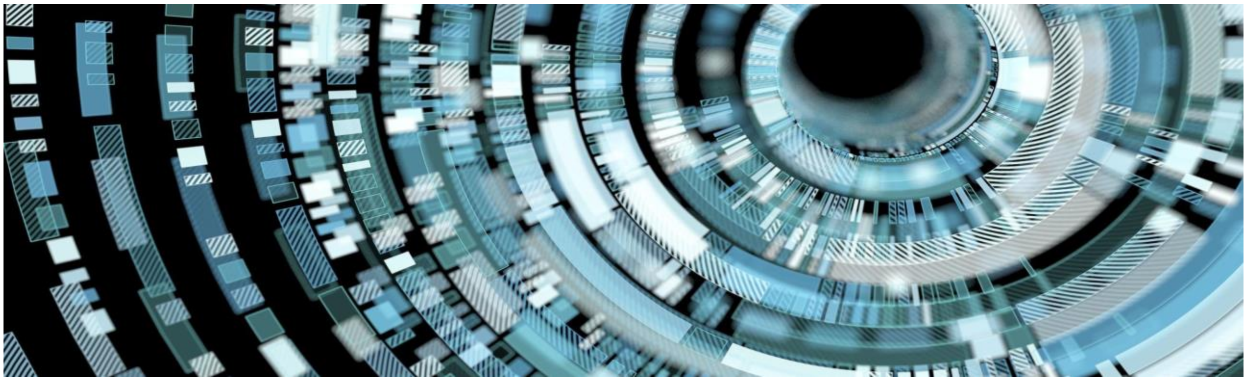


Machine Learning

Engineer Nanodegree

Udacity

Douglas Wong 2022

Capstone Project Report

Table of Contents

Project Definition

Project Overview

Starbucks is by far one of the world's largest franchise coffee shops. They are known for having implemented one of the most successful information technology solutions, which has allowed them to grow into industry leaders. Starbucks debuted its mobile order-ahead app feature in late 2014 and it quickly caught on with Starbucks Rewards members *(a)*. Starbucks has also become a leading mobile payment app that competes with Google Pay, Apple Pay, and Samsung Pay(*c). Starbuck is also well-known for its loyalty program My Starbucks® Rewards where it able to offer individualized offers to their member. The program grew by 16 percent year over year in the first quarter of 2020, reaching 18.9 million active users *(b)*. The membership increase is correlated to the increase in sales growth *(c).*

According to Starbucks, a quarter of their transactions will be completed over the phone by the end of 2020 *(a)*. This suggests that the rewards app accounts for a sizable portion of their revenue. In the recent decade, the use of artificial intelligence (AI) in marketing has grown exponentially and machine learning is providing a huge advantage to target customers, predict product performance and customer behavior *(d)*.

In this capstone project, we want to look at how the customers used the Starbucks rewards app so that we can improve earnings through targeted offers to drive sales.

In this project, we will be looking at the derived data from the Starbucks reward mobile app. The Starbucks app rewards registered customers on its platform to entice them to make purchases. There are 3 main types of offers that are sent to the customer.

1. Buy one get one Offer (BOGO)
2. Discount Offer
3. Informational Offer

In a BOGO offer, a user needs to spend a certain amount to get the reward. In the discount offer, the user receives a reward equal to the fraction of the amount spent. In an information offer, there will be no reward nor minimum amount spent.

However not all customer response to the same marketing campaign, some customers will response to campaign regardless of reward such as recurring customer, while certain customer such as new customer need to be attracted via discount.

The data that has been collected by the app reward is a data mine which offers us insights on customer base spending habits, and thus based on this valuable data, we can utilize this using Machine Learning to increase the ROI of the marketing campaign.

Problem Statement

Every company invests money in marketing campaigns, expecting that it will be successful in bringing in more profit as an outcome. Therefore, it is imperative that we can increase the return on investment (ROI) by identifying the most effective offer type to be offered to the different subgroups of our customer base.

Based on the dataset that we have which we obtained from the Starbucks reward mobile app, we are proposing to utilize machine learning methodology to build a model to predict the success of campaign offer type. This can allow us to determine which offer should be targeted at different subgroups of customers as well.

The proposed plan was to merge the portfolio of offer, the profile and transcript together into a big dataset where we can analyze the data. We will also determine if a particular offer is successful based on the record of the transactions that occur during the duration of an offer. Besides investigating if an offer is successful, we will also be looking into the amount of profit that each offer brings in as another metric that we can investigate.

We will consider a successful offer as follows. It must satisfy two conditions, offers received must be viewed, and the transaction must occur during the duration of the offer.

If an offer were received but not viewed, it would mean that the customer would have made the purchase regardless of the offer. It is imperative that we exclude the transactions that occur that are not due to an offer to analyze the success rate of a particular offer

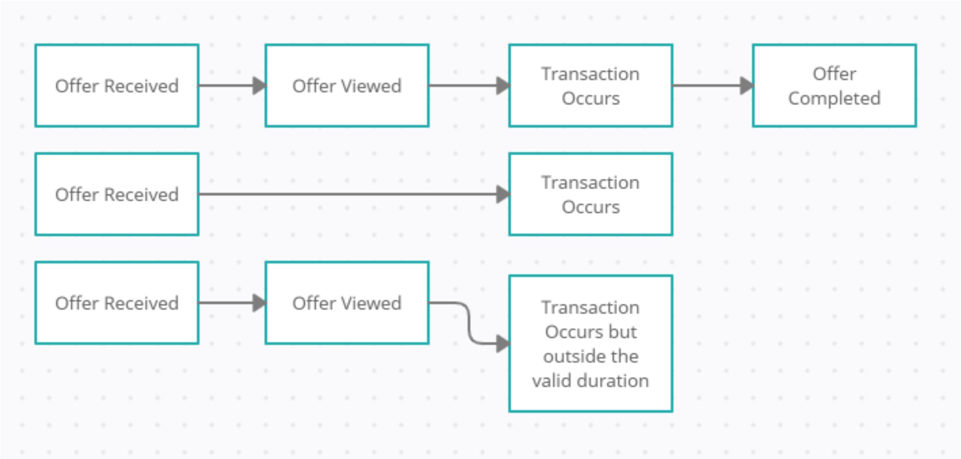


Figure 1 Valid Offer Flowchart Diagram

We will be utilizing the Amazon SageMaker platform for its integrated development environments, creating a Sagemaker notebook instance. It is a machine learning compute instance running the Jupyter Notebook App. We will use this environment to do our data processing, then we will upload our training data to Amazon S3 Cloud Object Storage. It will also allow us to deploy our trained model as an endpoint which can be called by a lambda function to be consumed by applications. However, in this problem we will only be testing the data via the endpoint in sagemaker.

As we have a labeled dataset, we will be using supervised learning algorithms to predict if an offer is going to be successful. We will be exploring and comparing three algorithms, scikit-learn library logistic regression, scikit-learn library random forest, and XGBoost algorithm provided by Amazon SageMaker. Logistic regression is a statistical method for predicting binary classes *(e)*. Random forests classifier is an ensemble of decision trees trained on randomly selected data samples, then the best prediction from each tree and select the best solution by voting*(f)*. XGBoost is an efficient implementation of gradient boosting. Gradient boosting is an algorithm that combines many weak learning models together to create a strong predictive model *(g)*. The XGBoost algorithm would be our main model as we expect it to perform the best if properly tuned.

Finally, we will also look at the feature importance to describe how important that a particular feature in a model is for predicting the success of an offer. This will help us in identifying the best features to target while building an offer dataset that can be used to predict the success rate of Starbuck Offer Campaign.

Metrics

To see how well our supervised classification model performs here, we assessed and compared both the accuracy score and the F1 score of the baseline linear regression model against random forest and XGBoost gradient boosting.

We compared accuracy here since the accuracy of the True positive and True negative is important in our cases, ie to see if the data model can successfully predict if an offer campaign will be successful. The more accurate the model is, the more successful our marketing campaign is.

In the case where the class distribution of our data might be uneven, we also compared the F1 Score as it will give a better measure of the incorrectly classified cases than the Accuracy Metric. F1 score is the weighted average of precision and recall value. Only when the recall and precision is high, we can achieve a high F1 Score. We will then tune the hyperparameter of the model that had the highest accuracy and F1 score.

I have also look at the `**Feature Importance**` via random forest classifier to estimate feature importance to describe how important that a particular feature in a model at predicting the success of an offer.

Analysis

Data Exploration and Visualization

We have done exploratory data analysis on our Starbucks Capstone Challenge dataset, to summarize and visualize the main characteristic of our dataset. Exploratory data analysis is the most crucial step in Machine learning as it allows us to analyze and understand our data. This has allowed us to plan for the data preprocessing that was needed to analyze the dataset.

There are 3 data files in this project

- **`portfolio.json`**
- **`profile.json` -** demographic data for each customer
- **`transcript.json` -** records for transactions, offers received, offers viewed, and offers completed

Here is the schema and explanation of each variable in the files:

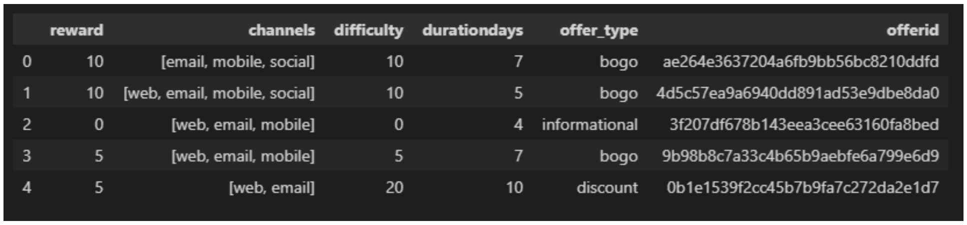
1) `portfolio.json` Size: 10 rows x 6 fields

- **id** (string)-offer id
- **offer_type** (string)-the type of offer ie BOGO, discount, informational
- **difficulty** (int)-the minimum required to spend to complete an offer
- **reward** (int)-the reward is given for completing an offer
- **duration** (int)-time for the offer to be open, in days
- **channels** (list of strings)

This offer portfolio contains data regarding the type of offers meta data about each offer. There are 3 types of offers.

1. **Buy one get one Offer (BOGO)**. Users need to spend a certain amount to get the reward
2. **Discount Offer**. A user gets a reward money off the purchase price based on the percentage of the amount spent.
3. **Informational Offer**. Purely informational, there is no requirement or reward.

Offers can be delivered via multiple channels such as the web, email, mobile and social media. The offers also have a duration period where they remain valid before they expire.



	reward	channels	difficulty	durationdays	offer_type	offerid
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

Figure 2 Portfolio Data Example

The steps that I have undertaken to clean the portfolio data are as followed.

As we are going to merge the several dataset together further down, we are going to rename the *id* to *offerid*, and the *duration* to *durationdays*,

The channels data are stored as data array and therefore must be converted to columns via one hot encoding for model training.

From the histogram below, we can identify that all offers contain email as a delivery medium and thus we will be removing this feature from our dataset as it bears no predictive value.

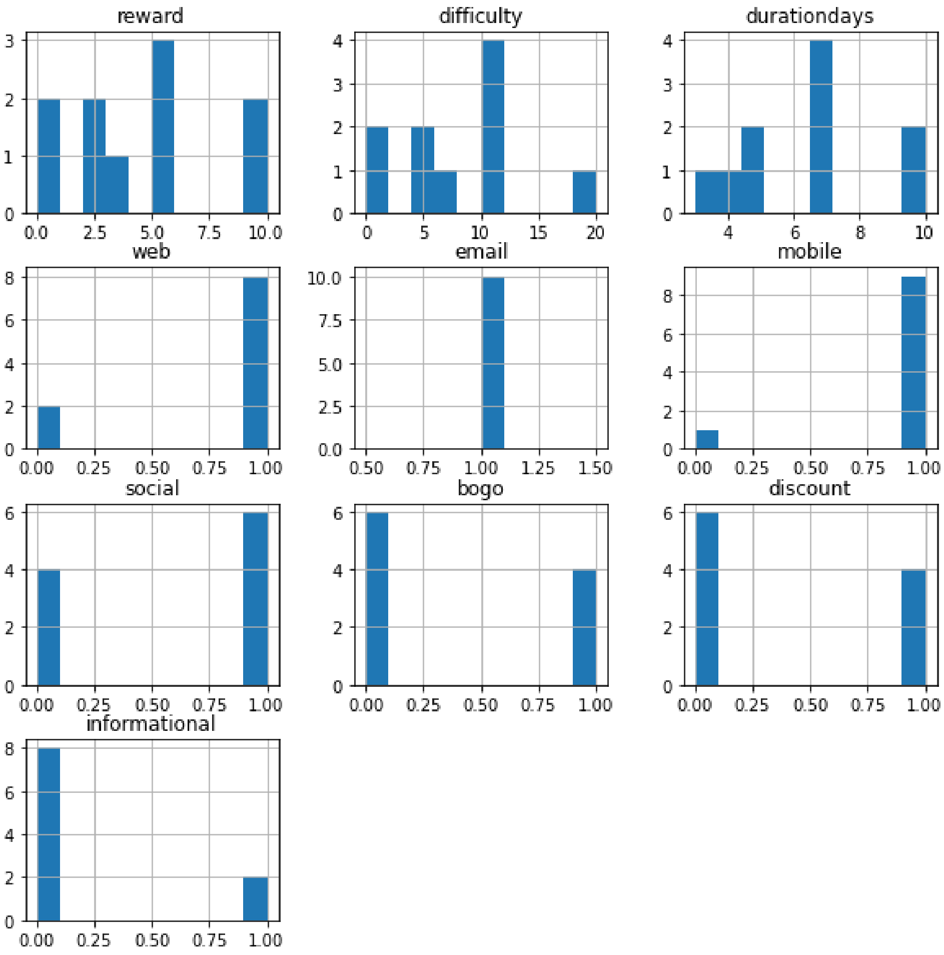


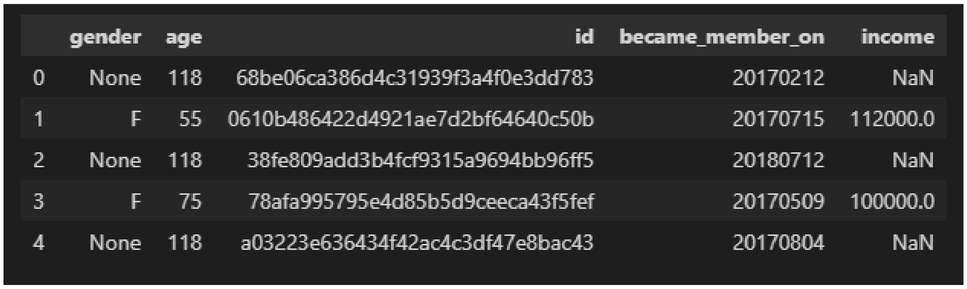
Figure 3 Offer Portfolio Histogram

The profile dataset contains the customer demographic data.

2) `profile.json` Size: 17000 rows x 5 fields

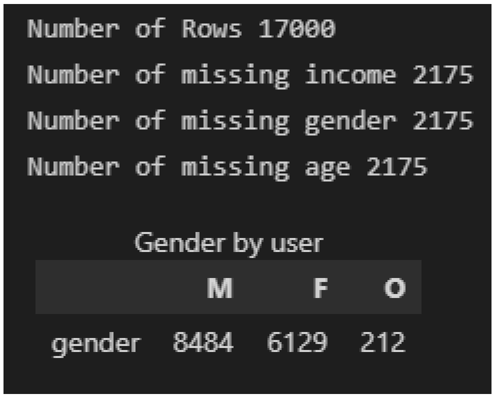
- age (int)-age of the customer
- became_member_on (int)-the date when customer created an app account
- gender (str)-gender of the customer (note some entries contain 'O' for other rather than M or F)
- id (str)-customer id

- income (float) - customer's income



	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

Figure 4 Profile Data



```
Number of Rows 17000
Number of missing income 2175
Number of missing gender 2175
Number of missing age 2175

       Gender by user
                 M      F     O
  gender      8484   6129   212
```

Figure 5 Missing profile data summary

Looking at the data, we immediately noticed that the income and gender are missing from the dataset, and out of the 17000 rows of data, 2175 records contain missing income and gender, the age is also missing and coded as the value 118. This could be due to the users have not accepted the privacy policy of the Starbuck reward app. We have figured out that the missing data was contribute to 12.79 % of the dataset. Instead of interpolating the data with the mean of data for age and income. We have decided to omit these customers from our dataset to ensure the quality of our dataset. We have also removed the 1.43% of the customers that have *gender* coded as *O.*

We have renamed the became_member_on to member_days after converting the days by subtracting the today date. Then we renamed the *id* to *customerid* so that it can be merged with the other dataset below.

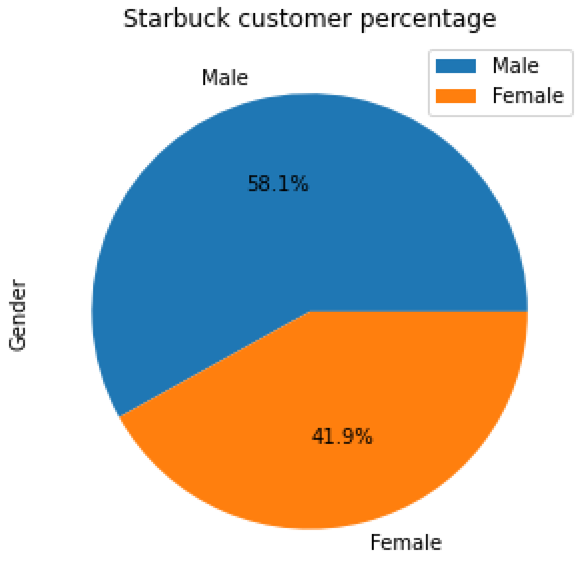


Figure 6 Starbuck customer percentage based on gender piechart

We have a higher percentage of Male Starbuck customer in our dataset.

Based on the distribution histogram plot below, while the age histogram for both genders are normally distributed, we have found out that the male is slightly younger than female at 52 mean years old compared to female at 57 mean years old.

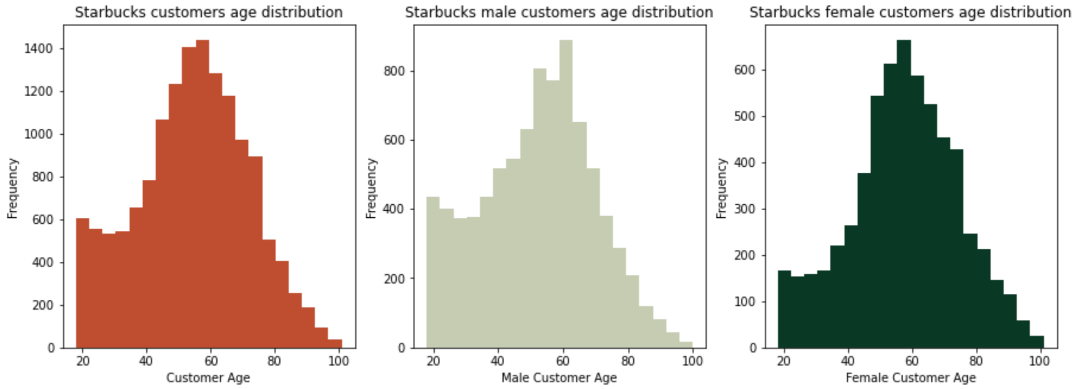


Figure 7 Starbuck customer age distribution histogram

Looking at the histogram of the data below, we can determine that the member days have a large left skew on the date that the member joined. It means that most customers that joined the Starbuck reward apps are recently joined, and this could be due to it becoming more popular over time. However, due to this reason, we believe that this is not a suitable predictive feature that we would like for our models due to strong biased.

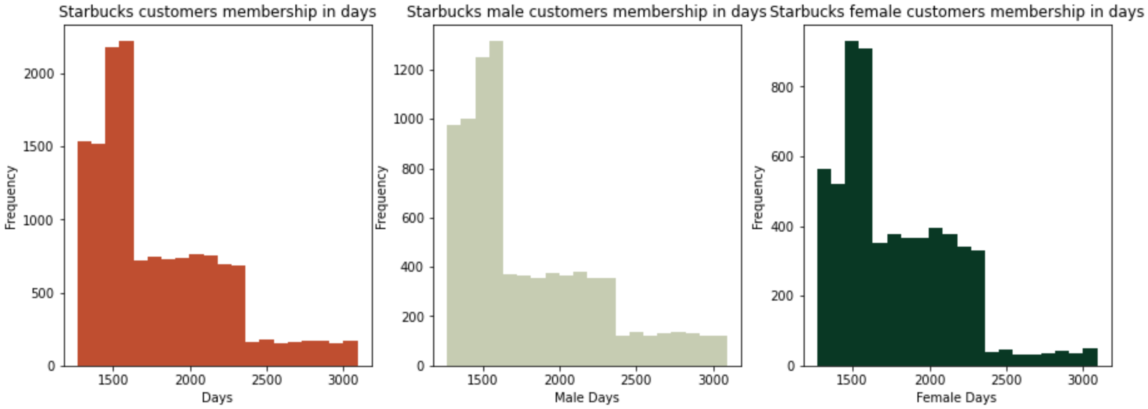
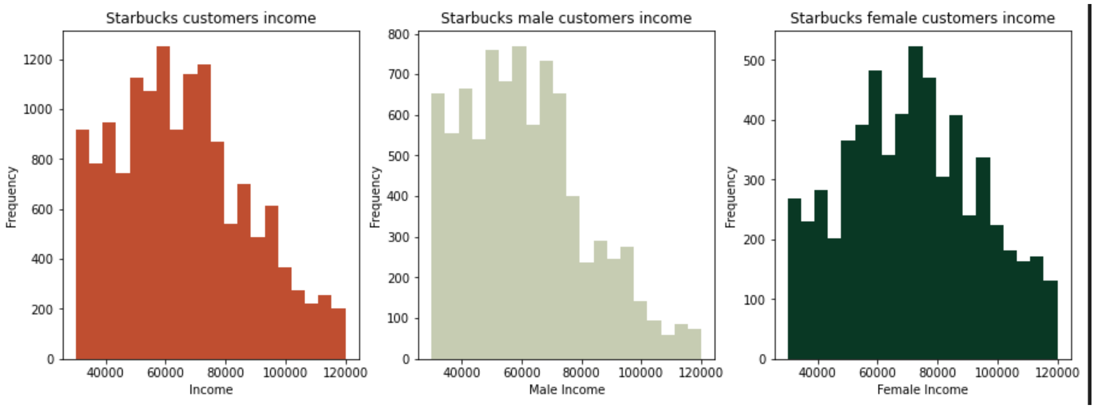


Figure 8 Starbuck customer membership days distribution histogram

The income graph has a right skew and this was contributed by the male income. We can see that the female has a higher mean income at 71k when compared to the male at mean 61k.

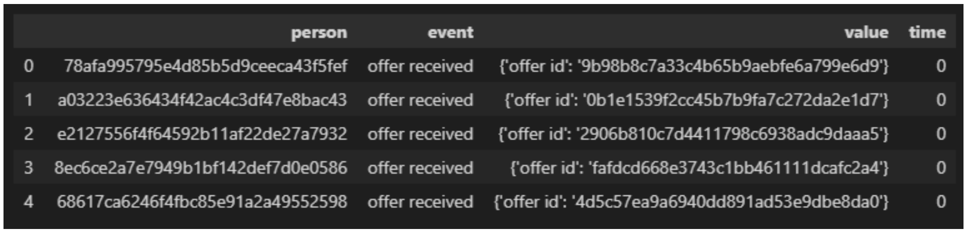


d

Figure 9 Starbuck customer income distribution histogram

3) `transcript.json` Size: 306534 rows x 4 fields

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



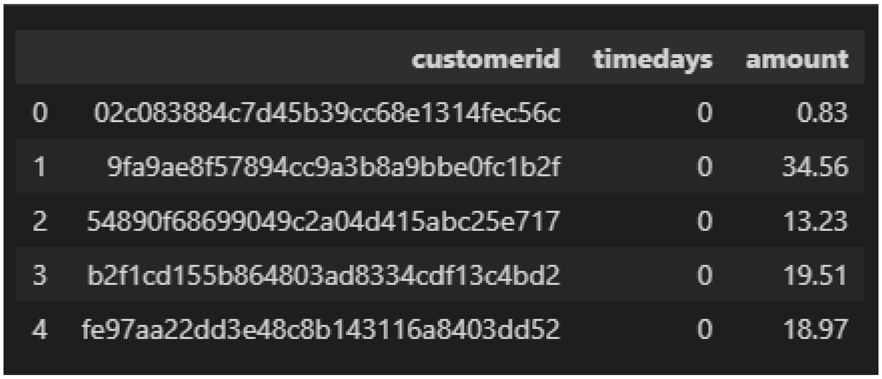
	person	event	value	time
0	78afa995795e4d85b5d9ceeca43f5fef	offer received	{'offer id': '9b98b8c7a33c4b65b9aebfe6a799e6d9'}	0
1	a03223e636434f42ac4c3df47e8bac43	offer received	{'offer id': '0b1e1539f2cc45b7b9fa7c272da2e1d7'}	0
2	e2127556f4f64592b11af22de27a7932	offer received	{'offer id': '2906b810c7d4411798c6938adc9daaa5'}	0
3	8ec6ce2a7e7949b1bf142def7d0e0586	offer received	{'offer id': 'fafdcd668e3743c1bb461111dcafc2a4'}	0
4	68617ca6246f4fbc85e91a2a49552598	offer received	{'offer id': '4d5c57ea9a6940dd891ad53e9dbe8da0'}	0

Figure 10 Transcript Data

The last part is the transaction data which describes the event where the customer made a purchase and the amount spent. It also recorded the event where they received an offer, viewed an offer, and completed an offer. Based on preliminary analysis, I have decided to split the *transcript* dataset into two different datasets, the *transaction* dataset, and the *offer* dataset.

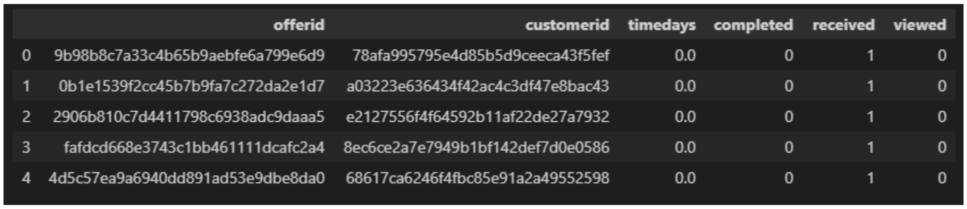
To prepare data for merging, I have taken the following steps to clean it. I have renamed the column *person* to *customerid*. Remove the customers that are not in our cleaned profile dataset. Then we converted the time from hours to days and renamed it to *timedays*.

We then filter off the event that contains the word *offer such as offer received, offer viewed, offer completed.* We then hot encode this offer events to their own column. The data that does not contain the event offer will be our transaction data. We will populate the *amount* column with the value of the transaction from the value column.



	customerid	timedays	amount
0	02c083884c7d45b39cc68e1314fec56c	0	0.83
1	9fa9ae8f57894cc9a3b8a9bbe0fc1b2f	0	34.56
2	54890f68699049c2a04d415abc25e717	0	13.23
3	b2f1cd155b864803ad8334cdf13c4bd2	0	19.51
4	fe97aa22dd3e48c8b143116a8403dd52	0	18.97

Figure 11 Transaction Dataset



	offerid	customerid	timedays	completed	received	viewed
0	9b98b8c7a33c4b65b9aebfe6a799e6d9	78afa995795e4d85b5d9ceeca43f5fef	0.0	0	1	0
1	0b1e1539f2cc45b7b9fa7c272da2e1d7	a03223e636434f42ac4c3df47e8bac43	0.0	0	1	0
2	2906b810c7d4411798c6938adc9daaa5	e2127556f4f64592b11af22de27a7932	0.0	0	1	0
3	fafdcd668e3743c1bb461111dcafc2a4	8ec6ce2a7e7949b1bf142def7d0e0586	0.0	0	1	0
4	4d5c57ea9a6940dd891ad53e9dbe8da0	68617ca6246f4fbc85e91a2a49552598	0.0	0	1	0

Figure 12 Offer Dataset

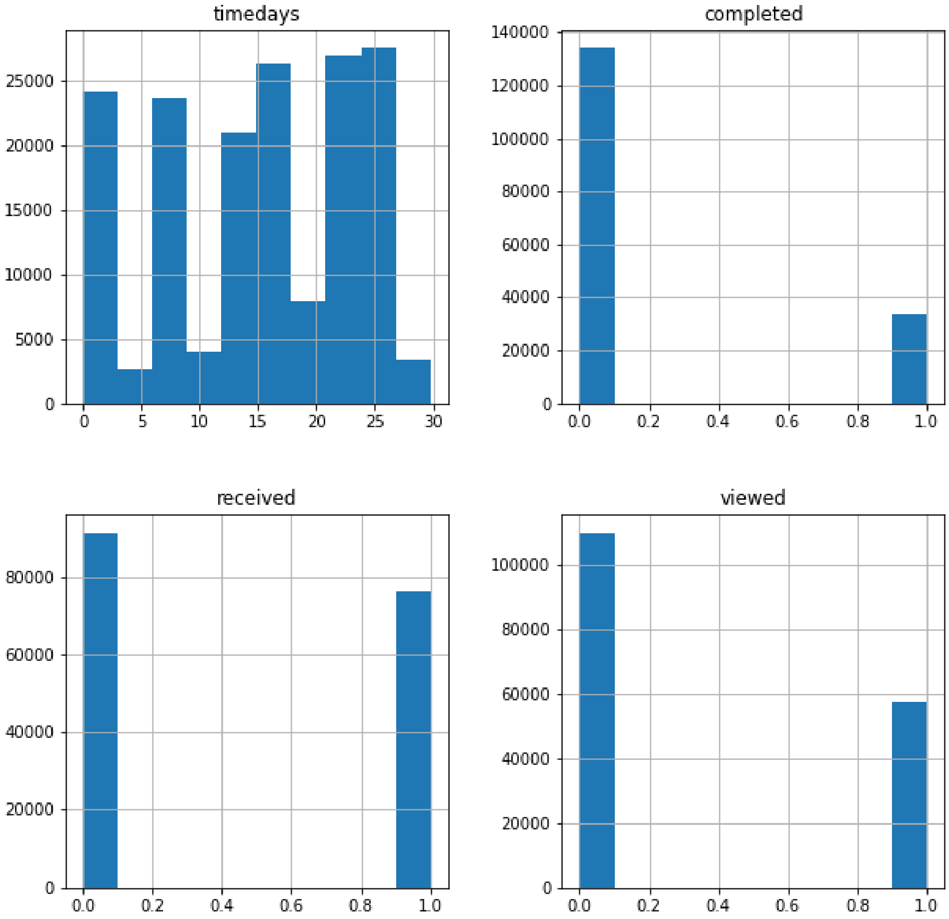


Figure 13 Offer Histogram

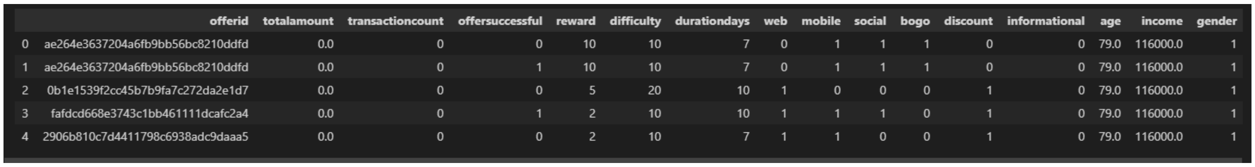
Based on these two datasets, we will process the customer transaction record.

We will consider a successful offer as follows. It must satisfy two conditions, offers received must be viewed, and the transaction must occur during the duration of the offer. What we have done here is to loop through every single offer that a particular customer received.

We first workout the end time of the offer by adding the *durationday* to the offer received *timedays*.

Then we check if the particular transaction is within the offer period, by comparing the transaction *timeday* to more than the current offer start time and the current offer end time. We then establish an offer that is successful if a customer viewed the offer and completed the transaction within the duration window. We also added up the amount of transaction of the successful offer and assigned it to *totalamount*. We also included the total count of the valid transaction *transactioncount* from that offer. This will allow us to investigate how many purchases a particular offer brings in as well.

Once we have done this for all the offers under all the customers, then we will merge the cleaned portfolio dataset and the cleaned profile dataset.

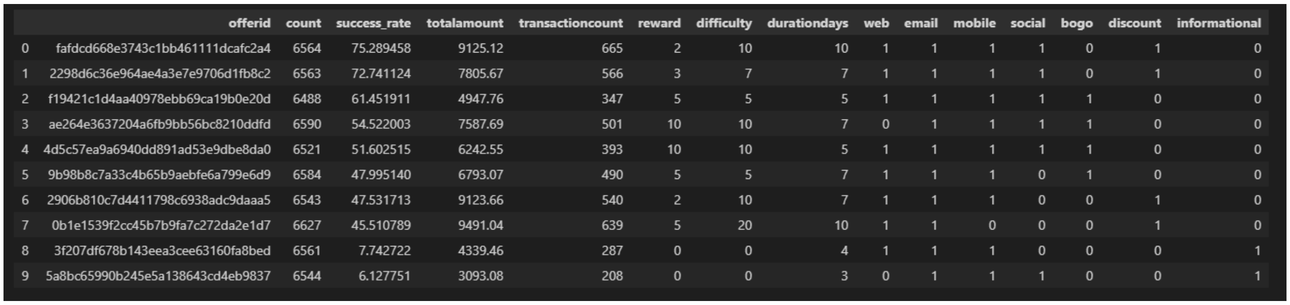


	offerid	totalamount	transactioncount	offersuccessful	reward	difficulty	durationdays	web	mobile	social	bogo	discount	informational	age	income	gender
0	ae264e3637204a6fb9bb56bc8210ddfd	0.0	0	0	10	10	7	0	1	1	1	0	0	79.0	116000.0	1
1	ae264e3637204a6fb9bb56bc8210ddfd	0.0	0	1	10	10	7	0	1	1	1	0	0	79.0	116000.0	1
2	0b1e1539f2cc45b7b9fa7c272da2e1d7	0.0	0	0	5	20	10	1	0	0	0	1	0	79.0	116000.0	1
3	fafdcd668e3743c1bb461111dcafc2a4	0.0	0	1	2	10	10	1	1	1	0	1	0	79.0	116000.0	1
4	2906b810c7d4411798c6938adc9daaa5	0.0	0	0	2	10	7	1	1	0	0	1	0	79.0	116000.0	1

Figure 14 Merged Dataset

Analyse the success rate, the total amount of the gross offer and the number of transactions that the offer bring in

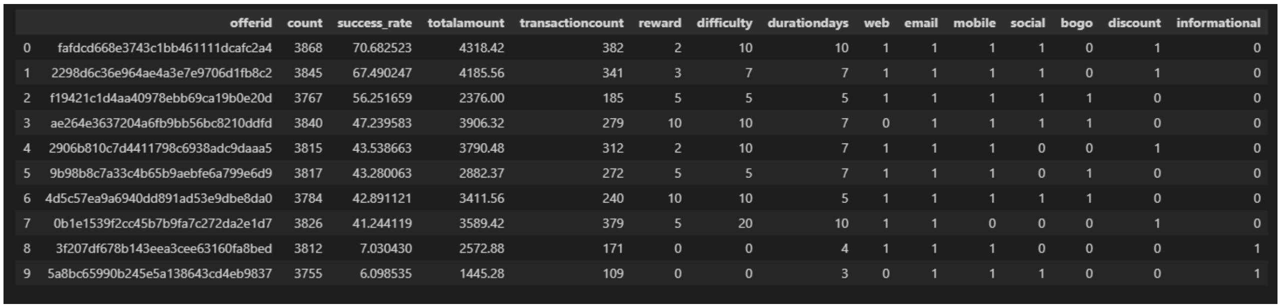
To analyze the resulting dataset, we first group the data by the *offerid*, and we can clearly see that offer ending as c2a4 has the highest success rate and brings in a total of 9125 dollars.



	offerid	count	success_rate	totalamount	transactioncount	reward	difficulty	durationdays	web	email	mobile	social	bogo	discount	informational
0	fafdcd668e3743c1bb461111dcafc2a4	6564	75.289458	9125.12	665	2	10	10	1	1	1	1	0	1	0
1	2298d6c36e964ae4a3e7e9706d1fb8c2	6563	72.741124	7805.67	566	3	7	7	1	1	1	1	0	1	0
2	f19421c1d4aa40978ebb69ca19b0e20d	6488	61.451911	4947.76	347	5	5	5	1	1	1	1	1	0	0
3	ae264e3637204a6fb9bb56bc8210ddfd	6590	54.522003	7587.69	501	10	10	7	0	1	1	1	1	0	0
4	4d5c57ea9a6940dd891ad53e9dbe8da0	6521	51.602515	6242.55	393	10	10	5	1	1	1	1	1	0	0
5	9b98b8c7a33c4b65b9aebfe6a799e6d9	6584	47.995140	6793.07	490	5	5	7	1	1	1	0	1	0	0
6	2906b810c7d4411798c6938adc9daaa5	6543	47.531713	9123.66	540	2	10	7	1	1	1	0	0	1	0
7	0b1e1539f2cc45b7b9fa7c272da2e1d7	6627	45.510789	9491.04	639	5	20	10	1	1	0	0	0	1	0
8	3f207df678b143eea3cee63160fa8bed	6561	7.742722	4339.46	287	0	0	4	1	1	1	0	0	0	1
9	5a8bc65990b245e5a138643cd4eb9837	6544	6.127751	3093.08	208	0	0	3	0	1	1	1	0	0	1

Figure 15 Offer analysis table

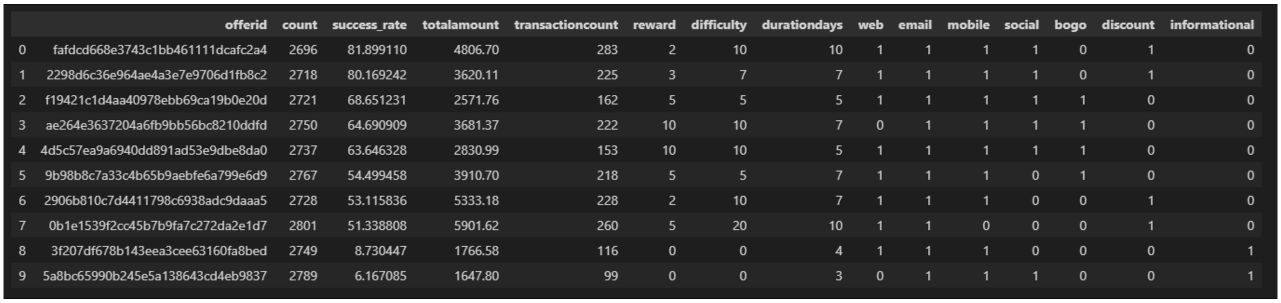
Data group by *offerid* and filtered by gender male



	offerid	count	success_rate	totalamount	transactioncount	reward	difficulty	durationdays	web	email	mobile	social	bogo	discount	informational
0	fafdcd668e3743c1bb461111dcafc2a4	3868	70.682523	4318.42	382	2	10	10	1	1	1	1	0	1	0
1	2298d6c36e964ae4a3e7e9706d1fb8c2	3845	67.490247	4185.56	341	3	7	7	1	1	1	1	0	1	0
2	f19421c1d4aa40978ebb69ca19b0e20d	3767	56.251659	2376.00	185	5	5	5	1	1	1	1	1	0	0
3	ae264e3637204a6fb9bb56bc8210ddfd	3840	47.239583	3906.32	279	10	10	7	0	1	1	1	1	0	0
4	2906b810c7d4411798c6938adc9daaa5	3815	43.538663	3790.48	312	2	10	7	1	1	1	0	0	1	0
5	9b98b8c7a33c4b65b9aebfe6a799e6d9	3817	43.280063	2882.37	272	5	5	7	1	1	1	0	1	0	0
6	4d5c57ea9a6940dd891ad53e9dbe8da0	3784	42.891121	3411.56	240	10	10	5	1	1	1	1	1	0	0
7	0b1e1539f2cc45b7b9fa7c272da2e1d7	3826	41.244119	3589.42	379	5	20	10	1	1	0	0	0	1	0
8	3f207df678b143eea3cee63160fa8bed	3812	7.030430	2572.88	171	0	0	4	1	1	1	0	0	0	1
9	5a8bc65990b245e5a138643cd4eb9837	3755	6.098535	1445.28	109	0	0	3	0	1	1	1	0	0	1

Figure 16 Offer analysis table group by male

Data group by *offerid* and filtered by gender female



	offerid	count	success_rate	totalamount	transactioncount	reward	difficulty	durationdays	web	email	mobile	social	bogo	discount	informational
0	fafdcd668e3743c1bb461111dcafc2a4	2696	81.899110	4806.70	283	2	10	10	1	1	1	1	0	1	0
1	2298d6c36e964ae4a3e7e9706d1fb8c2	2718	80.169242	3620.11	225	3	7	7	1	1	1	1	0	1	0
2	f19421c1d4aa40978ebb69ca19b0e20d	2721	68.651231	2571.76	162	5	5	5	1	1	1	1	1	0	0
3	ae264e3637204a6fb9bb56bc8210ddfd	2750	64.690909	3681.37	222	10	10	7	0	1	1	1	1	0	0
4	4d5c57ea9a6940dd891ad53e9dbe8da0	2737	63.646328	2830.99	153	10	10	5	1	1	1	1	1	0	0
5	9b98b8c7a33c4b65b9aebfe6a799e6d9	2767	54.499458	3910.70	218	5	5	7	1	1	1	0	1	0	0
6	2906b810c7d4411798c6938adc9daaa5	2728	53.115836	5333.18	228	2	10	7	1	1	1	0	0	1	0
7	0b1e1539f2cc45b7b9fa7c272da2e1d7	2801	51.338808	5901.62	260	5	20	10	1	1	0	0	0	1	0
8	3f207df678b143eea3cee63160fa8bed	2749	8.730447	1766.58	116	0	0	4	1	1	1	0	0	0	1
9	5a8bc65990b245e5a138643cd4eb9837	2789	6.167085	1647.80	99	0	0	3	0	1	1	1	0	0	1

Figure 17 Offer analysis table group by female

Based on the histogram below, while we have identified that while **offer 1 (c2a4)** and **2 (fb8c2)** are quite successful, we have also identified that female respond very well to **offer 7 (aaa5)** and **8 (e1d7)** as it brings in as much profit as offer 1 even though the success rate was only about 47% in both the gender. From this observation, we can find out that the amount of profit that the offer brings in is also a very important metric that we can investigate in the future.

The transaction chart and the amount spent chart do look similar.

We can also see that both the informational offers are the two worst successful offers that we have.

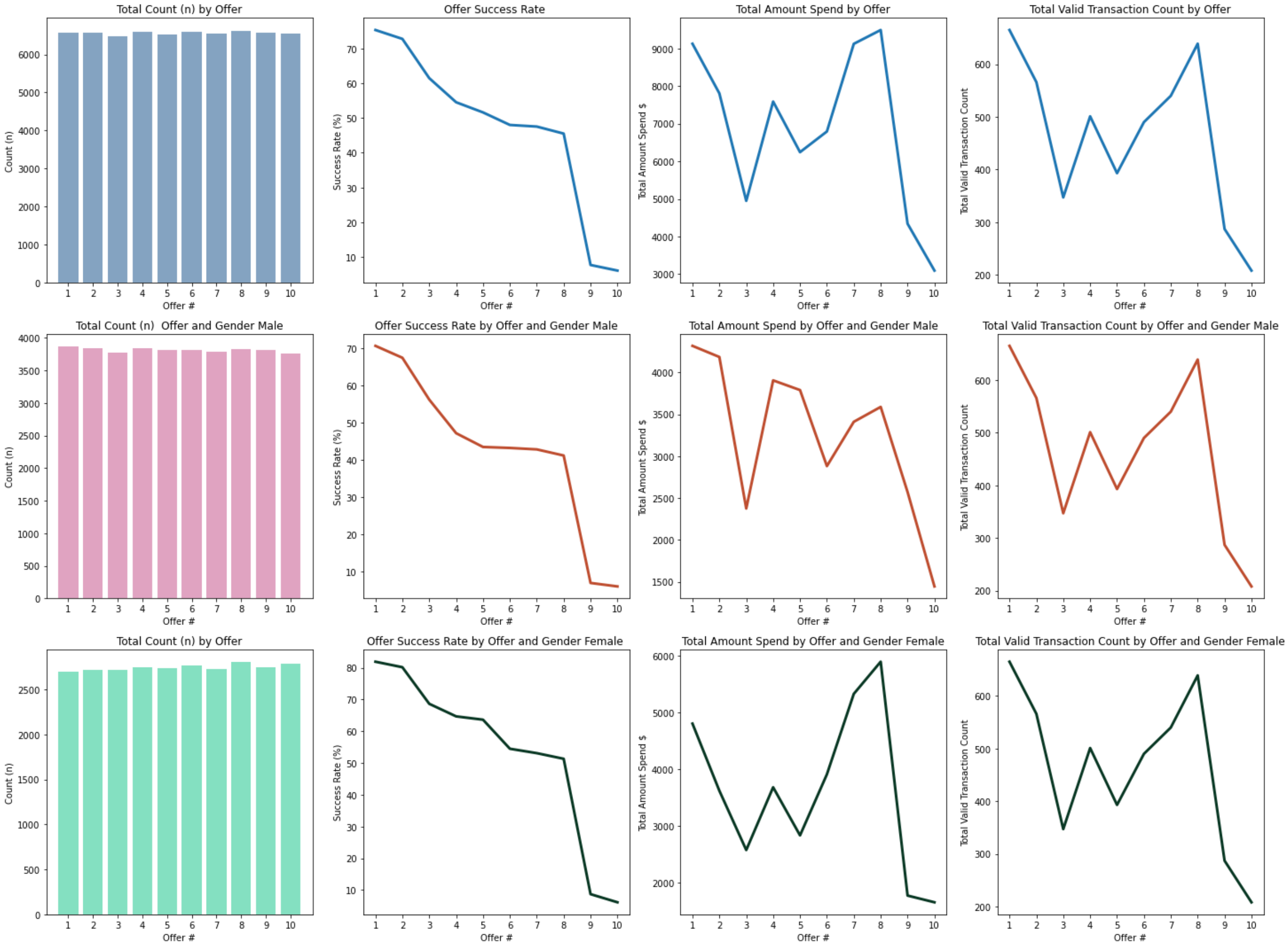
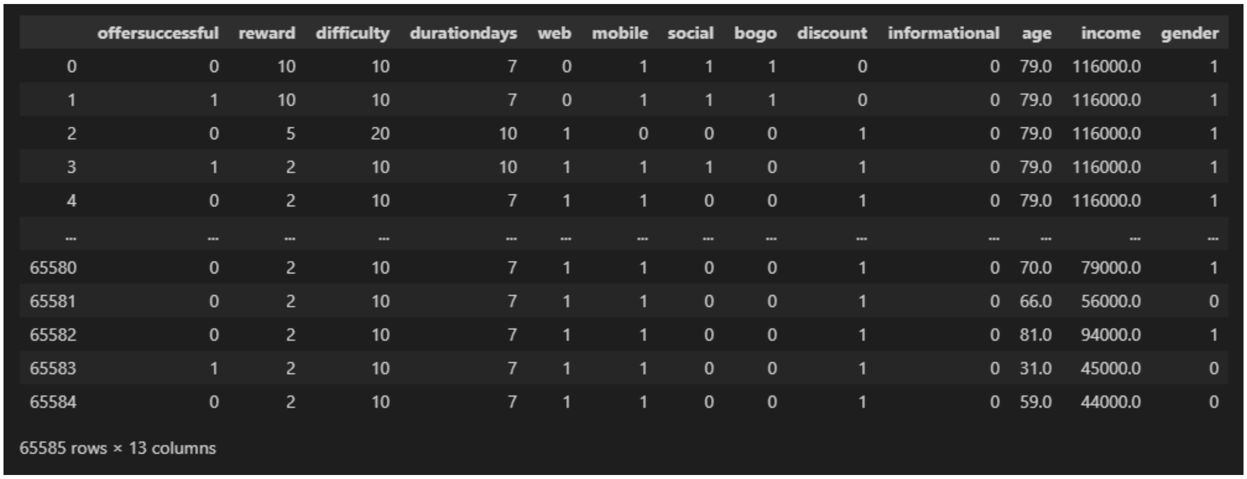


Figure 18 Offer analysis sub plot

Once we have done all the data preprocessing, we will then remove the unused data columns that are not used for model training. We have removed the *offerid* as it is a *guid*, *member_days* having a large negative skew that are not suitable for prediction, and the finally email columns as it was send to every offer.



	offersuccessful	reward	difficulty	durationdays	web	mobile	social	bogo	discount	informational	age	income	gender
0	0	10	10	7	0	1	1	1	0	0	79.0	116000.0	1
1	1	10	10	7	0	1	1	1	0	0	79.0	116000.0	1
2	0	5	20	10	1	0	0	0	1	0	79.0	116000.0	1
3	1	2	10	10	1	1	1	0	1	0	79.0	116000.0	1
4	0	2	10	7	1	1	0	0	1	0	79.0	116000.0	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...
65580	0	2	10	7	1	1	0	0	1	0	70.0	79000.0	1
65581	0	2	10	7	1	1	0	0	1	0	66.0	56000.0	0
65582	0	2	10	7	1	1	0	0	1	0	81.0	94000.0	1
65583	1	2	10	7	1	1	0	0	1	0	31.0	45000.0	0
65584	0	2	10	7	1	1	0	0	1	0	59.0	44000.0	0

65585 rows × 13 columns

Figure 19 Training Dataset

Algorithms and Techniques

We are trying to solve a supervised classification problem here where we are trying to predict the success of offers in Starbucks campaign.

As we will be looking at the following three algorithms. Scikit-learn library logistic regression, scikit-learn library random forest, and XGBoost algorithm provided by Amazon SageMaker.

Both Random Forest and XGBoost used decision tree as base learner and they can be used for both regression and classification problem. They both utilized ensemble methods which combine several decision trees to produce better predictive performance(k). All these three algorithms are well suited for our problems in this project, the logistic regression is chosen as a baseline model because it is easy to setup. We have good coverage of both the bagging and boosting techniques.

- **Logistic Regression** is a statistical method for predicting binary classes which are suitable for our problem. Logistic regression is quite easy to implement.(i)

- **Random Forests Classifier** is an ensemble of decision trees trained on randomly selected data samples, then the best prediction from each tree and select the best solution by voting. Random forest is an extension of the **bagging technique**. In this technique it also allows us to use a random subset of data, and random selection features rather than using all the features that we have, to grow the decision tree**.** (k)

- **XGBoost** is an efficient implementation of gradient boosting. XGBoost used gradient **boosting technique** decision tree algorithm. Gradient boosting is an algorithm that combines many weak learning models together to create a strong predictive model *(g)***.** It is prone to over fitting and requires careful tuning of different hyperparameters. XGboost is well known for its efficiency in learning model, and it works well on categorical data and limited datasets. (k)

Benchmark

In this project we have decided to use our starting Logistic regression model as our go to benchmark our model to measure our model performance as Logistic regression is quick and easy to implement. Our model performance should be performed better than the benchmark simple Logistic regression model. Our model benchmark here accuracy score and f1 score is 0.704429 and 0.702342 respectively.

We have also compared it with other contemporary work done in the same dataset that we found in the (reference sha821) which has a of accuracy score of 0.80 and an F1 score of 0.73 .

Methodology

Data Processing

We have documented the data preprocessing steps above (please refer to above EDA section), where we did the data transformation and prepared the merged dataset. We have removed missing data to ensure the quality of our model instead of using median values.

However, to prepare the data for the training, we have also taken the following steps that are necessary to prepare the data for the model training.

We have removed the *offerid and the customerid* as these columns are a *guid* which are not suitable for training. We also removed the derived data columns such as the *totalamount* and *tracsactioncount* as these are not a feature. This is the best time to remove the data that is not a good predictor as well such as the *offer_start_time*, *email* and *member_days*. Some features are a bad predictor as correlation does not always imply causation

Email is a bad predictor due to all the offers are distributed via email. Membership days are a bad predictor as we seen from the graph as we have a large left skew which mean that more customers become a member in the recent years, so this will give us false positive as where considerable number of transactions are made when you have when the user base grown. This is clearly a bias in the data that we should be removing.

Project Overview

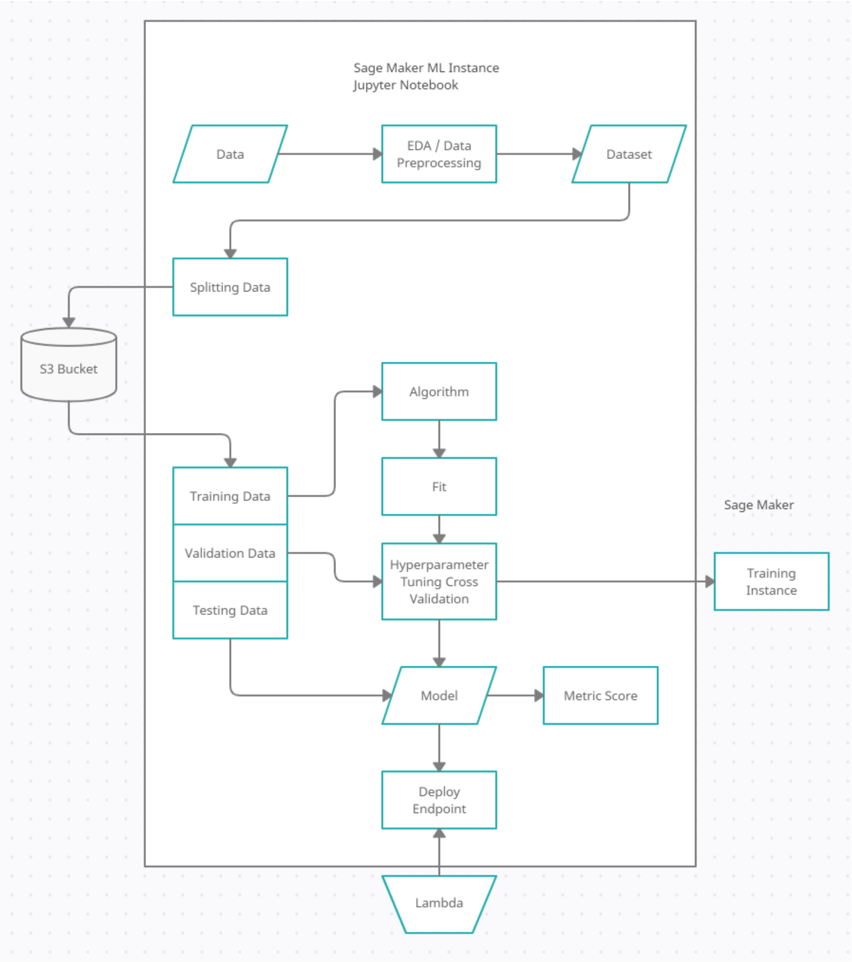


Figure 20 high level process flowchart for implementation steps

Data Splitting - With the processed data, we will then split the processed data into 6:2:2 training data, validation data and testing data using train_test_split function from scikit-learn. The training dataset will be used to train our algorithm, however, selecting the best performing algorithm using the test dataset will lead to biases and hence we needed a validation dataset to do cross-validation. The test dataset is used to calculate our accuracy score and F1 score.

To ensure repeatably of our code, we used the same fix `random_state` value throughout our project. We also removed the header and index from our merged data due to it being required to be removed by SageMaker.

Implementation of Logistic Regression and Random Forest

In is relatively simple to implement both the Logistic Regression and the Random Forest model, we can do this by initializing a Logistic Regression classifier or Random Forest classifier then proceed to train the model with the default parameters. We can then try to improve the model by introducing a randomized search cross validation. In here we trained a total of 100 models over 3 folds of cross validation with a total of 300 models. This performs a randomized search on hyper parameters and optimized by cross-validated the search over parameter settings. However, in contrast to the grid search, not all parameter values are tried out, but rather a fixed number of parameter settings is sampled from the specified distributions. We have opted to use the randomized search cross validation here as it is more efficient while producing a similar performance.

Implementation of XGBoost Model

The data that has been split via train_test_split method from Sklearn library is then uploaded to our predefined S3 bucket.

We first start by creating an Amazon Sagemaker estimator using the latest Amazon SageMaker XGBoost training containers. We also set the hyperparameter values for our XGboost Model.

We then setup a Sagemaker Hyperparameter Tuner and train a total of 10 models, of which a total of 3 models are being trained at the same time. We used the AUC area under the curve to compare the trained model here. Then we fit the XGBoost model using the hyperparameter tuner. Once the hyperparameter job is finished. We get the download the estimator that has the best training job attached. The trained model can then be deployed to an Amazon SageMaker endpoint and return a Sagemaker predictor object.

We then start a new transform job for our trained model to get the inferences on our dataset which are then saved to the Amazon S3. We then download the inferences data to our local environment as test.csv.out. This file will be our prediction object after we parse the csv file using panda which we can use to calculate our metric and make prediction with our test data,

As for complications that occurred during coding, we did encounter issues where we can't calculate our metric accuracy and f1score when using XGBoost with the objective binary:logistic. This is because it returns the prediction in a continuous value of prediction, e.g. [0.7,0.3,0.4,0.9] while our test value is in binary 1 and 0. We do have to convert the prediction value to binary 1 and 0 using a threshold value of 0.5. Prediction values that are higher than 0.5 are considered successful.

We can then deploy the trained model to an Amazon SageMaker endpoint and return a Sagemaker predictor object. We can call the endpoint using this predictor object by parsing in our values to get a prediction. The data returned will be in a comma delimited value and we need to convert this to numpy array before we can use it.

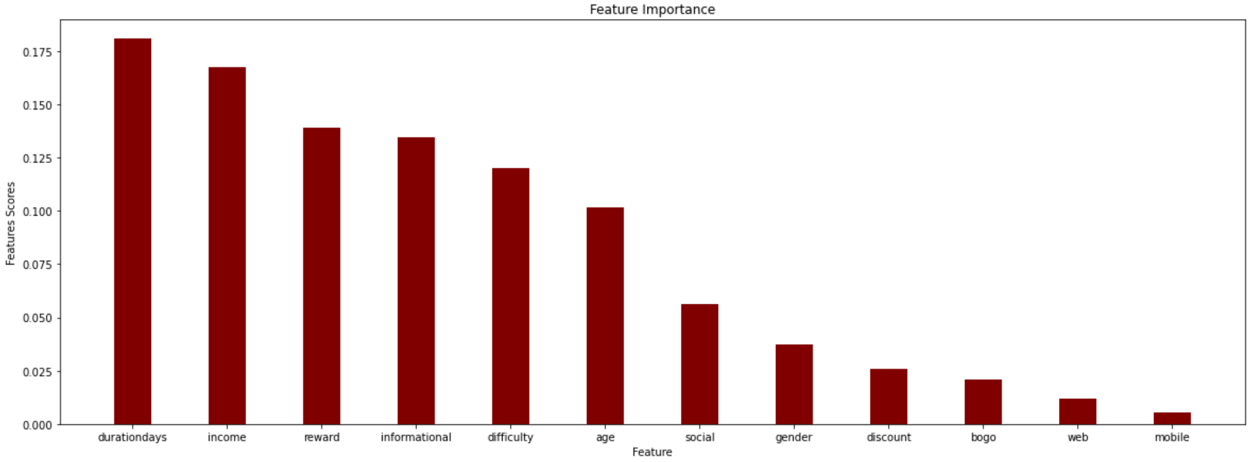


Figure 21 Feature Importance Graph

We have generated the feature importance graph from the best random forest model. From the feature importance graph, we determine that the duration days have the most predictive features score. Where the longer the offer (durationdays), the more successful an offer is going to be. The next feature is going to be the *informational* offer type, we know that *informational* offer type is the worst performing offer type there is. The next one on the list is *reward* followed by *income* and *difficulty*.

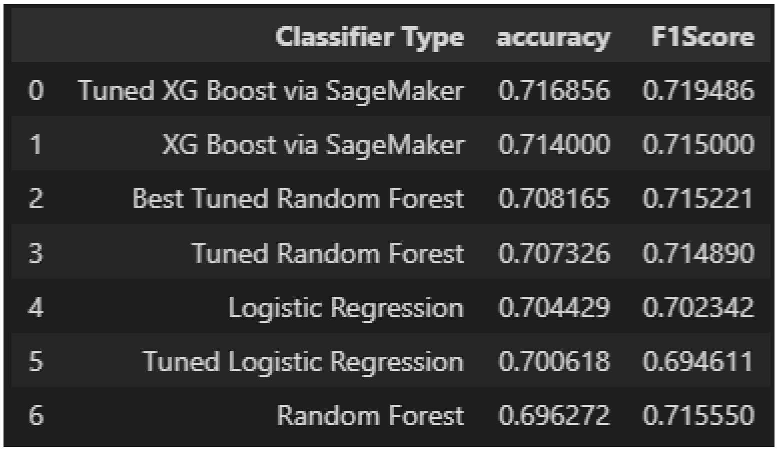
Based on this feature importance score alone, we realized that out of the top 5 feature importance are from the offer dataset, we could offer a longer duration offer type that is not an informational offer type, either a higher discount or a BOGO offer that have a lower difficulty to the higher income people.

This is also reflected by the offer success rate and amount histogram, where the longest duration offer has a higher success rate of offer and total amount gross profit generated. We can then create offers and data profiles data to be fitted into the model to predict if the success of an offer. We could also create promotional offers that are targeted at different genders as we have figured out that different genders have a tendency to a different type of offer as well as different spending power as well.

Refinement

With both the logistic regression models and the random forest, we have performed a randomized search cross validation across all these models. The best parameter for the RandomForestClassifier that we are using are (max_depth=10, max_features='sqrt', min_samples_split=5, n_estimators=250)

To refine the hyperparameter that we have for our best model, we have opted to do further randomized search with more hyperparameter instead of GridSearchCV as RandomizedSearchCV offer the best ratio between the speed and efficiency.



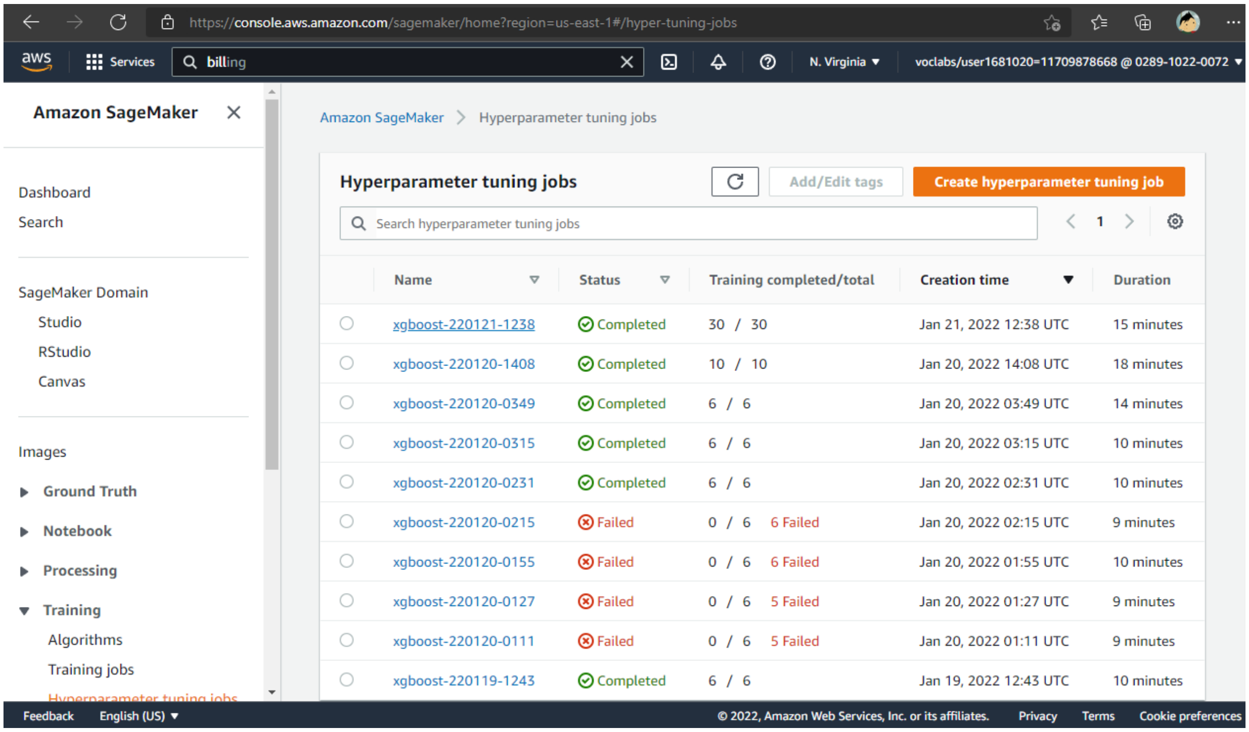
	Classifier Type	accuracy	F1Score
0	Tuned XG Boost via SageMaker	0.716856	0.719486
1	XG Boost via SageMaker	0.714000	0.715000
2	Best Tuned Random Forest	0.708165	0.715221
3	Tuned Random Forest	0.707326	0.714890
4	Logistic Regression	0.704429	0.702342
5	Tuned Logistic Regression	0.700618	0.694611
6	Random Forest	0.696272	0.715550

Figure 22 Comparison of all the Model and their metric score

We have further increased the accuracy from 0.707326 to 0.708165 and the F1 score from 0.714890 to 0.715000 for our Random Forest model after performing a further randomized search cross validation. The best tuned parameter for the RandomForestClassifier that we are using are (max_depth=12, max_features='sqrt', min_samples_split=5, n_estimators=250)

In the case of XGboost using SageMaker, we have utilized the HyperparameterTuner from SageMaker which allow us to run hyperparameter tuning job to select the best trained model. We have increased the range of the hyperparameter tuning and increased the number of jobs to get the best tune results. The hyperparameter ranges that we used are increased as followed 'max_depth 5 to 50, 'eta': 0.02 to 0.4, 'min_child_weight': 2 to 5, 'num_round' 400 to 800, 'subsample':0.60 to 0.80 and 'gamma 0.5, 3.5. Based on the highest auc score which is our tuning_objective_metric used here, our best hyperparameter was as followed.

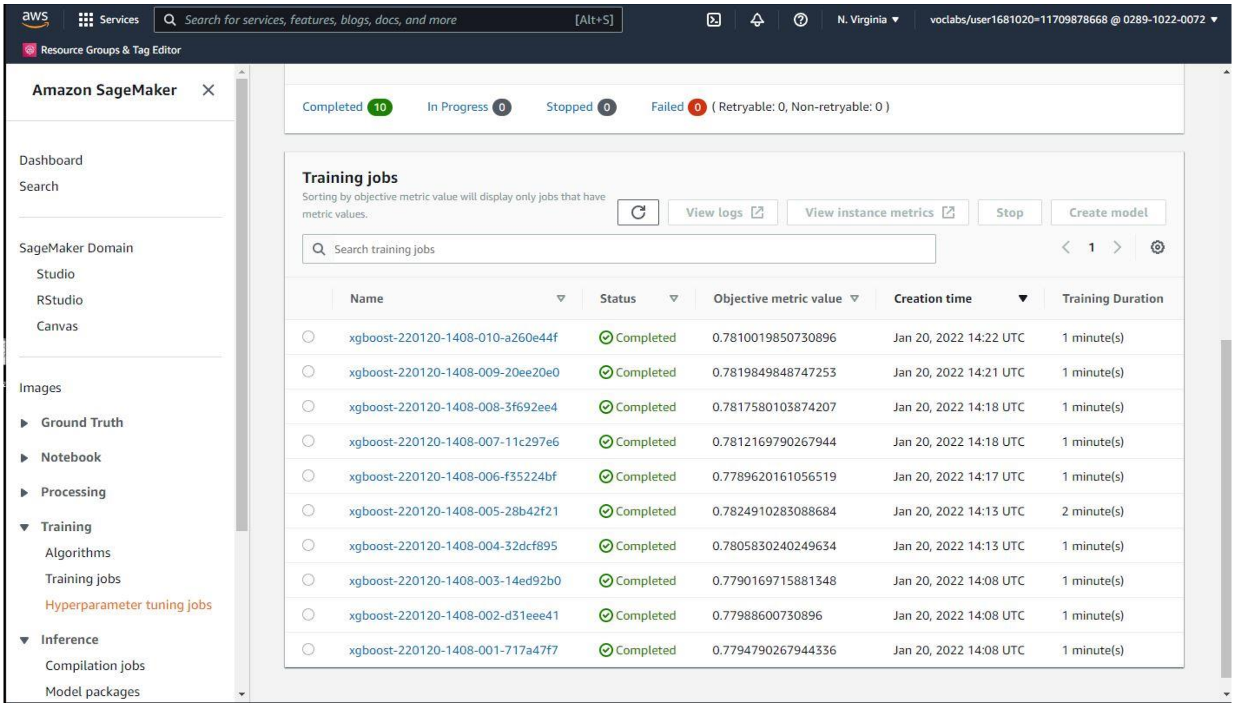
{'_tuning_objective_metric': 'validation:auc', 'early_stopping_rounds': '30', 'eta': '0.027217889534806458', 'gamma': '2.927982866805479', 'max_delta_step': '3', 'max_depth': '8', 'min_child_weight': '3', 'num_round': '730', 'objective': 'binary:logistic', 'silent': '0', 'subsample': '0.7979210498324821'}


https://console.aws.amazon.com/sagemaker/home?region=us-east-1#/hyper-tuning-jobs
aws
Services
billing
N. Virginia ▾
voclabs/user1681020=11709878668 @ 0289-1022-0072 ▾
Amazon SageMaker
Dashboard
Search
SageMaker Domain
Studio
RStudio
Canvas
Images
▸ Ground Truth
▸ Notebook
▸ Processing
▾ Training
Algorithms
Training jobs
Hyperparameter tuning jobs
Amazon SageMaker > Hyperparameter tuning jobs
Add/Edit tags
Create hyperparameter tuning job
Search hyperparameter tuning jobs
1
Feedback
English (US) ▾
© 2022, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

Hyperparameter tuning jobs

	Name	Status	Training completed/total	Creation time	Duration
○	xgboost-220121-1238	Completed	30 / 30	Jan 21, 2022 12:38 UTC	15 minutes
○	xgboost-220120-1408	Completed	10 / 10	Jan 20, 2022 14:08 UTC	18 minutes
○	xgboost-220120-0349	Completed	6 / 6	Jan 20, 2022 03:49 UTC	14 minutes
○	xgboost-220120-0315	Completed	6 / 6	Jan 20, 2022 03:15 UTC	10 minutes
○	xgboost-220120-0231	Completed	6 / 6	Jan 20, 2022 02:31 UTC	10 minutes
○	xgboost-220120-0215	Failed	0 / 6 6 Failed	Jan 20, 2022 02:15 UTC	9 minutes
○	xgboost-220120-0155	Failed	0 / 6 6 Failed	Jan 20, 2022 01:55 UTC	10 minutes
○	xgboost-220120-0127	Failed	0 / 6 5 Failed	Jan 20, 2022 01:27 UTC	9 minutes
○	xgboost-220120-0111	Failed	0 / 6 5 Failed	Jan 20, 2022 01:11 UTC	9 minutes
○	xgboost-220119-1243	Completed	6 / 6	Jan 19, 2022 12:43 UTC	10 minutes

Figure 26 Hyperparameter Tuning job in sagemaker



Completed 10 In Progress 0 Stopped 0 Failed 0 (Retryable: 0, Non-retryable: 0)

Training jobs

Sorting by objective metric value will display only jobs that have metric values.

	Name	Status	Objective metric value	Creation time	Training Duration
○	xgboost-220120-1408-010-a260e44f	Completed	0.7810019850730896	Jan 20, 2022 14:22 UTC	1 minute(s)
○	xgboost-220120-1408-009-20ee20e0	Completed	0.7819849848747253	Jan 20, 2022 14:21 UTC	1 minute(s)
○	xgboost-220120-1408-008-3f692ee4	Completed	0.7817580103874207	Jan 20, 2022 14:18 UTC	1 minute(s)
○	xgboost-220120-1408-007-11c297e6	Completed	0.7812169790267944	Jan 20, 2022 14:18 UTC	1 minute(s)
○	xgboost-220120-1408-006-f35224bf	Completed	0.7789620161056519	Jan 20, 2022 14:17 UTC	1 minute(s)
○	xgboost-220120-1408-005-28b42f21	Completed	0.7824910283088684	Jan 20, 2022 14:13 UTC	2 minute(s)
○	xgboost-220120-1408-004-32dcf895	Completed	0.7805830240249634	Jan 20, 2022 14:13 UTC	1 minute(s)
○	xgboost-220120-1408-003-14ed92b0	Completed	0.7790169715881348	Jan 20, 2022 14:08 UTC	1 minute(s)
○	xgboost-220120-1408-002-d31eee41	Completed	0.77988600730896	Jan 20, 2022 14:08 UTC	1 minute(s)
○	xgboost-220120-1408-001-717a47f7	Completed	0.7794790267944336	Jan 20, 2022 14:08 UTC	1 minute(s)

Figure 27 Training Jobs Trained by Hyperparameter Tuning

Results

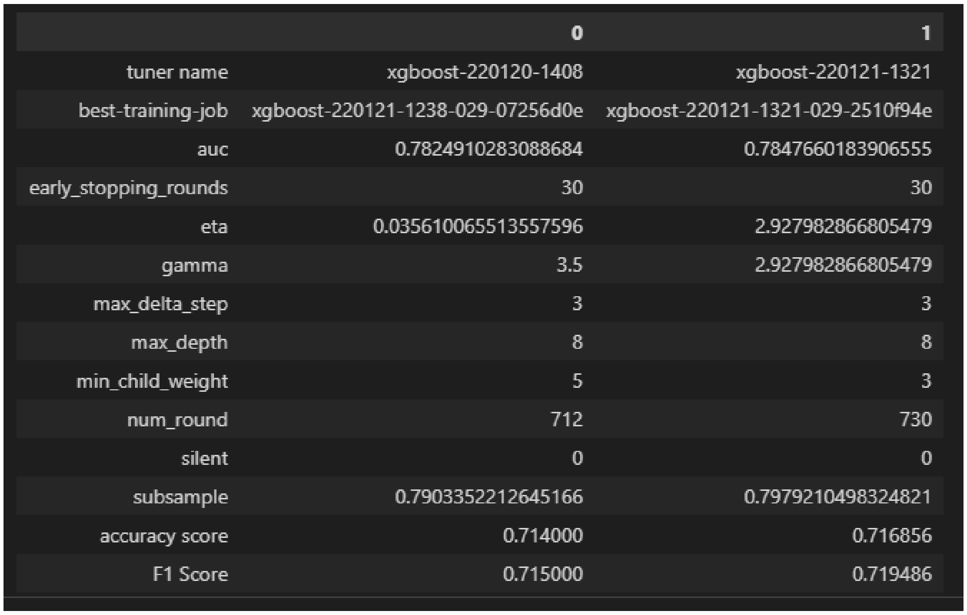
Model Evaluation and Validation

We assessed the final model we chose using test data, which is independent from the training data, which represents our new offer and customer profile data set. It outperformed the training data set. This indicates that the model was able to discover patterns in the dataset across three different datasets: training, validation, and test.

In the case of our main Model XGboost using SageMaker, we have utilized the HyperparameterTuner from SageMaker which allow us to run hyperparameter tuning job to select the best trained model. XGBoost provides large range of hyperparameters which we can tune to improve the performance. We try not to increase the max_depth too much to prevent overfitting. We used the auc (area under the curve) metric to compare the trained models.

Based on the highest auc score which is our tuning_objective_metric used here, our best hyperparameter was as follow.

{'_tuning_objective_metric': 'validation:auc', 'early_stopping_rounds': '30', 'eta': '0.027217889534806458', 'gamma': '2.927982866805479', 'max_delta_step': '3', 'max_depth': '8', 'min_child_weight': '3', 'num_round': '730', 'objective': 'binary:logistic', 'silent': '0', 'subsample': '0.7979210498324821'}



	0	1
tuner name	xgboost-220120-1408	xgboost-220121-1321
best-training-job	xgboost-220121-1238-029-07256d0e	xgboost-220121-1321-029-2510f94e
auc	0.7824910283088684	0.7847660183906555
early_stopping_rounds	30	30
eta	0.035610065513557596	2.927982866805479
gamma	3.5	2.927982866805479
max_delta_step	3	3
max_depth	8	8
min_child_weight	5	3
num_round	712	730
silent	0	0
subsample	0.7903352212645166	0.7979210498324821
accuracy score	0.714000	0.716856
F1 Score	0.715000	0.719486

Figure 28 Comparison between the two tuner best training job hyperparameter and the improvement of metric score

Figure Comparison of two hyperparameter tuner in XGboost showing an improvement in both the accuracy score and F1Score.

We have increased the continuous range of our hyperparameter and increased the number of max_jobs from 10 to 40 to allow more training to improve the performance,

The accuracy score and the F1 score increased in the subsequent tuning, and we can see that the eta is increased from 0.035 to 2.927, gamma is reduced to 2.92 from 3.5 and a slight increase of num round. The Gamma is the minimum loss reduction to create a partition, a larger gamma means it is more conservative. The eta is the learning rate which increase our modeling speed, so that we need less round. However, in order to get the best model performance, we should increase set the eta as 0.01 and increase the number of rounds and the max jobs as much as possible.

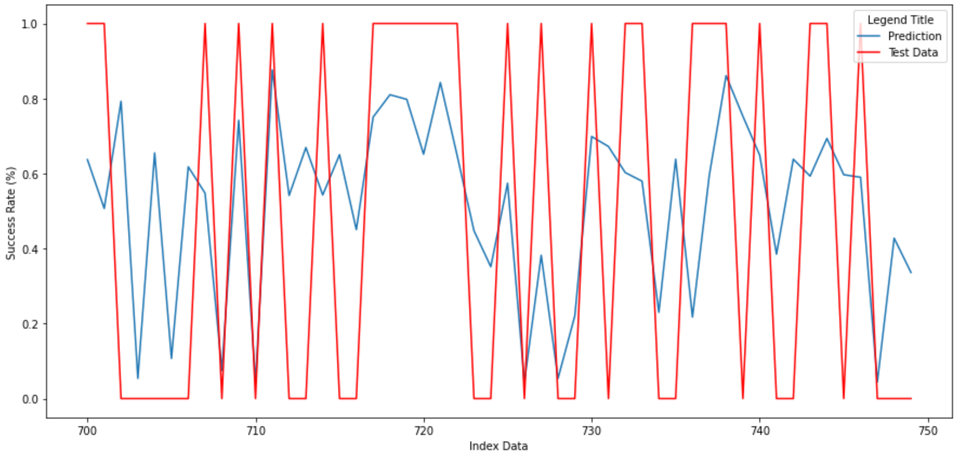
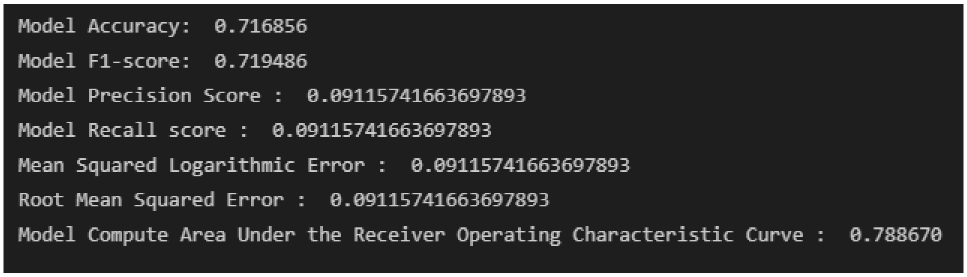


Figure 29 Model performance graph based on the test data to visualize the offer success rate and the prediction score outcome.

Our best performing XGBoost model have an AUC score of 0.785, and we have satisfied that we have selected the best model with overall performance. We have a very low Root Mean Square Error which indicates good fit.



```
Model Accuracy:  0.716856
Model F1-score:  0.719486
Model Precision Score :  0.09115741663697893
Model Recall score :  0.09115741663697893
Mean Squared Logarithmic Error :  0.09115741663697893
Root Mean Squared Error :  0.09115741663697893
Model Compute Area Under the Receiver Operating Characteristic Curve :  0.788670
```

Figure 29 Best XGboost Metric data showing the performance of our model.

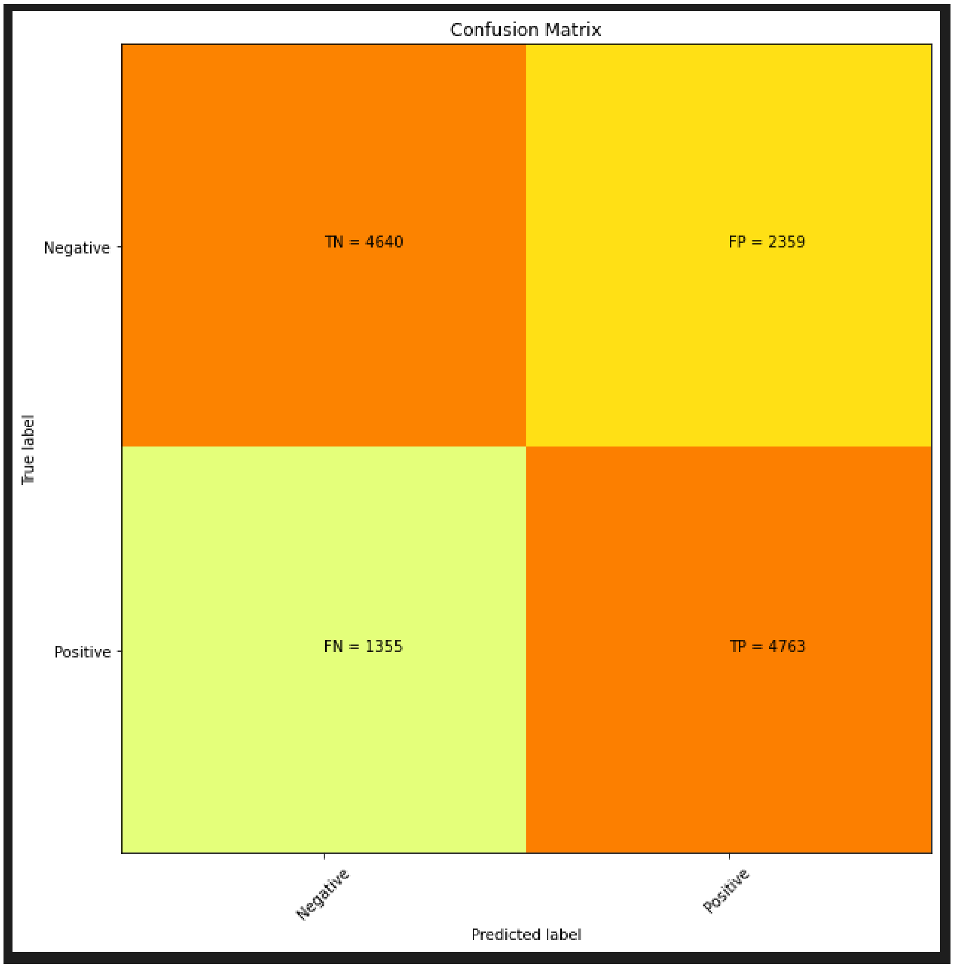
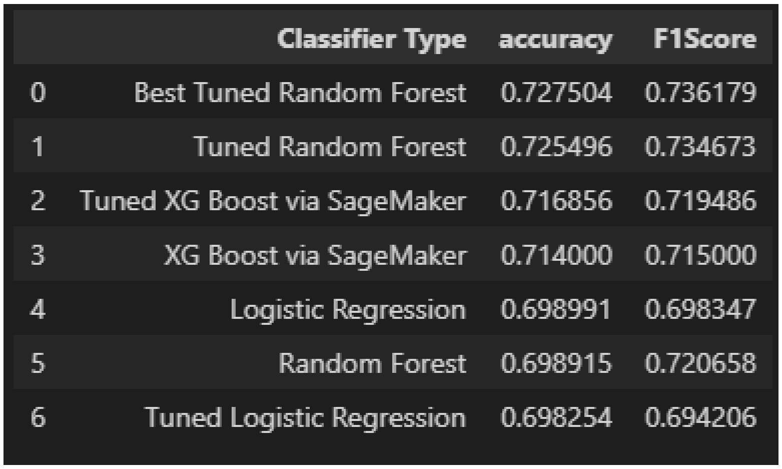


Figure 30 Best XGboost confusion matrix.

Justification

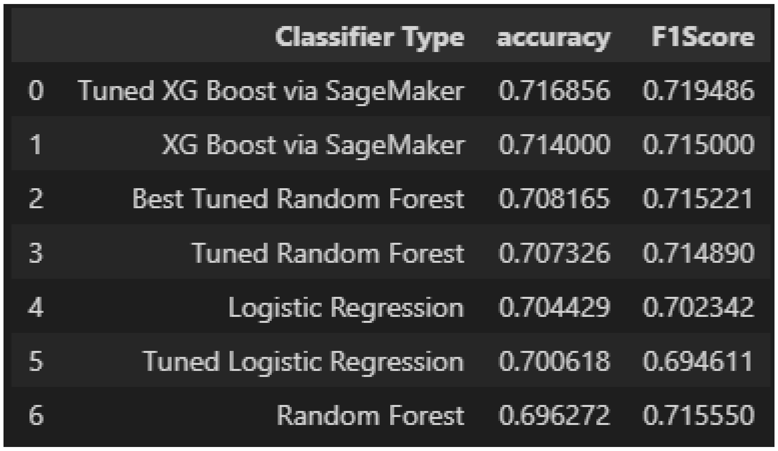
We then compare the accuracy and the F1 score of all the models. Based on the results, we can see that we have the highest accuracy score and F1 score with the Random Forest classifier with training data, and highest score from XGBoost model with testing data.

Comparison of metric score on both the training data set to investigate overfitting,



	Classifier Type	accuracy	F1Score
0	Best Tuned Random Forest	0.727504	0.736179
1	Tuned Random Forest	0.725496	0.734673
2	Tuned XG Boost via SageMaker	0.716856	0.719486
3	XG Boost via SageMaker	0.714000	0.715000
4	Logistic Regression	0.698991	0.698347
5	Random Forest	0.698915	0.720658
6	Tuned Logistic Regression	0.698254	0.694206

Figure 31 Metric Score from training dataset



	Classifier Type	accuracy	F1Score
0	Tuned XG Boost via SageMaker	0.716856	0.719486
1	XG Boost via SageMaker	0.714000	0.715000
2	Best Tuned Random Forest	0.708165	0.715221
3	Tuned Random Forest	0.707326	0.714890
4	Logistic Regression	0.704429	0.702342
5	Tuned Logistic Regression	0.700618	0.694611
6	Random Forest	0.696272	0.715550

Figure 32 Metric Score from test dataset

The Tuned XG Boost Accuracy and F1 Score are performing well, and it is safe to say that we did not overfit our model, overfitting is a scenario where the model performs well on training data but performs poorly on data not seen during training. This usually happens when the model has memorized the training data instead of learning the relationships between features and labels. Our metric accuracy score and F1 Score are performing similarly good with a score of about 0.7 on average.

We do expect the XGBoost to perform better although it is usually harder to tune as it has a larger number of hyperparameter to tune and easier to overfit the training data. The random forest is easier to tune as we only mainly need to tune the number of features to randomly select from a set of features. All the models that we implemented have a similar performance and can predict at around 71.6% of the success rate of an offer based on our input offer and demographic profile data.

Our final accuracy score and F1 score of our highest performing model XGBoost was 0.716856 and 0.719486 when compared to our benchmark model was higher than our simple logistic regression model of 0.7044 and 0.7023. If we compare our best model XGBoost with the other models, it gives a similar result when compared to all other models that we have run so far.

When we compare our performance to the benchmark model of contemporary works found on similar datasets, accuracy score of 0.80 and an F1 score of 0.73, our performance is comparable to them. The benchmark model of this contemporary works work used a grid search cross validation in their work which could lead to a slightly better improvement metric score. Using GridSearch could lead to a better model performance if we have the time and processing power.

I have also decided to use traditional supervised training classifier as this is a simple project however we could theoretically further improve the results by utilizing AutoGluon tabular prediction to explore performance of different models, however we might be spending more time on hyperparameter tuning.

Conclusion

It has been a wonderful experience for me working on this project because I have been able to put my newfound knowledge of this course into practice. This project has an open objective as there is no set way for us to achieve our capstone project. We can freely plan our own goal. This has been an invaluable experience for me as a capstone project provides me with a real-life example project with a very tight project schedule.

We have primarily looked at and built a model to look at predict if an offer is successful or not, however as we discovered in the histogram data above, the amount of gross profit is also important. And hence we could also be looking at the total amount of gross profit, the number of transactions and the offer success ratio of the amount of profit return and the success rate. An offer might be successful, but the total amount of purchase might be low.

References

a) Starbucks Industry AI Case Study (Domain Background)

https://info.formation.ai/rs/435-BMS-371/images/Starbucks_CaseStudy_final.pdf

b) Starbucks Industry (Domain Background)

https://brainstation.io/magazine/digital-loyalty-lifts-starbucks-q1-to-record-7-1-billion

c) Starbucks Industry (Domain Background)

https://www.geekwire.com/2021/quarter-starbucks-orders-u-s-now-paid-smartphone/

d) AI in marketing

https://www.sciencedirect.com/science/article/pii/S2667096820300021

e) Logistic Regression Definition.

https://www.datacamp.com/community/tutorials/understanding-logistic-regression-python

f) RandomForest Definition.

https://www.datacamp.com/community/tutorials/random-forests-classifier-python

g) XGBoost https://docs.aws.amazon.com/sagemaker/latest/dg/xgboost-HowItWorks.html

h) Accuracy vs. F1-Score.

https://medium.com/analytics-vidhya/accuracy-vs-f1-score-6258237beca2

i) Baseline Model from contemporary works done on the same dataset.

https://medium.com/@sha821/starbucks-capstone-project-4b1eb8015bee

j) Udacity Starbucks Capstone Challenge Notebook and Dataset

k) Differences Between Random Forest vs XGBoost https://www.educba.com/random-forest-vs-xgboost/

l) Logistic regression https://careerfoundry.com/en/blog/data-analytics/what-is-logistic-regression/