Machine Learning Engineer Nanodegree

Udacity

Starbucks Capstone Challenge
Project Report

Aya Tarek Ali
May 17th, 2020

I. Definition

a. Project Overview

This project is based on simulated data of customer behavior and offers reactions driven from Starbucks rewards mobile APP.
Starbucks periodically sends offers and advertisement information about the new products to the app users. These offers can be one of three types: Buy One Get One (BOGO), a special discount, or an informational message and some users may not receive any offers for certain period. Each of these offers has a validity period, even the informational ones.

Completed offers don't mean necessarily that they were actually completed, for example: a customers may receive an offer "Spend 10 dollars and receive free 10 dollars to spend", but never really views this offer, then the customer spends 12 dollars so there will be an offer completion record, however the customer wasn't influenced by that offer because he didn't view the offer.

Understanding the customer behavior will help the business in taking decisions regarding who to target with new offers and discounts which means spending money on campaigns more wisely, and also targeting customers who won't respond to any offer by creating newly customized offers to change their mind.

This is a very rich subject and can be used in taking lots of decisions, and to solve multiple problems starting from which offer to target the customer to whether the customer will respond to offers or not and it can be improved by collecting and introducing new features.

The dataset in this project is provided by Udacity and Starbucks, and it has features that describes the demographic information about different customers, historical transactions made by these customers and the kind of offers they received and how they responded to each one of them along with the offers features that describes the difficulty, type, reward, durations and channels that were used in campaigning that offer.

b. Problem Statement

Merchants like Starbucks spend money on marketing campaigns, however not all customers respond to the offers the same way. Some customers don't view the offers because they are opposed to marketing campaigns, others may be not interested in that particular offer they received.

Each offer sent and responded to due to any of the reasons I've mentioned before means a wasted resource and a loss in profit if that offer was sent to the right person instead.

The problem here is to identify whether the customer will respond and complete these offers or not, so we can only target customers who will more likely complete them, hence increase the profit.

My approach for solving this issue, will be to predict the customer response based on his demographic group features. I will use machine learning techniques to study and learn from the customers and offers features and the relationships between them.

I will train three different models: Logistic Regression, Decision Tree and Random Forest and evaluate each one of them and use them to predict the customer response and select the best model based on an evaluation metrics that will be described later.

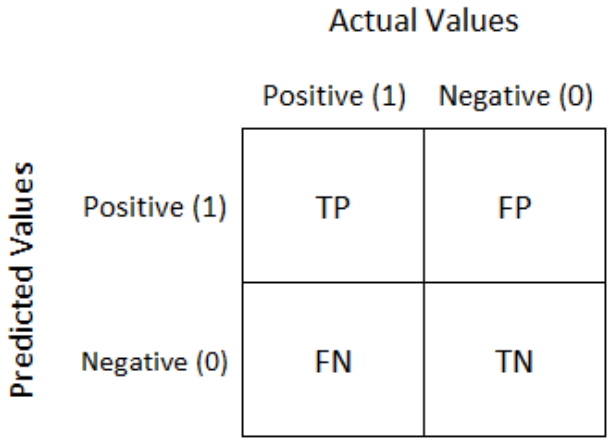
The strategy for solving this problem will include the below steps:

1. **Data loading and understanding**
   We will load the data files and explore what data we have by visualizing and printing the data features.
2. **Data cleaning and preparation**
   We need to remove the null values, remove the outliers and apply one hot encoding if needed.
3. **Data analysis**
   After cleaning and preparing the data, we need to combine the three files into one dataset and start visualizing the each feature to understand the relation between the customer profile, transactions and the offers he receive.
4. **Splitting the data**
   First we split the data into training and testing datasets, then apply the cross validation over the training to find the best folds with the higher accuracy to be used in training and evaluating the models.
5. **Define and train the models**
6. **Save the trained models**
7. **Load and test the models**
8. **Evaluate and compare the accuracy & confusion matrix**

c. Metrics

The evaluation metrics of the three models will be compared using the same metric, we will compare the Precision, Recall, Confusion Matrix and Accuracy% and the counts of TP, TN, FP and FN.

A high accuracy that is close to 100% means that the model is overfitting, so my accuracy threshold will be 75% and the lowest number of FN and the highest number of TP as well. The low number of FN, means that we are not discarding customers who will respond to an offer by incorrectly predicting they will not.



II. Analysis

a. Data Exploration and Visualization

In this part we will explore the data in order to understand the features and the characteristics of each one of them and how does it affect the decision we need to make.

We will use data visualization to extract relations between different features, we will also apply some data engineering techniques to help identify any inconsistency in the data and to exclude the unnecessary features.

Then we will use the final features to construct the training data and feed our models with.

The data that we will use in this project is contained in three files:

1. **Portfolio.json**

   Contains the offer ids and Meta data about each offer, there are 10 different offers with 6 features:

   - id (string) - offer id
   - offer_type (string) - type of offer (BOGO, discount, informational)
   - difficulty (int) - minimum required spend to complete an offer
   - reward (int) - reward given for completing an offer
   - duration (int) - time for offer to be open, in days (validity)
   - channels (list of strings) - web, email, mobile, social,…

```
1- portfolio Data exploration

Data shape (rows, cols): (10, 6)

Missing data? False

Data description:
```

	difficulty	duration	reward
count	10.000000	10.000000	10.000000
mean	7.700000	6.500000	4.200000
std	5.831905	2.321398	3.583915
min	0.000000	3.000000	0.000000
25%	5.000000	5.000000	2.000000
50%	8.500000	7.000000	4.000000
75%	10.000000	7.000000	5.000000
max	20.000000	10.000000	10.000000

```
Data Sample:
```

	channels	difficulty	duration	id	offer_type	reward
0	[email, mobile, social]	10	7	ae264e3637204a6fb9bb56bc8210ddfd	bogo	10
5	[web, email, mobile, social]	7	7	2298d6c36e964ae4a3e7e9706d1fb8c2	discount	3
6	[web, email, mobile, social]	10	10	fafdcd668e3743c1bb461111dcafc2a4	discount	2
9	[web, email, mobile]	10	7	2906b810c7d4411798c6938adc9daaa5	discount	2
8	[web, email, mobile, social]	5	5	f19421c1d4aa40978ebb69ca19b0e20d	bogo	5
2	[web, email, mobile]	0	4	3f207df678b143eea3cee63160fa8bed	informational	0
3	[web, email, mobile]	5	7	9b98b8c7a33c4b65b9aebfe6a799e6d9	bogo	5
7	[email, mobile, social]	0	3	5a8bc65990b245e5a138643cd4eb9837	informational	0
1	[web, email, mobile, social]	10	5	4d5c57ea9a6940dd891ad53e9dbe8da0	bogo	10
4	[web, email]	20	10	0b1e1539f2cc45b7b9fa7c272da2e1d7	discount	5

```
offer type column unique values:
['bogo' 'informational' 'discount']
```

2. profile.json

Contains the demographic data for 17,000 customers, each with 5 features:

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

- id (str) - customer id
- income (float) - customer's income, Notice that the missing record has age=118, gender=none and income =NaN.

```
2- Profile Data exploration

Data shape (rows, cols): (17000, 5)

Missing data? True

Data description:
```

	age	became_member_on	income
count	17000.000000	1.700000e+04	14825.000000
mean	62.531412	2.016703e+07	65404.991568
std	26.738580	1.167750e+04	21598.299410
min	18.000000	2.013073e+07	30000.000000
25%	45.000000	2.016053e+07	49000.000000
50%	58.000000	2.017080e+07	64000.000000
75%	73.000000	2.017123e+07	80000.000000
max	118.000000	2.018073e+07	120000.000000

	age	became_member_on	gender	id	income
3107	21	20170501	M	f444db6612f649c79084b88b7e1eb83f	31000.0
6796	118	20171201	None	13bb9b5a8be748d6ae7ed5ad3ce902a6	NaN
8027	41	20170102	F	0d2c8f6e2cd441bbbbfaf0036c41630d	80000.0
11475	27	20170920	M	e19f1901e79f4e35a8343c73184982f6	32000.0
11345	118	20140127	None	e9ea9187fd4a41f59531f59dd3a511e3	NaN
4912	60	20161030	M	6c1f1d11a0f344a58559c93bf519963f	33000.0
15548	68	20170115	F	52ca1a63c646427d8fe5b1c2c633c86c	51000.0
16211	118	20171108	None	cd8a88670e404eef9d4edc3010dd84aa	NaN
16169	69	20171121	F	a8adc69426a34064b249082dd3d8cada	54000.0
12508	45	20160101	F	ab77eeb4f3624d6dbb679f8f767ba06d	69000.0

```
gender column unique values:
[None 'F' 'M' 'O']
```

- Notice that there are some missing records with age=118, gender=none and income =NaN.
- Min income is 30,000 and max is 120,000

- Min age is 18 and max is 118

transcript.json

Contains data about the customer transactions and offer status for 306534 events with 4 features:

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

```
3- transcript Data exploration

Data shape (rows, cols): (306534, 4)

Missing data? False

Data description:
```

	time
count	306534.000000
mean	366.382940
std	200.326314
min	0.000000
25%	186.000000
50%	408.000000
75%	528.000000
max	714.000000

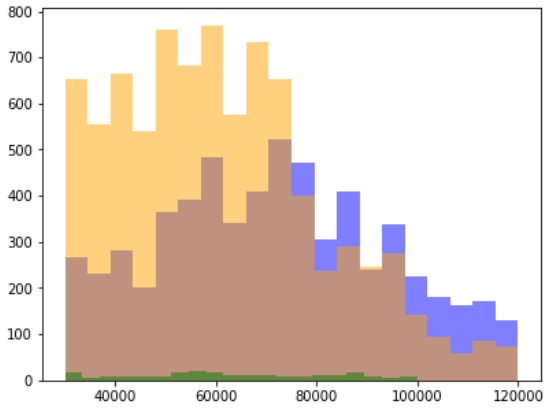
	event	person	time	value
95004	offer completed	9ec21a8257b54fc08d10726f5bc21a09	246	{'offer_id': 'f19421c1d4aa40978ebb69ca19b0e20d...
180607	offer viewed	a10fcf3dacaf4bedacc1626f26c15311	438	{'offer id': 'f19421c1d4aa40978ebb69ca19b0e20d'}
204533	offer received	11e2d9f6ef474d13bcf1db21f5c7bb62	504	{'offer id': 'ae264e3637204a6fb9bb56bc8210ddfd'}
148246	offer viewed	60e115418b204beea2ca20d3cc82e65d	396	{'offer id': '2298d6c36e964ae4a3e7e9706d1fb8c2'}
192426	transaction	402c544727ae422bbe2d1ad7efcaf49a	468	{'amount': 23.29}
80234	transaction	4b54746836c34bc9a0db105b7d144de8	198	{'amount': 3.04}
55024	offer received	b784713fea5f44f794f87259662eb766	168	{'offer id': 'ae264e3637204a6fb9bb56bc8210ddfd'}
295541	transaction	b51ff766c190452790e95b8066264e80	666	{'amount': 0.09}
296377	transaction	9c48c9b259b042f2b7cebe52df7fc36b	672	{'amount': 3.7}
278310	transaction	2ab61c2469b74c17b79992f4aaeee110	612	{'amount': 4.39}

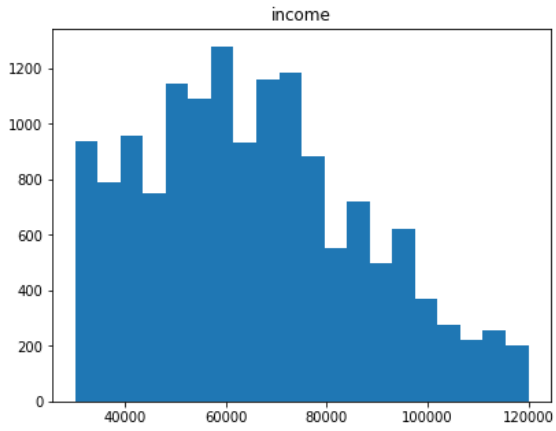
```
event column unique values:
['offer received' 'offer viewed' 'transaction' 'offer completed']
```

```
transaction        138953
offer received      76277
offer viewed        57725
offer completed     33579
Name: event, dtype: int64
```

- No missing rows in the data.
- Some records corresponds to the customer id with age =118, and since we're removing them from the profile data we need to drop their transactions as well.
- Completed offers represent 44% from the total received.

Exploring the Income:



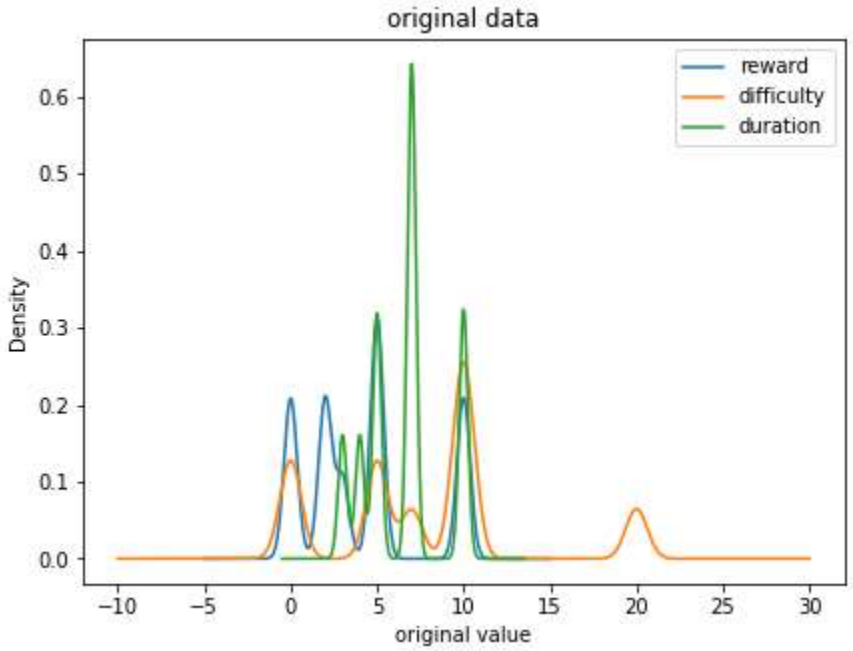


The above histogram shows that:

- Men's income is concentrated between 30K and 80K
- Women have higher % of having an income >=80K, which may impact the offer response rate

- From the right histogram, I think we can separate the income to categories each with width of 20K, starting with less than 50K to less than 130K

Exploring the reward, difficulty, duration:

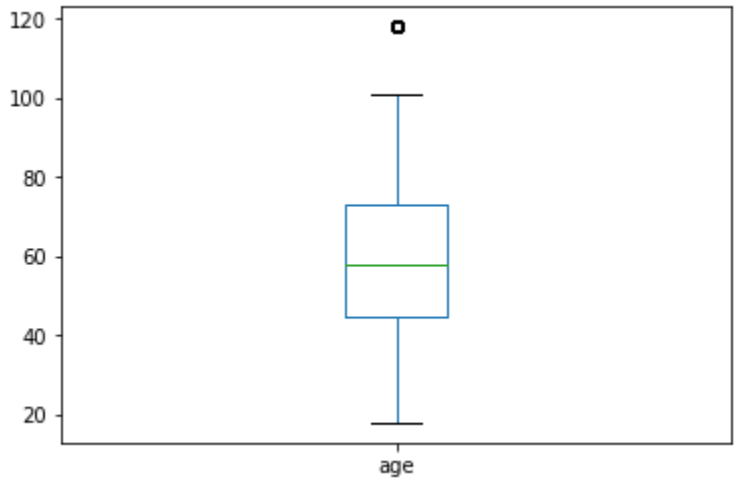


The above density plot shows that:

- We will need to scale some features such as difficulty, duration and reward as we don't want them to impact the training as they have large values than the other features we created of 0 and 1.

Exploring the Age:

The above density plot shows that:



- We have an outlier value of 118 in the age column that must be excluded, so it won't impact the training data.

b. Algorithms and Techniques

As we noticed during the exploration of the data, we have multiple features to preprocess before feeding the training model with.

Our main target is to transform any string values with decimal ones, 1s or 0s, remove the outliers and scale the features, so they won't impact the training by gaining more importance than other features with relatively small values.

For example, we have the offer_type column that we will need to separate into multiple columns and use the "one-hot encoding" technique to assign 1s and 0s against the corresponding offer.

After applying the mentioned steps, I will then group the three datasets by person and offer_id and start working on my "label" column, which will be the column that will be considered as the final customer response that I'm predicting. I will construct the "label" column using the "offer completed" column, as an indication that the customer will respond to this offer.

I'm aware that not all completed offers are actually viewed, but from a business point of view, since this customer is willing to spend the offer difficulty amount, I will consider this customer to be targeted by the campaign.

Now, to predict the customer response, I choose to train **Logistic Regression Classifier** as my main model.

c. Benchmark

For my benchmark results I will train two additional models: **Decision Tree** and **Random Forest.**

The three models will be compared using the same metric, we will compare the Precision, Recall, Confusion Matrix and Accuracy% and the counts of TP, TN, FP, and FN. A high accuracy that is close to 100% means that the model is overfitting, so my accuracy threshold will be 75% and the lowest number of FN and the highest number of TP as well.

III. Methodology

a. Data Preprocessing

First, we will apply data preprocessing on each of the three datasets:

1. portfolio

- Change the id column name to offer_id.
- Convert the offer_type labels to multiple columns:
  - bogo
  - discount
  - informational
- Separate the channels column into multiple channels and replace with binary labels
- delete the email feature, since all offers are sent via email

	difficulty	duration	offer_id	reward	bogo	discount	informational	mobile	social	web
0	10	7	ae264e3637204a6fb9bb56bc8210ddfd	10	1	0	0	1	1	0
1	10	5	4d5c57ea9a6940dd891ad53e9dbe8da0	10	1	0	0	1	1	1
2	0	4	3f207df678b143eea3cee63160fa8bed	0	0	0	1	1	0	1
3	5	7	9b98b8c7a33c4b65b9aebfe6a799e6d9	5	1	0	0	1	0	1
4	20	10	0b1e1539f2cc45b7b9fa7c272da2e1d7	5	0	1	0	0	0	1
5	7	7	2298d6c36e964ae4a3e7e9706d1fb8c2	3	0	1	0	1	1	1
6	10	10	fafdcd668e3743c1bb461111dcafc2a4	2	0	1	0	1	1	1
7	0	3	5a8bc65990b245e5a138643cd4eb9837	0	0	0	1	1	1	0
8	5	5	f19421c1d4aa40978ebb69ca19b0e20d	5	1	0	0	1	1	1
9	10	7	2906b810c7d4411798c6938adc9daaa5	2	0	1	0	1	0	1

2. profile

- Change the id column name to person.
- Remove the missing records with age=118, gender=none and income=NaN.
- Convert age to categories 10s, 20s, 30s and so on.
- Extract the year of join from the became_member_on to group by, no need for keeping the day or month
- Convert income to categories, each with width of 20K, starting with less than 50K to less than 130K
- Convert the gender to columns F, M and O

	person	10s_age	20s_age	30s_age	40s_age	50s_age	60s_age	70s_age	80s_age	90s_age	100s_age	income<=50000
1	0610b486422d4921ae7d2bf64640c50b	0	0	0	0	1	0	0	0	0	0	0
3	78afa995795e4d85b5d9ceeca43f5fef	0	0	0	0	0	0	1	0	0	0	0
5	e2127556f4f64592b11af22de27a7932	0	0	0	0	0	1	0	0	0	0	0
8	389bc3fa690240e798340f5a15918d5c	0	0	0	0	0	1	0	0	0	0	0
12	2eeac8d8feae4a8cad5a6af0499a211d	0	0	0	0	1	0	0	0	0	0	0
13	aa4862eba776480b8bb9c68455b8c2e1	0	0	0	0	0	1	0	0	0	0	0
14	e12aeaf2d47d42479ea1c4ac3d8286c6	0	1	0	0	0	0	0	0	0	0	1
15	31dda685af34476cad5bc968bdb01c53	0	0	0	0	0	1	0	0	0	0	0
16	62cf5e10845442329191fc246e7bcea3	0	0	0	1	0	0	0	0	0	0	0
18	6445de3b47274c759400cd68131d91b4	0	0	0	0	1	0	0	0	0	0	1

income<=70000	income<=90000	income<=110000	income<=130000	joined_year	F	M	O
0	0	0	1	2017	1	0	0
0	0	1	0	2017	1	0	0
1	0	0	0	2018	0	1	0
1	0	0	0	2018	0	1	0
1	0	0	0	2017	0	1	0
1	0	0	0	2017	1	0	0
0	0	0	0	2014	0	1	0
0	1	0	0	2016	1	0	0
1	0	0	0	2014	0	1	0
0	0	0	0	2017	0	1	0

3. transcript

- Separate the value column data into multiple columns
- One hot encoding for the event column
- I will create a new column to indicate whether this offer was successfully responded to or not, 1 for success and 0 for not.
- Group the responses or number of times the offer was received, viewed and completed per customer in one row, since our problem is to identify which promo to send to which customer, we need to represent the customer responses to offers more clear than it is in the transcript data. We need to have one record per "customer and offer" that concludes whether the customer responded to that specific offer or not.

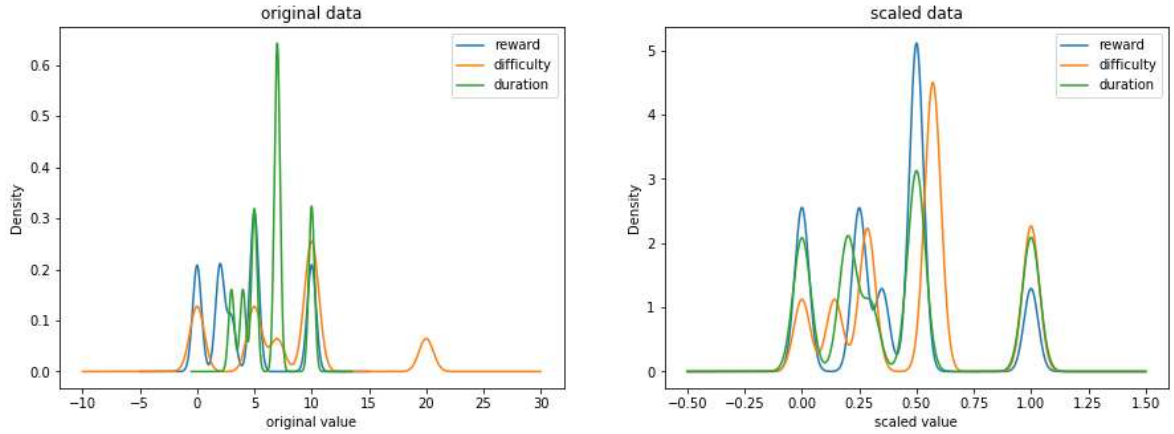
Now let's join all data sets together and take a more generalized look at the features.

Now we have finished all the data cleaning and merging, we can make final check and combine the needed features, apply some features scaling and check the correlation between the features, but before this I will filter out the below features:

- We don't need the offer received, offer viewed and transactions features anymore as we only care if the offer was responded to or not.
- Remove the person column as the model will work on the customer info and it doesn't really care about a specific customer.
- Informational offers can only be viewed, so it will impact the model if one type of offers always had the same response from the customer.

Preparing for training (Scaling)

To prepare the data we will need to scale some features such as difficulty, duration and reward as we don't want them to impact the training as they have large values than the other features we created of 0 and 1.



Preparing for training (Correlation Matrix)

Now let's take a look on the correlation between the features we chose, to make sure we excluded all the highly correlated features.

The highest correlation is between the difficultly and reward features = 0.8, which is a very acceptable ratio, so we won't exclude any more features.

	difficulty	duration	reward	bogo	discount	mobile	social	web	10s_age	20s_age	30s_age	40s_age	50s_age	60s_a
difficulty	1.000000	0.160711	0.808800	0.185439	0.741807	0.531189	0.185570	0.340687	0.010037	0.002133	0.005289	0.001768	0.003343	0.001
duration	0.160711	1.000000	0.465200	0.792408	0.287798	0.078430	0.287354	0.120237	0.002433	0.001797	0.000599	0.006186	0.000531	0.001
reward	0.808800	0.465200	1.000000	0.030666	0.598324	0.742775	0.156038	0.243598	0.008070	0.003872	0.003231	0.000590	0.004404	0.001
bogo	0.185439	0.792408	0.030666	1.000000	0.666752	0.273687	0.249026	0.104960	0.000371	0.000547	0.001628	0.006785	0.001438	0.000
discount	0.741807	0.287798	0.598324	0.666752	1.000000	0.410478	0.166076	0.409560	0.004959	0.005512	0.003355	0.005670	0.001107	0.002
mobile	0.531189	0.078430	0.742775	0.273687	0.410478	1.000000	0.410246	0.168115	0.007072	0.000689	0.002341	0.002636	0.007569	0.005
social	0.185570	0.287354	0.156038	0.249026	0.166076	0.410246	1.000000	0.409792	0.003436	0.004195	0.004985	0.002534	0.008766	0.006
web	0.340687	0.120237	0.243598	0.104960	0.409560	0.168115	0.409792	1.000000	0.000428	0.002925	0.000207	0.000743	0.000874	0.000
10s_age	0.010037	0.002433	0.008070	0.000371	0.004959	0.007072	0.003436	0.000428	1.000000	0.037842	0.040114	0.050448	0.066136	0.059
20s_age	0.002133	0.001797	0.003872	0.000547	0.005512	0.000689	0.004195	0.002925	0.037842	1.000000	0.109072	0.137171	0.179827	0.161
30s_age	0.005289	0.000599	0.003231	0.001628	0.003355	0.002341	0.004985	0.000207	0.040114	0.109072	1.000000	0.145404	0.190621	0.170
40s_age	0.001768	0.006186	0.000590	0.006785	0.005670	0.002636	0.002534	0.000743	0.050448	0.137171	0.145404	1.000000	0.239729	0.214
50s_age	0.003343	0.000531	0.004404	0.001438	0.001107	0.007569	0.008766	0.000874	0.066136	0.179827	0.190621	0.239729	1.000000	0.281
60s_age	0.001141	0.001517	0.001535	0.000995	0.002829	0.005922	0.006384	0.000791	0.059282	0.161191	0.170866	0.214884	0.281707	1.000
70s_age	0.003072	0.004112	0.004405	0.001094	0.003464	0.001653	0.003508	0.002617	0.043472	0.118203	0.125298	0.157578	0.206579	0.185
80s_age	0.004529	0.001666	0.002685	0.002614	0.000722	0.000626	0.005347	0.003834	0.028805	0.078323	0.083024	0.104413	0.136882	0.122
90s_age	0.002328	0.000998	0.000849	0.002296	0.003394	0.001052	0.000619	0.001106	0.015575	0.042349	0.044891	0.056455	0.074011	0.066
100s_age	0.001009	0.007182	0.001938	0.005173	0.003810	0.003621	0.000398	0.005314	0.003858	0.010491	0.011120	0.013985	0.018334	0.016
income<=50000	0.002909	0.004514	0.004190	0.004028	0.001330	0.003156	0.004724	0.001154	0.061455	0.169842	0.135990	0.013271	0.100098	0.069
income<=70000	0.005561	0.006008	0.007180	0.003135	0.001126	0.005149	0.004022	0.001910	0.020198	0.044606	0.051445	0.060461	0.048265	0.034
income<=90000	0.006193	0.001583	0.006109	0.001965	0.004968	0.005589	0.002064	0.001736	0.045426	0.114961	0.097033	0.016702	0.072845	0.048
income<=110000	0.002701	0.002650	0.003237	0.002033	0.000192	0.002877	0.002080	0.003985	0.042928	0.116724	0.101912	0.055840	0.087722	0.053
income<=130000	0.002560	0.005801	0.000232	0.006814	0.006111	0.002505	0.002383	0.005062	0.021165	0.057549	0.061003	0.054240	0.046219	0.052

b. Implementation

I will split my data into training and testing datasets using train_test_split as 80% training and 20% testing ratios. Then, I will save the data sets to csv files. I will export the csv files into data frames to be trained by the three models.

For training my models I will be using:

- Cross validation method to get the best training data to avoid over and under fitting.
- Grid search for parameter selection and comparing accuracies of different hyper-parameters and to select the best parameter for each model, this will save me time in trying different parameters and comparing the results:

1- Logistic Regression

I will split the data into 4 folds, cross_val_score will divide them into 3 training sets and 1 testing set, so the test is 25% of the training data set, which is 20% of the total data.

I used the following configurations:

'C': 0.1,1,10,100

'max_iter' : 50,100,1000,10000

```
LR best parameter: {'C': 1, 'max_iter': 50}
LR best score: 75.20%
```

2- Decision Tree

I used same number of folds and the below configurations:

'max_depth': [4,6,8,9,10,20],
'splitter' : ['best', 'random'],
'min_samples_split' : np.arange(3, 15),
'criterion' : ['gini', 'entropy']

```
DT best parameter: {'criterion': 'entropy', 'max_depth': 8, 'min_samples_split': 3, 'splitter': 'best'}
DT best score: 77.27%
```

3- Random Forest

I used same number of folds and the below configurations:

'max_depth': [4,6,8,10,12],
'n_estimators' : [20,50,100,200],
'min_samples_split' : np.arange(3, 15),
'criterion' : ['gini', 'entropy']

```
RF best parameter: {'criterion': 'entropy', 'max_depth': 10, 'min_samples_split': 9, 'n_estimators': 100}
RF best score: 77.38%
```

The training accuracy for the three models are close and above 75% which is acceptable as it exceeds the threshold I've set earlier, however the Random forest has the highest score 77.38%. Now let's test our three models and compare the metrics results through the accuracy score and confusion matrix.

IV. Results

a. Model Evaluation and Validation

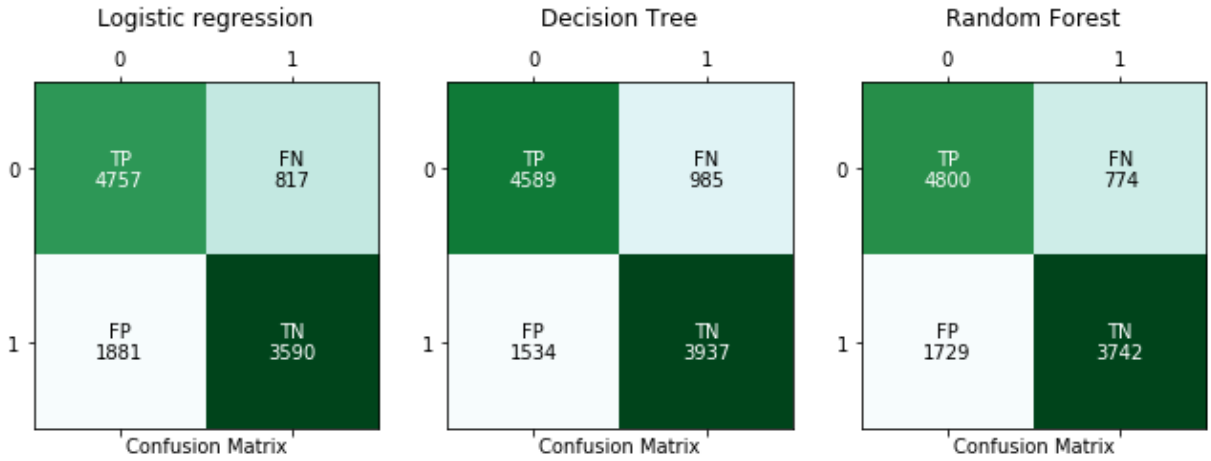
After loading the models, it's time to test each model and see check testing scores, we will notice that the accuracy score are almost the same, with a very small degradation and the Random Forest still has the highest score of 77.34%

The confusion matrix shows that the random forest has the highest accuracy TP count 4800 and the lowest FN count 774.

The below results shows that our Random Forest model is able to predict the customer response with a high accuracy and recall and can make be mistaken with the FN percentage = 16%, which is an acceptable range.

The Random Forest model is really powerful when it comes to avoid overfitting, errors and bias and that's why it was a bit expected for me that it would perform better than the other models.

	Accuracy %	Precision	Recall
Logistic Regression	75.57%	0.7166	0.8534
Decision Tree	77.19%	0.7495	0.8233
Random Forest	77.34%	0.7352	0.8611



b. Model Refinement

Since the highest accuracy belongs to the Random forest model, we will take this model and apply some refinement on by changing the hyper parameters and check if we can get a higher accuracy or not. I will test the model using Grid search:

```
RF best parameter: {'criterion': 'entropy', 'max_depth': 14, 'min_samples_split': 14, 'n_estimators': 250}
RF best score: 77.10%
```

And another trial:

```
RF best parameter: {'criterion': 'entropy', 'max_depth': 9, 'min_samples_split': 4, 'n_estimators': 20}
RF best score: 77.15%
```

It seems we will not get any better results than the very first trial, which was acceptable already, so no more refinement needed.

c. Conclusion

In this project I addressed the problem of not all customers respond to the offers sent by merchants the same way. Some customers don't view the offers because they are not interested, others may be not interested in that particular offer they received.

My model will predict and identify whether the customer will respond and complete these offers or not.

First, I downloaded the datasets and explored each of them and started to eliminate some of the non-related features and to normalize other features that has non-numeric values in order to be able to feed the models with more meaningful data.

After that I merged the three datasets into one grouped by customers that simply illustrate the relation between the customer demographic data, his/her previous transactions and Reponses to offers and the offers descriptive data.

I chose to train 3 models for this binary classification problem: Logistic Regression, Decision tree and Random Forest.

I chose to use the Cross Validation for training and splitting my data to choose the best folds to train, which helped in improving the training/learning accuracy of my models.

Then, I decided to work with Grid search to find the best model hyper parameters which also helped in improving the accuracy of my models, it saved me a lot of time and gave me a clear view of the best hyper parameters to use.

The Random Forest model performed the best with a testing accuracy of 77.34% and a very small FN percentage of 16%.

As an improvement to the model, I can only think of creating more features that may help in training the model, for example: we can create features that describe the

frequency of the customer visits to Starbucks in a month and another feature to describe the payment method (cash/CC).

As a final suggestion, we need to analyze the behavior of the non-responsive customers and may be target them with a more customized offers which can be also done using machine learning.