Udacity
Machine Learning Engineer Nanodegree
Starbucks Capstone Project Report

Giorgos Moysiadis

25th September 2021

I. Definition

Project Overview

Today, understanding user behavior and taking action based on data is the key to customer success and profitability. Starbucks, an international coffee chain, has successfully developed a mobile application platform that once every few days sends out an offer for its users through it.

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.

Problem Statement

The goal of the project is to create a Machine Learning model that will best determine whether or not a user will accept and complete a sent out personalized offer. Due to the fact that users do not receive the same offer, it will provide a heavy challenge during the data exploration phase. This will improve the acceptation rate of the offers and in return increase the overall profit for Starbucks

II. Analysis

Data exploration

Our Data consists of three data sets (three json files) that are provided by Udacity:

portfolio.json - containing offer ids and meta data about each offer (duration, type, etc.) (10 offers x 6 attributes)

- id (string) - offer id
- offer_type (string) - type of offer (i.e. BOGO, discount, informational)
- difficulty (int) – minimum required spend to complete an offer
- reward (int) – reward given for completing an offer

- duration (int) – time for offer to be open, in days
- channels (list of strings, ie web, email, mobile, social)

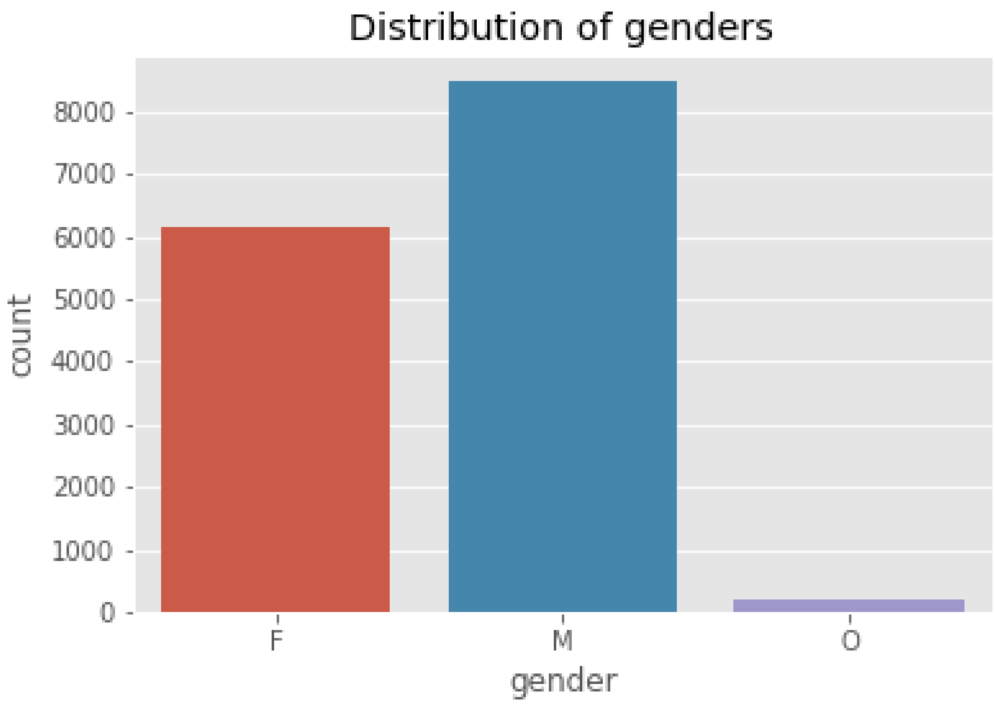
	reward	channels	difficulty	duration	offer_type	id
0	10	[email, mobile, social]	10	7	bogo	ae264e3637204a6fb9bb56bc8210ddfd
1	10	[web, email, mobile, social]	10	5	bogo	4d5c57ea9a6940dd891ad53e9dbe8da0
2	0	[web, email, mobile]	0	4	informational	3f207df678b143eea3cee63160fa8bed
3	5	[web, email, mobile]	5	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

profile.json - demographic data for each customer (17000 users x 5 attributes)

- age (int) – missing values are encoded as 118
- gender(string) – Male, Female, Other or null
- id (string) – a unique string/hash for each user
- became_member_on (datetime) – date that the user registered
- income (int) – yearly 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
...	...	...	...	...	...
16995	F	45	6d5f3a774f3d4714ab0c092238f3a1d7	20180604	54000.0
16996	M	61	2cb4f97358b841b9a9773a7aa05a9d77	20180713	72000.0
16997	M	49	01d26f638c274aa0b965d24cefe3183f	20170126	73000.0
16998	F	83	9dc1421481194dcd9400aec7c9ae6366	20160307	50000.0
16999	F	62	e4052622e5ba45a8b96b59aba68cf068	20170722	82000.0

Below is shown the distribution of genders among customers.



transcript.json - records for transactions, offers received, offers viewed, and offers completed (306534 events x 4 attributes)

- person (string) – refers to the person that that has created an event through receiving, viewing or completing an offer
- event (string) – contains the events that a user has done through a sent out offer that he received
- value (dictionary) – different values depending on event type
  - offer id: (string) – the id of the offer that was sent out
  - amount: (int) money spent in "transaction"
  - reward: (int) money or points gained from "offer completed"
- time: (int) – hours after start of test

	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
...	...	...	...	...
306529	b3a1272bc9904337b331bf348c3e8c17	transaction	{'amount': 1.5899999999999999}	714
306530	68213b08d99a4ae1b0dcb72aebd9aa35	transaction	{'amount': 9.53}	714
306531	a00058cf10334a308c68e7631c529907	transaction	{'amount': 3.61}	714
306532	76ddbd6576844afe811f1a3c0fbb5bec	transaction	{'amount': 3.5300000000000002}	714
306533	c02b10e8752c4d8e9b73f918558531f7	transaction	{'amount': 4.05}	714

Offer Types

There are three types of offers that are sent out to the user: buy-one-get-one (BOGO), discount, and informational.

In a BOGO offer, the user is required to spend a certain amount to get a reward equal to that threshold amount.

In a discount offer, the user is rewarded with a certain amount of cut off based on the money that was spend in his transaction.

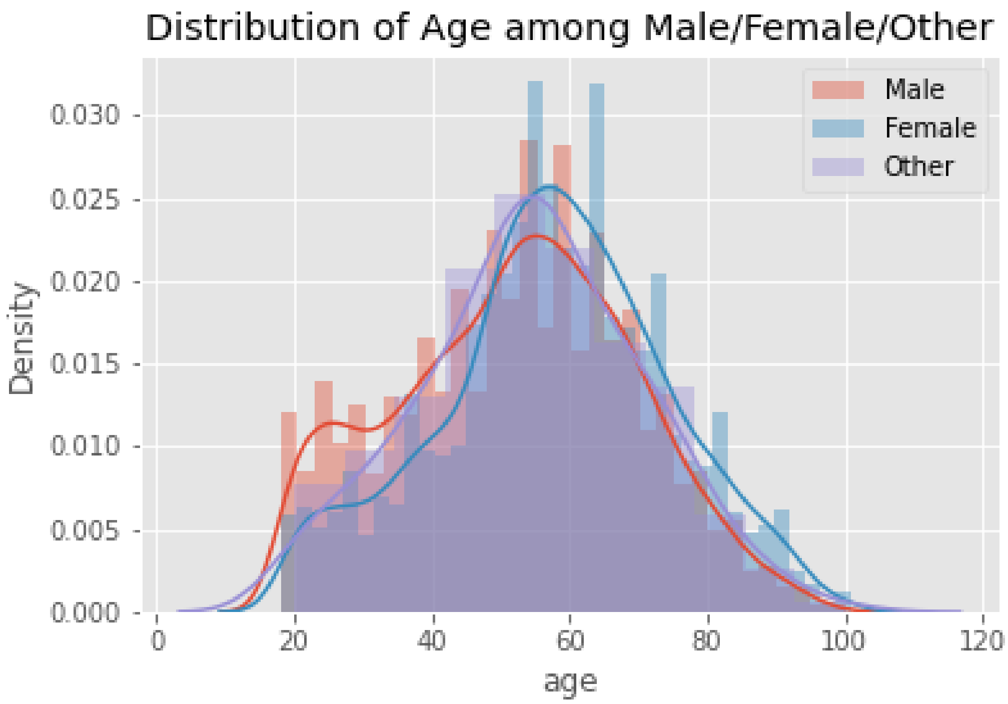
In an information offer, the user is sent out, a notification or an email, depending on the channels, regarding some information that may concern him for the future.

III. Methodology

Data Preprocessing

The first step was to clean up each data set by removing any missing values or outliers encoded for missing values, such as 118 in age column in the profile data set.

After having removed the missing values the following graph displays the age distribution among the gender attribute.

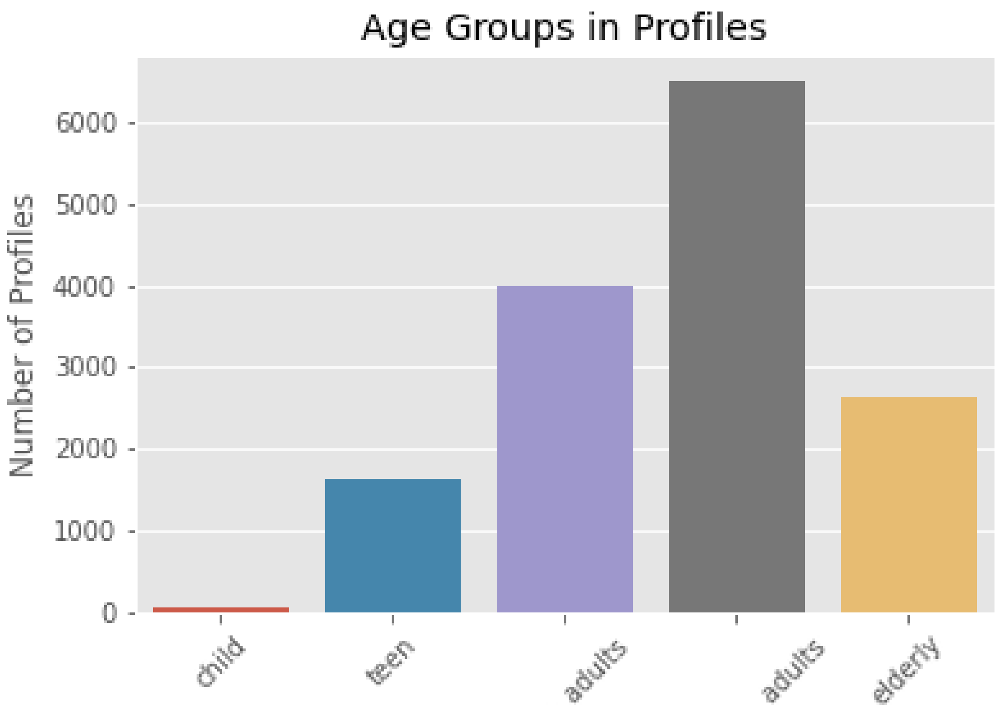


Next we are going to apply one hot encoding on the gender attribute. In general, one hot encoding allows the representation of categorical data to be more expressive and there are some machine learning algorithms that cannot work with categorical data directly.

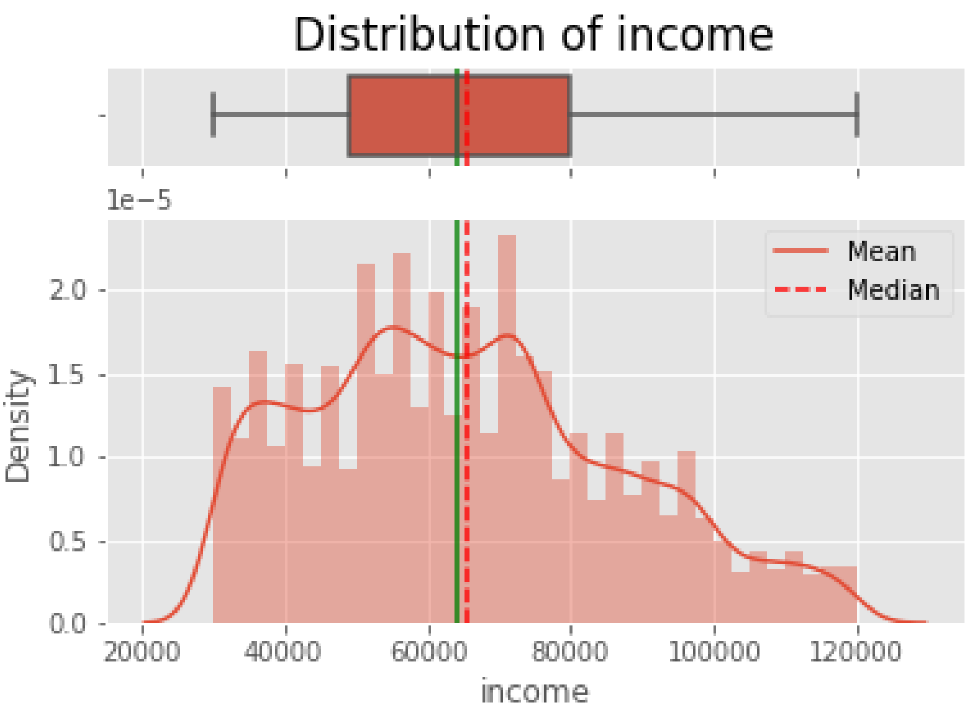
With that into consideration, one hot encoding will be applied to following attributes that are going to be converted from numerical values into categorical.

Due to the fact that “age” is a continuous numerical value it seemed a better option to cluster the ages into age groups, meaning that from a certain young age to an older one would consist of a certain age group.

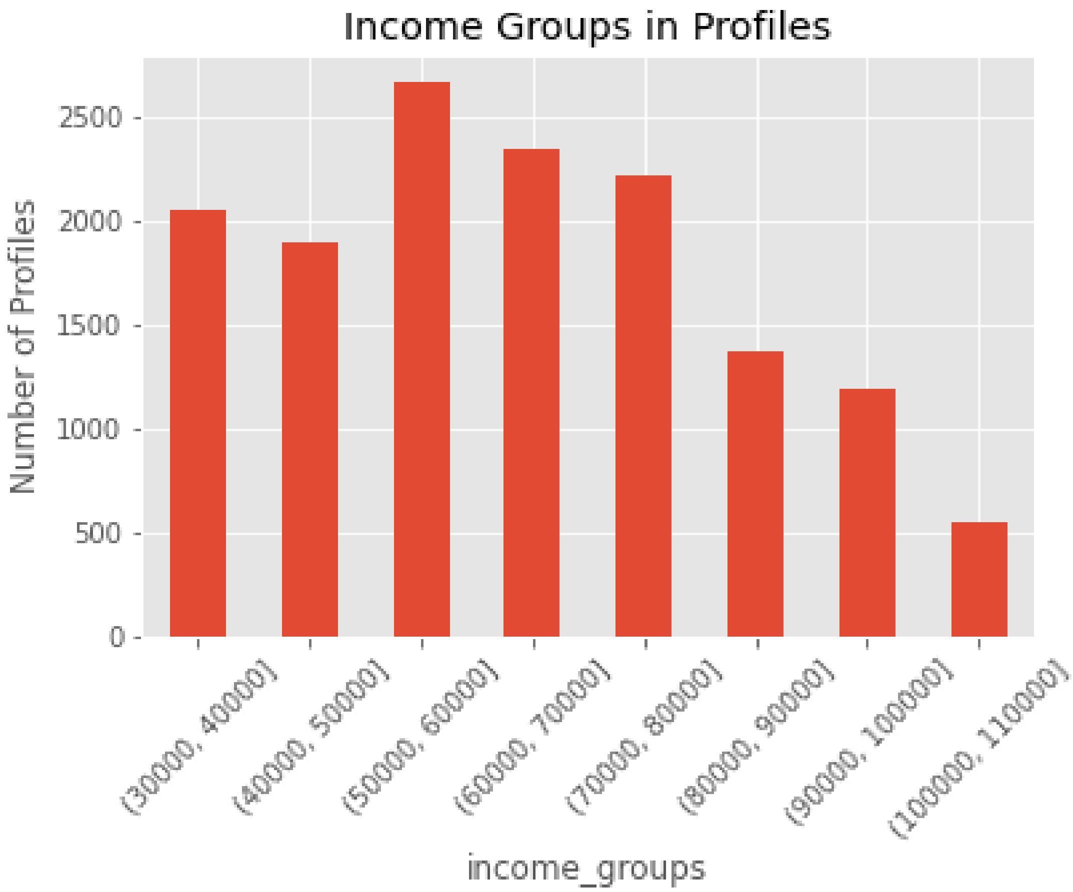
With that into consideration “age” was broken down into 5 categorical age groups of 'child', 'teen', 'young_adults', 'middle_age_adults', 'elderly', and they were distributed as it can be seen below.



Following comes the income distribution, and it seems that it follows a somewhat normal distribution.

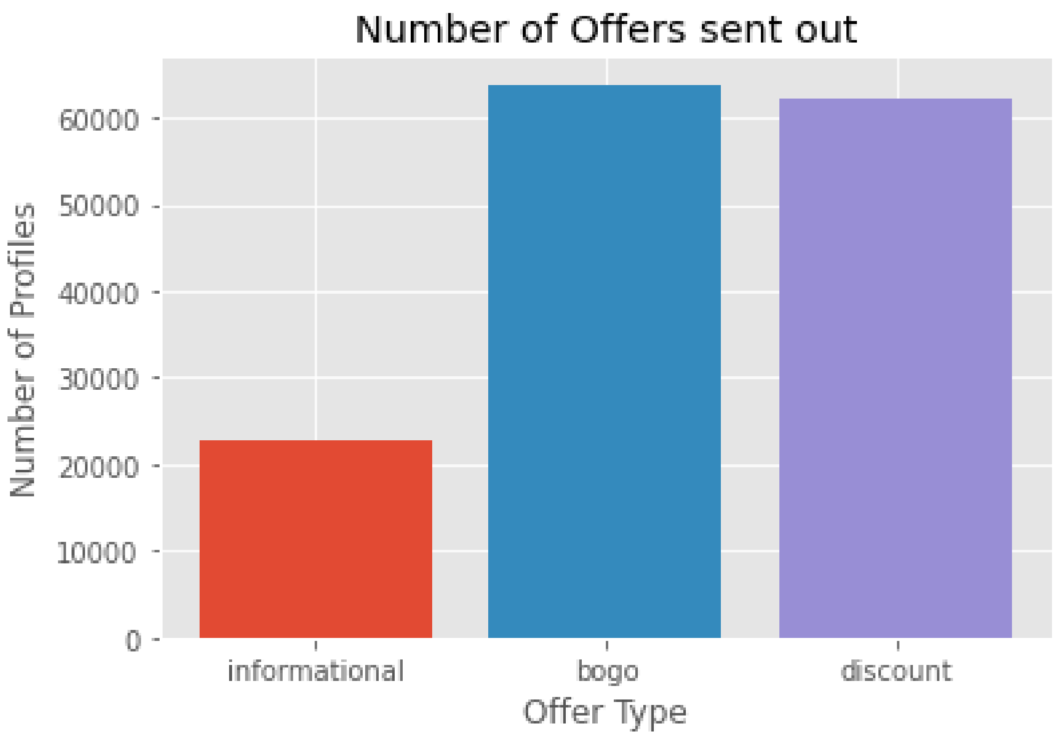


But just like age, “income” is also continuous numerical attribute and hence we can also partition it into separate groups. Below are the resultant groups of income.



Number of sent out offers to customers.

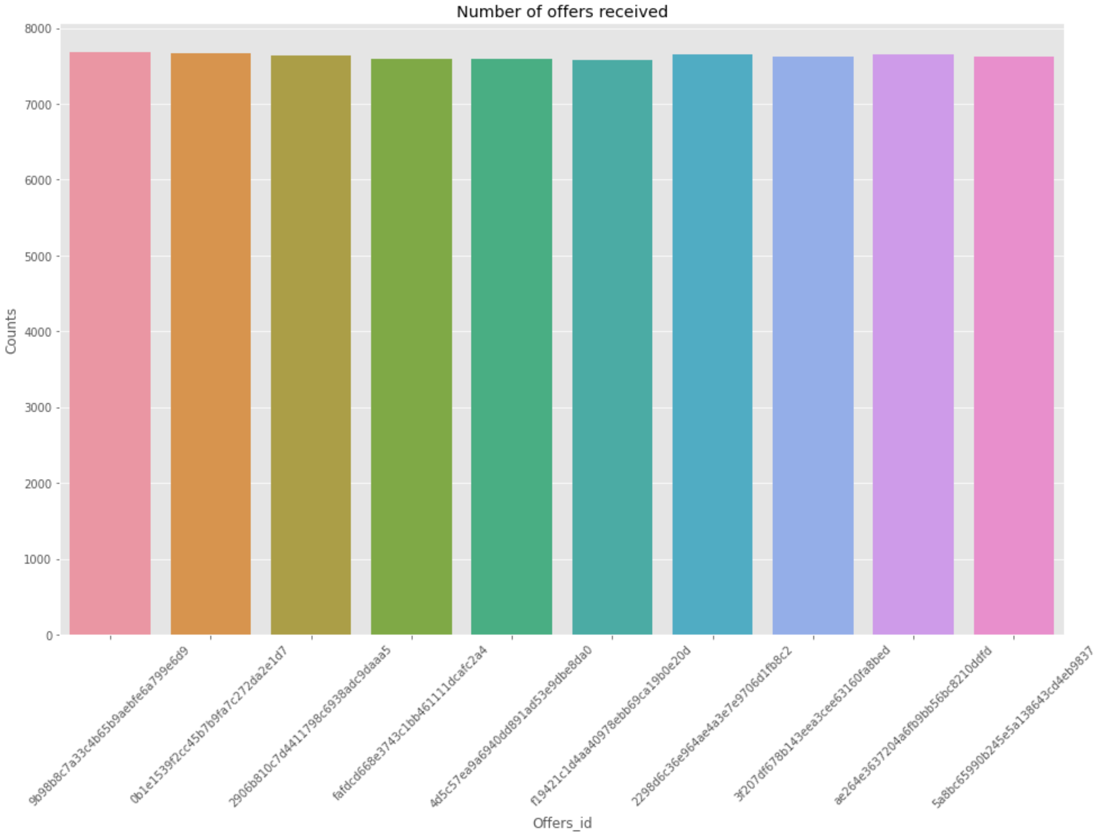
As mentioned above, one hot encoding is also applied here, with the categories of the offers in the portfolio data set.



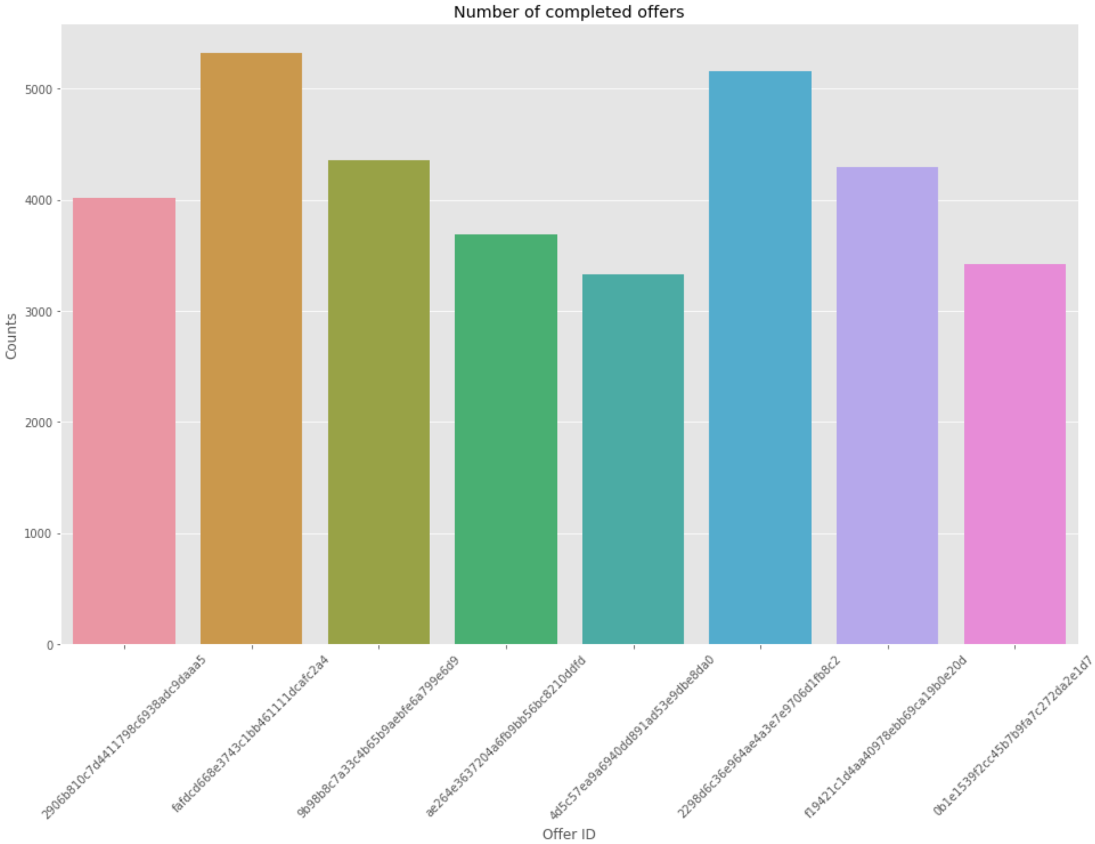
Events that users do when they receive an offer

It can be seen that from the total offers that are received by the customers, only half of them are completed.

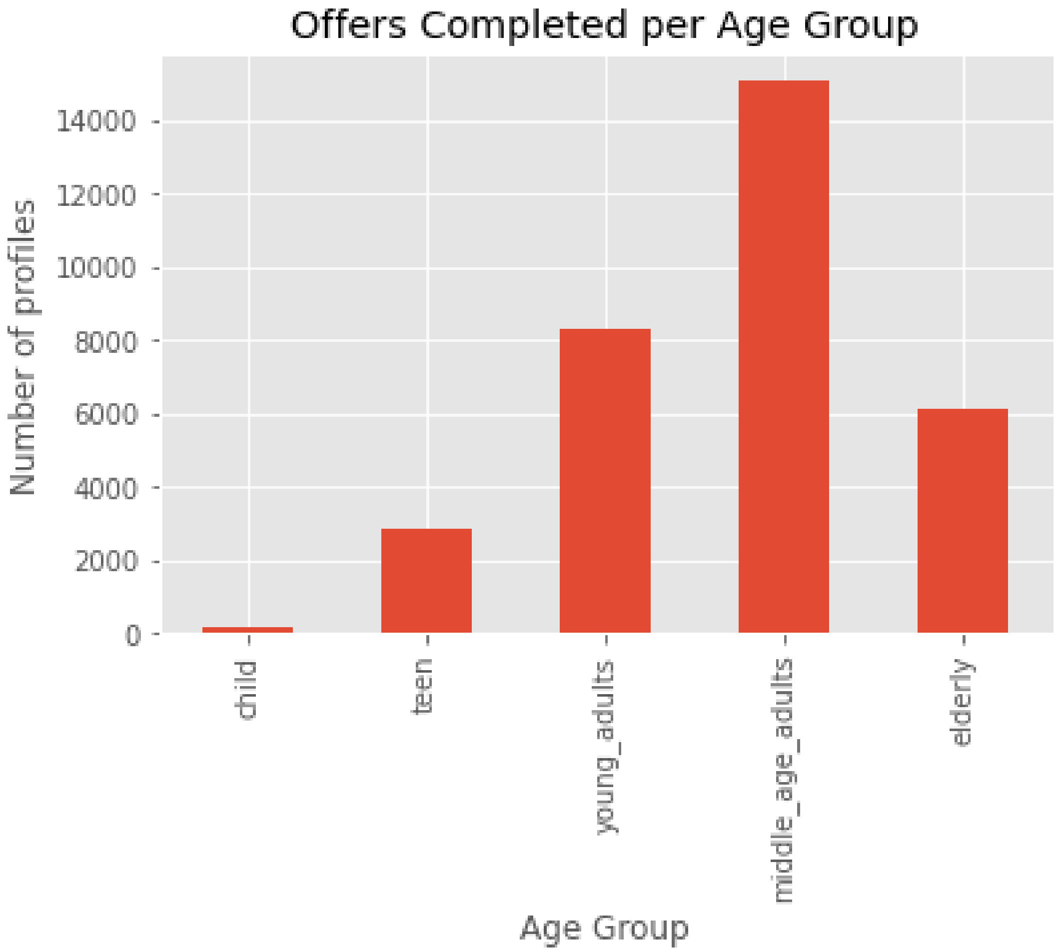
Number of sent out offers based on offer id.



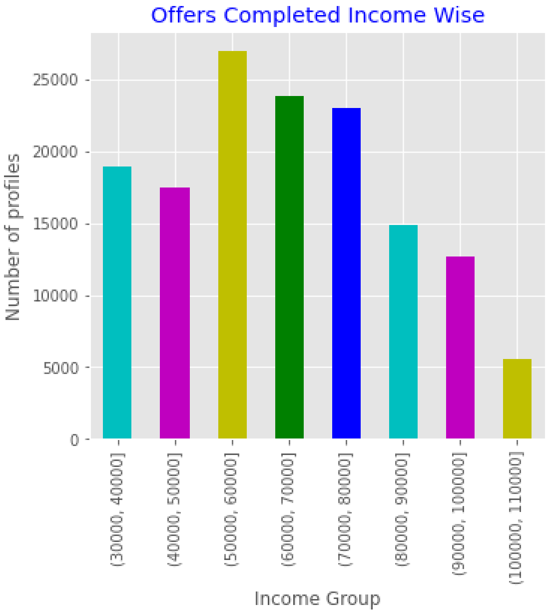
Number of completed offers based on offer id.



Number of Completed Offers per Age Group



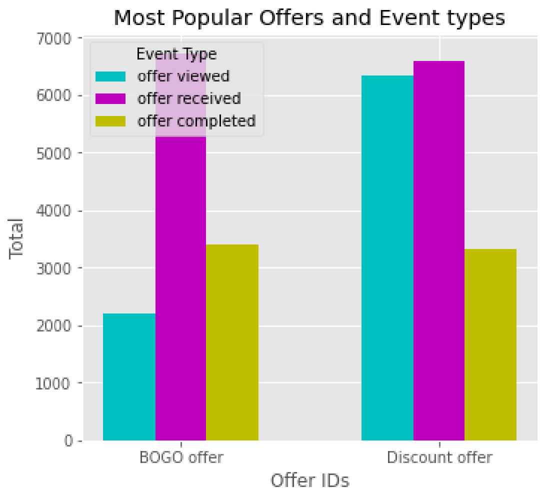
Gender and income distribution of completed offers

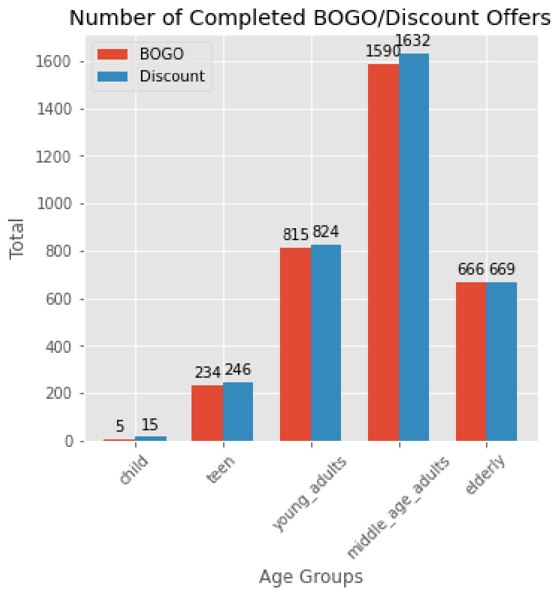


From the graphs shown above, we can calculate how much of the total customer transactions are a part of. Roughly around the 62% of the customers around the middle age groups complete offers.

BOGO and Discount Offers

The following diagram showcases the popularity of two of the most difficult but also the most rewarding BOGO and discounts offers that are sent out to customers.





It can easily be seen that, due to its nature, the discount offer is receives the most views out of the 2 offers. Although in regards with the number of completed offers among the age groups there is a slight edge for the discount offer.

Merge Data

The next step is to combine those data sets and distinguish what attributes are going to be input features and label. Finally, we are going to normalize some of the numerical attributes that are left out, such as time, difficulty, duration and reward. We have the total columns that are shown below.

```
data.columns
```

```
Index([      'customer_id',               'time',            'offer_id',
         'offer completed',     'offer received',        'offer viewed',
                  'reward',         'difficulty',            'duration',
                     'web',              'email',              'mobile',
                  'social',               'bogo',            'discount',
           'informational',             'gender',                 'age',
        'became_member_on',                  'F',                   'M',
                       'O',     (30000, 40000],      (40000, 50000],
          (50000, 60000],     (60000, 70000],      (70000, 80000],
          (80000, 90000],    (90000, 100000],    (100000, 110000],
        (110000, 120000],              'child',                'teen',
          'young_adults', 'middle_age_adults',             'elderly'],
      dtype='object')
```

Train – Test

We are going to assign "offer completed" as the target attribute, since we want a model to predict whether or not a customer will complete as sent out offer. We are going to drop 'customer_id', 'offer received', 'offer viewed', 'informational', 'offer_id, 'offer completed', 'became_member_on' and 'gender'. And we are left with the following features.

```
scaled_features.columns
```

```
Index([             'time',             'reward',          'difficulty',
                'duration',                'web',               'email',
                  'mobile',             'social',                'bogo',
                'discount',                'age',                   'F',
                       'M',                  'O',      (30000, 40000],
          (40000, 50000],      (50000, 60000],      (60000, 70000],
          (70000, 80000],      (80000, 90000],     (90000, 100000],
        (100000, 110000],    (110000, 120000],               'child',
                    'teen',       'young_adults', 'middle_age_adults',
                 'elderly'],
      dtype='object')
```

Splitting our data with 70% used for training and the rest 30% for testing we have the following distribution of the samples.

```
Training samples: 104163
Testing samples: 44642

Target distribution
Train set
0    0.781967
1    0.218033
Name: offer completed, dtype: float64
Test set
0    0.781977
1    0.218023
Name: offer completed, dtype: float64
```

IV. Results

Model Evaluation and Validation

I will be using Logistic regression model as a benchmark model in which to compare our models' performance to because it is fast and simple to implement.

- Logistic Regression: model using logistic function to predict the result
- Decision Tree Classifier: model using series decision nodes to predict the result
- Random Forest Classifier: model using multiple decision tree classifier to predict the result

We will implement the:

- ROC (Receiver Operating Characteristic) and AUC (Accuracy Under Curve) score,
- Precision: How many predicted values are relevant?
- Recall: How many relevant items are selected, and
- F1 score: A weighted average of the precision and recall

to compare the performance of the models.

Logistic Regression

```
LogisticRegression Precision score: 0.58
[[34893    16]
 [ 9711    22]]
```

As it can be seen the model assumes that almost everyone will complete an offer that they receive, while for people who **are** going to complete the offer the model works perfectly, for people that are **not** going to complete an offer we have almost all of our samples falsely categorized.

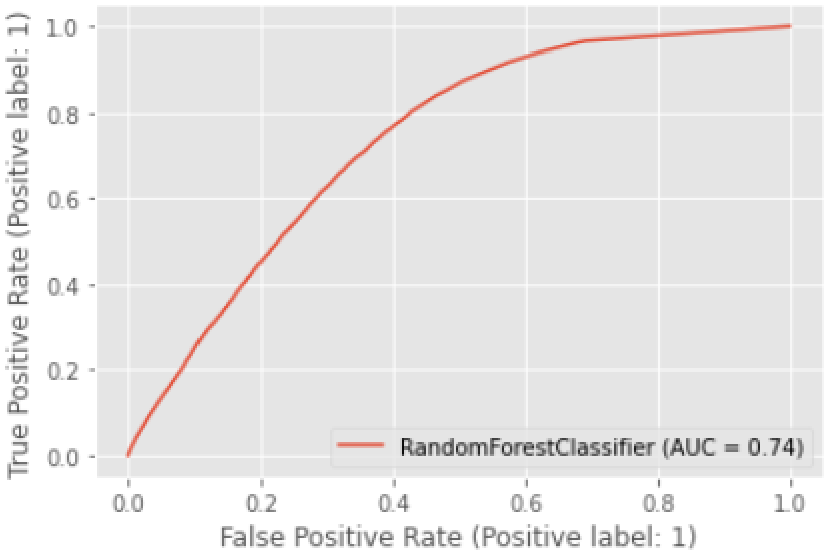
Decision Tree

```
DecisionTreeClassifier Precision score: 0.5
[[30490  4419]
 [ 5342  4391]]
```

Now the DecisionTree classifier, while for the most part it has categorized people that **are** going to complete offers correctly, for people that are **not** going to complete any offers it shows mixed results.

Random Forest

```
RandomForestClassifier Precision score: 0.41
[[30966  3943]
 [ 6994  2739]]
```



Hyperparameter optimization

Our next step will be to try to apply GridSearchCV, which is a hyperparameter optimization technique.

Hyperparameter optimization is the search of certain variables, that a machine learning model utilizes during its training, that determine and optimize the performance, they are key characteristics, such as choosing a different learning method, the max depth of a decision tree, or the number of hidden layers in a neural network etc.

```
hyper_params = [
    {'C':[0.001, 0.005, 0.01, 0.05, 0.1, 0.5, 1]},
    {'criterion': ['gini', 'entropy'],
     'max_depth': range(1,10),
     'min_samples_split': range(1,10),
     'min_samples_leaf': range(1,5)},
    {'max_depth':[2,5,10], 'n_estimators':[100,300]}
]
```

After training on these parameters the GridSearchCV will provide us with the appropriate hyperparameters to supply our classifiers in order to achieve the best performance.

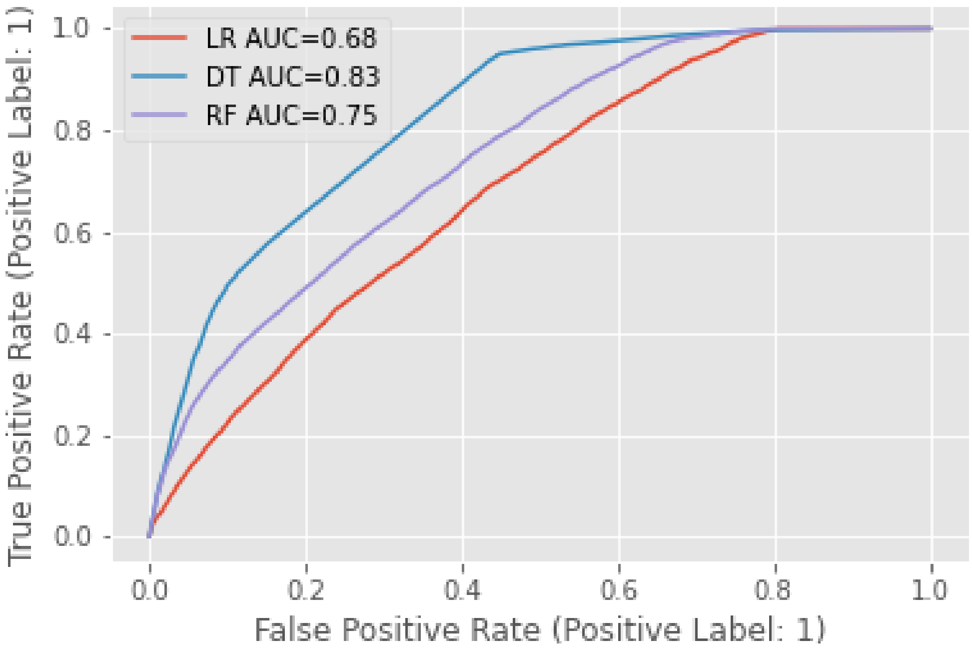
```
LogisticRegression Precision score: 0.0
[[34909     0]
 [ 9733     0]]

DecisionTreeClassifier Precision score: 0.61
[[32361  2548]
 [ 5679  4054]]

RandomForestClassifier Precision score: 0.65
[[34213   696]
 [ 8451  1282]]
```

As it can be seen the LogisticRegression model classified everyone as if they would all complete a sent out offer.

Although, there is an increase in precision for the rest of the models which is a very good indicator, as the precision metric asks the question "How many predicted values are relevant?" and 60% is certainly above average but definitely should be taken for reconsideration.



	Logistic Regression	Decision Tree	Random Forest
accuracy	0.78	0.82	0.8
precision	0.0	0.61	0.65
recall	0.0	0.42	0.13
auc	0.68	0.83	0.75
f1	0.0	0.5	0.22
mse	0.22	0.18	0.2
confusion_matrix	[[34909, 0], [9733, 0]]	[[32361, 2548], [5679, 4054]]	[[34213, 696], [8451, 1282]]

After applying GridSearch we can definitely see that there is a better performance, on almost all metrics, from the Decision Tree classifier as it has the lowest false positives out of the 3 total classifiers, and it is something that should really be taken into consideration regarding sending personalized offers in hopes that the selected customer will complete the offer.

V. Conclusions

So now we have to decide which one is the best model, and we have two types of wrong values:

- False Positive, means the client in reality did **NOT COMPLETE** an offer, but the **MODEL** thinks/predicts they did, which is called a **Type I Error**.
- False Negative, means the client in reality **COMPLETED** an offer, but the **MODEL** thinks/predicts they didn't, which is called a **Type II Error**

In my opinion:

- Type II error is the most harmful, because the model thinks that a customer **WILL NOT** complete an offer, but of course in reality they did, and the reason behind this is that our model is not able to find the "loyal" customers, the ones that complete offers.
- Type I error is not good, but it's ok, because in the end we try to find new customers that **maybe** complete offers that are sent out to them, and of course we are going to find some bumps in the road, as the saying tells, in order to find those.

Now based on the resultant confusion matrices one could say that using a simple LogisticRegression will suffice, because it thinks that everyone will complete an offer and the business needs more and more users to buy their products through their sent out offers or not.

Theoretically, there is no harm that **BUT** the constant notifications and reminders of new offers may irritate some customers and may result in using less and less the brand and the product.

In conclusion, from the runs above it can be easily seen that the clear choice is the Decision Tree classifier. Now, based on the available computing power, it can be upgraded a more complicated model, such as a deep learning model, through the PyTorch or TensorFlow frameworks.

VI. References

- https://en.wikipedia.org/wiki/Receiver_operating_characteristic
- https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html
- https://en.wikipedia.org/wiki/F-score
- https://matplotlib.org/stable/gallery/lines_bars_and_markers/barchart.html