UDACITY

Machine Learning Engineer Nanodegree Program

Capstone Project

Starbucks Capstone Challenge

AHMED SAFWAT EWIDA

DEC ,2019

Table of Contents

A DEFINITION

01

Project overview

Domain Background

Machine learning (ML) has become an increasingly important part of IT today. This effect is seen both in how IT leverages machine learning to improve operations and in how IT supports and enables the lines of business (LOBs). Still, organizations have limited understanding on its effective use and have made limited progress in associating it with business outcomes.

Admittedly , The Companies which will lead in the future are those who will be interested in implementing the machine learning algorithms on the enormous amount of data base which they have , they will be the pioneers in their field.

STARBUCKS is one of flagship Worldwide companies which has been established since 31st March 1971 and have worldwide coffeehouse chain, and has a tremendous database of users , that is why I am interested in implementing my capstone project for STARBUCKS Capstone Challenge as I believe that I can implement a good Machine Learning Model for one of the most Worldwide prestigious companies.

Customers' Concerns are the goal for all companies all over the world , what people like? , how much they want to pay?, when do they are capable to pay? , what is the gender and age of those people who are interested and capable to pay? are very important questions, and the answer comes from Historical data which we have to implement a deep learning algorithms to it , and building machine Learning Algorithms according to those Historical data to maximize Companies s' profits.

02

Problem Statement

Problem Statement

The Problem Statement as mentioned in **Starbucks Capstone Challenge ,analysing the data set for STARBUCKS Customers and building a Model t**hat predicts whether or not someone will respond and complete to an offer.
We have an enormous number of users , some of them are making transaction either the received or not received an offer ,others are just viewing the offers without completing it , others responding to specific type of offers and completing it.
We have to make analysis for those who are receiving , viewing and make transaction within the offer period and those customers are our target.
Analysing the demographic feature for the above mentioned customers , their gender , age , income , the membership period and the type of offers which they are interested are the most important step before building our Model to stand on the Features which we will use in our Model.
The customers who are not influenced by the offers , or they purchase either they have received an offer or not are NOT in out target.
Cleaning , analysing and Visualizing the data consumes 90 % of the efforts to build a good model.

Problem Statement

1-The Below flow chart for the users Whom received , viewed ,completed the offer and make transaction within the offer period and those customers are our target .

2-we will track the amount of money which has been spent by customers within the offer period and till the offer completed , to track the profits that can be gained by each customer for each offer.

3-we will do our statistics analysis and data visualization to understand the role of the features which controlling our model ,such as : Customers ‘s gender , Customers ‘s age ,customers ‘s membership , Customers ‘s income , offer duration …..etc.

4-We will do assumptions that all transactions executed within the offer period -for the customers whom completing the offers- will be through utilizing offers .

Solution Statement

We will Follow the below process in our Problem Solution:

Solution Statement

- **Fetching the Data:**
  The Data sets mentioned in the previous slide to be converted to CSV Files , and to be ready for next step.
- **Clean /preparation Data:**
  1.Wrangle data and prepare it for training
  2.remove duplicates, correct errors, deal with missing values, normalization, data type conversions, …etc.)
- **Data Visualizing and analysis:**
  1.Visualize data to help detect relevant relationships between variables.
  2.Split into training and evaluation sets
- **Taring Model:**
  The goal of training is to make a prediction correctly as often as possible.
- **Evaluating the Model:**
  1.Uses some metric or combination of metrics to measure the performance of model.
  2.shuffling the data and selecting 20/80 ratio for test/train data set.
  3.Hyper-parameter tuning, which is a corner stone for Model efficiency and performance improvement.
  4.Using test set data which have to predict the output.

03

Evaluation Metrics

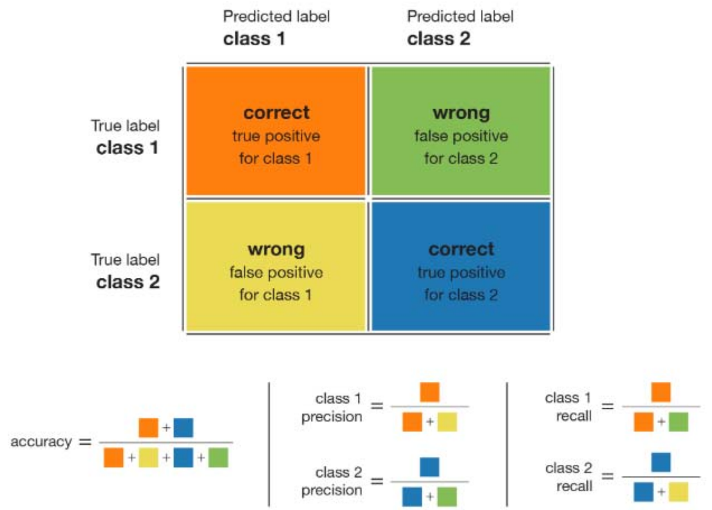
Evaluation Metrics

Our Problem is Classification Problem with imbalanced nature , that will lead us to use the following Metrics:

roc_auc_score : Compute Area Under the Receiver Operating Characteristic Curve (ROC AUC) from prediction scores.

Precision : The proportion of positive cases that were correctly identified.

Recall : The proportion of actual positive cases which are correctly identified.



The confusion matrix and the metrics that can be derived from it.

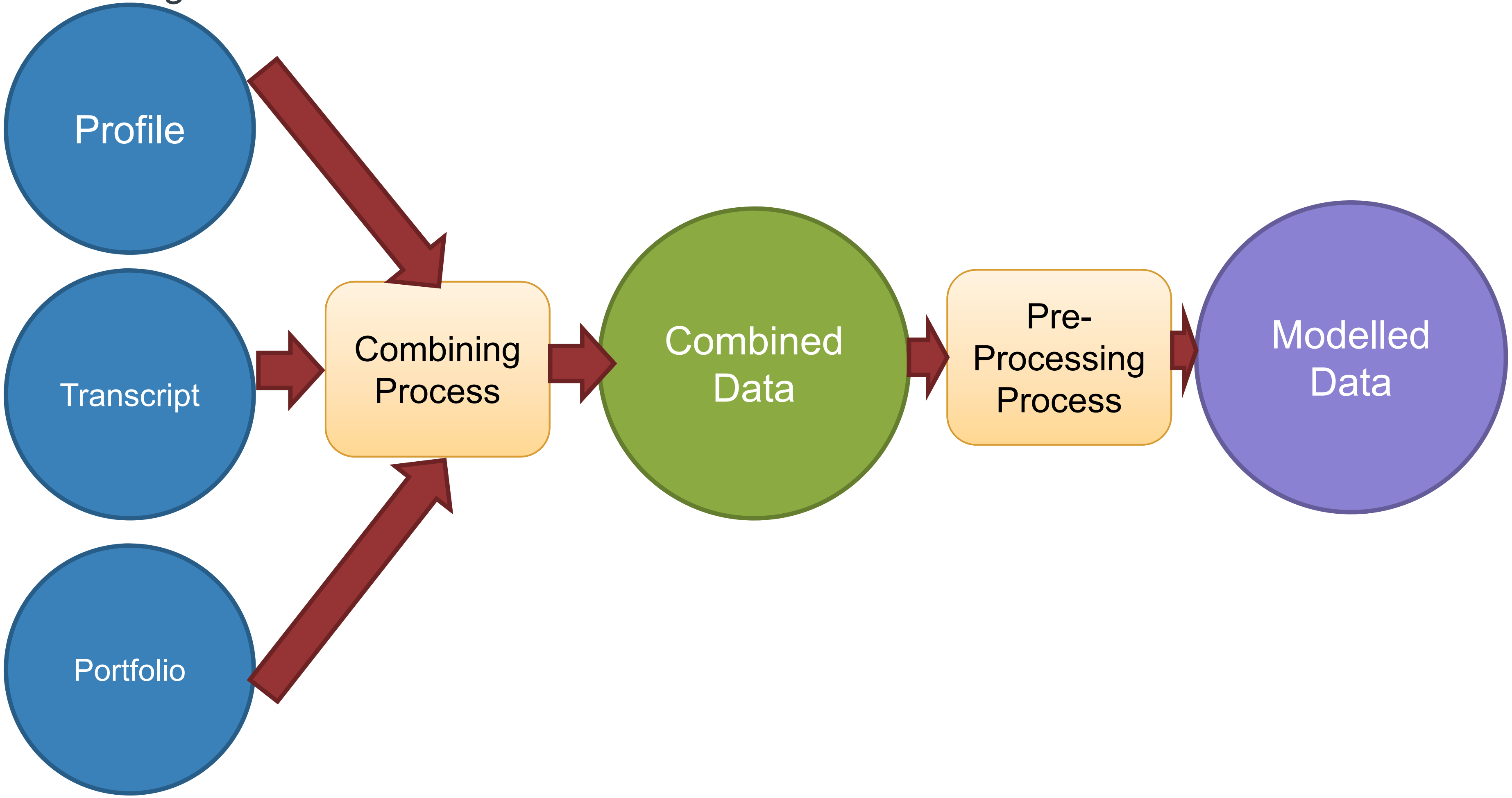
B ANALYSIS

01

Data Exploration

Datasets and Inputs

Our Data consists of three data sets (three json files) , we will follow the hereunder process till reaching to our Modelled data which we will be used in our Model training and testing.



Datasets and Inputs

We have three JSON Files :

- portfolio.json - containing offer ids and meta data about each offer (duration, type, etc.)
- profile.json - demographic data for each customer
- transcript.json - records for transactions, offers received, offers viewed, and offers completed

portfolio.json: shape (10 rows x 6 columns)

- id (string) - offer id
- offer_type (string) - type of offer ie 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)

Datasets and Inputs

profile.json:shape (2175 rows x 5 columns) with 17000 unique users.

- 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

transcript.json: (306534 rows x 4 columns)

- 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 or transaction amount depending on the record

Cleaning Data sets and reframing

UDACITY

Profile Data Set:

1-Dividing the age Column to five age groups:

- **-Child : less than 18 years old.**
- **-Teen :between 30 and 18 years old.**
- **-Young adults : between 50 and 30 years old.**
- **-Middle age adults : between 70 and 50 years old.**
- **-Elderly : between 70 and 50 years old.**

2-Transform the became_member_on Column to Month / year Format.

3-Claculating the Customer subscription cumulative number of days since the customer has been started his subscription.

4-Dropping the NA Values.

	age	became_member_on	gender	customer	income	age_groups	member_launch_Cum_days	member_launch_year
1	55	2017-07-15	F	0610b486422d4921ae7d2bf64640c50b	112000.0	middle_age_adults	16908	2017
3	75	2017-05-09	F	78afa995795e4d85b5d9ceeca43f5fef	100000.0	elderly	16841	2017
5	68	2018-04-26	M	e2127556f4f64592b11af22de27a7932	70000.0	middle_age_adults	17193	2018
8	65	2018-02-09	M	389bc3fa690240e798340f5a15918d5c	53000.0	middle_age_adults	17117	2018
12	58	2017-11-11	M	2eeac8d8feae4a8cad5a6af0499a211d	51000.0	middle_age_adults	17027	2017
13	61	2017-09-11	F	aa4862eba776480b8bb9c68455b8c2e1	57000.0	middle_age_adults	16966	2017
14	26	2014-02-13	M	e12aeaf2d47d42479ea1c4ac3d8286c6	46000.0	teen	15660	2014
15	62	2016-02-11	F	31dda685af34476cad5bc968bdb01c53	71000.0	middle_age_adults	16388	2016
16	49	2014-11-13	M	62cf5e10845442329191fc246e7bcea3	52000.0	young_adults	15933	2014
18	57	2017-12-31	M	6445de3b47274c759400cd68131d91b4	42000.0	middle_age_adults	17077	2017

Cleaning Data sets and reframing

UDACITY

Transcript Data Set:

1-Dividing the value Column to offer id and amount columns.

2-changing the name of person Column to Customer Column.

	event	customer	time	offer_id	amount
0	offer received	78afa995795e4d85b5d9ceeca43f5fef	0	9b98b8c7a33c4b65b9aebfe6a799e6d9	0
1	offer received	a03223e636434f42ac4c3df47e8bac43	0	0b1e1539f2cc45b7b9fa7c272da2e1d7	0
2	offer received	e2127556f4f64592b11af22de27a7932	0	2906b810c7d4411798c6938adc9daaa5	0
3	offer received	8ec6ce2a7e7949b1bf142def7d0e0586	0	fafdcd668e3743c1bb461111dcafc2a4	0
4	offer received	68617ca6246f4fbc85e91a2a49552598	0	4d5c57ea9a6940dd891ad53e9dbe8da0	0
5	offer received	389bc3fa690240e798340f5a15918d5c	0	f19421c1d4aa40978ebb69ca19b0e20d	0
6	offer received	c4863c7985cf408faee930f111475da3	0	2298d6c36e964ae4a3e7e9706d1fb8c2	0
7	offer received	2eeac8d8feae4a8cad5a6af0499a211d	0	3f207df678b143eea3cee63160fa8bed	0
8	offer received	aa4862eba776480b8bb9c68455b8c2e1	0	0b1e1539f2cc45b7b9fa7c272da2e1d7	0
9	offer received	31dda685af34476cad5bc968bdb01c53	0	0b1e1539f2cc45b7b9fa7c272da2e1d7	0
10	offer received	744d603ef08c4f33af5a61c8c7628d1c	0	0b1e1539f2cc45b7b9fa7c272da2e1d7	0
11	offer received	3d02345581554e81b7b289ab5e288078	0	0b1e1539f2cc45b7b9fa7c272da2e1d7	0
12	offer received	4b0da7e80e5945209a1fdddfe813dbe0	0	ae264e3637204a6fb9bb56bc8210ddfd	0
13	offer received	c27e0d6ab72c455a8bb66d980963de60	0	3f207df678b143eea3cee63160fa8bed	0
14	offer received	d53717f5400c4e84affdaeda9dd926b3	0	0b1e1539f2cc45b7b9fa7c272da2e1d7	0

Cleaning Data sets and reframing

Portfolio Data Set:

1-Dividing the Channels Column to Web, email , Mobile and Social media Columns .

2-changing the name of id Column to offer id Column to mange the merging between the data set .

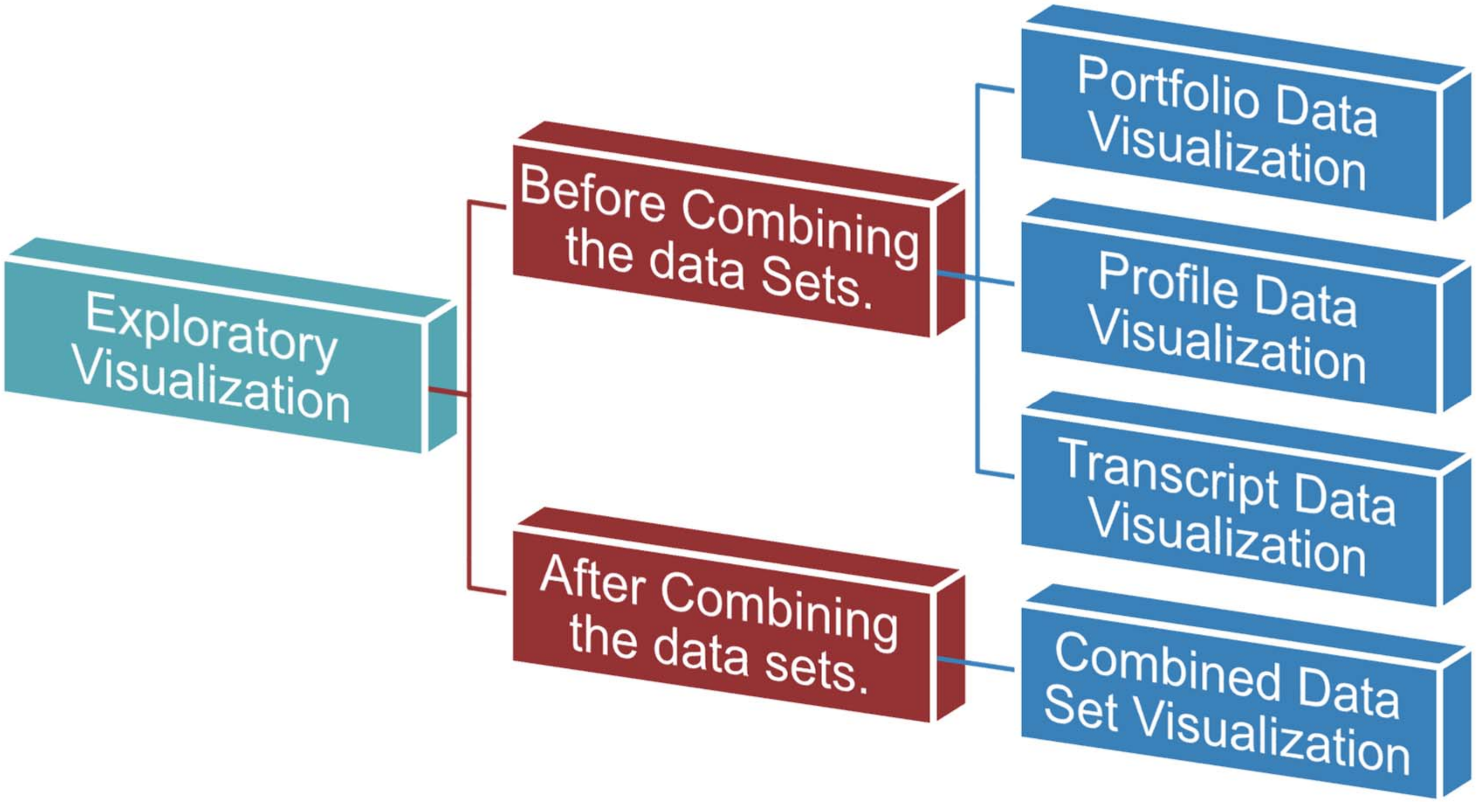
3-Dropping the Channels Column.

	difficulty	duration	offer_id	offer_type	reward	web	email	mobile	social
0	10	168	ae264e3637204a6fb9bb56bc8210ddfd	bogo	10	0	1	1	1
1	10	120	4d5c57ea9a6940dd891ad53e9dbe8da0	bogo	10	1	1	1	1
2	0	96	3f207df678b143eea3cee63160fa8bed	informational	0	1	1	1	0
3	5	168	9b98b8c7a33c4b65b9aebfe6a799e6d9	bogo	5	1	1	1	0
4	20	240	0b1e1539f2cc45b7b9fa7c272da2e1d7	discount	5	1	1	0	0
5	7	168	2298d6c36e964ae4a3e7e9706d1fb8c2	discount	3	1	1	1	1
6	10	240	fafdcd668e3743c1bb461111dcafc2a4	discount	2	1	1	1	1
7	0	72	5a8bc65990b245e5a138643cd4eb9837	informational	0	0	1	1	1
8	5	120	f19421c1d4aa40978ebb69ca19b0e20d	bogo	5	1	1	1	1
9	10	168	2906b810c7d4411798c6938adc9daaa5	discount	2	1	1	1	0

02

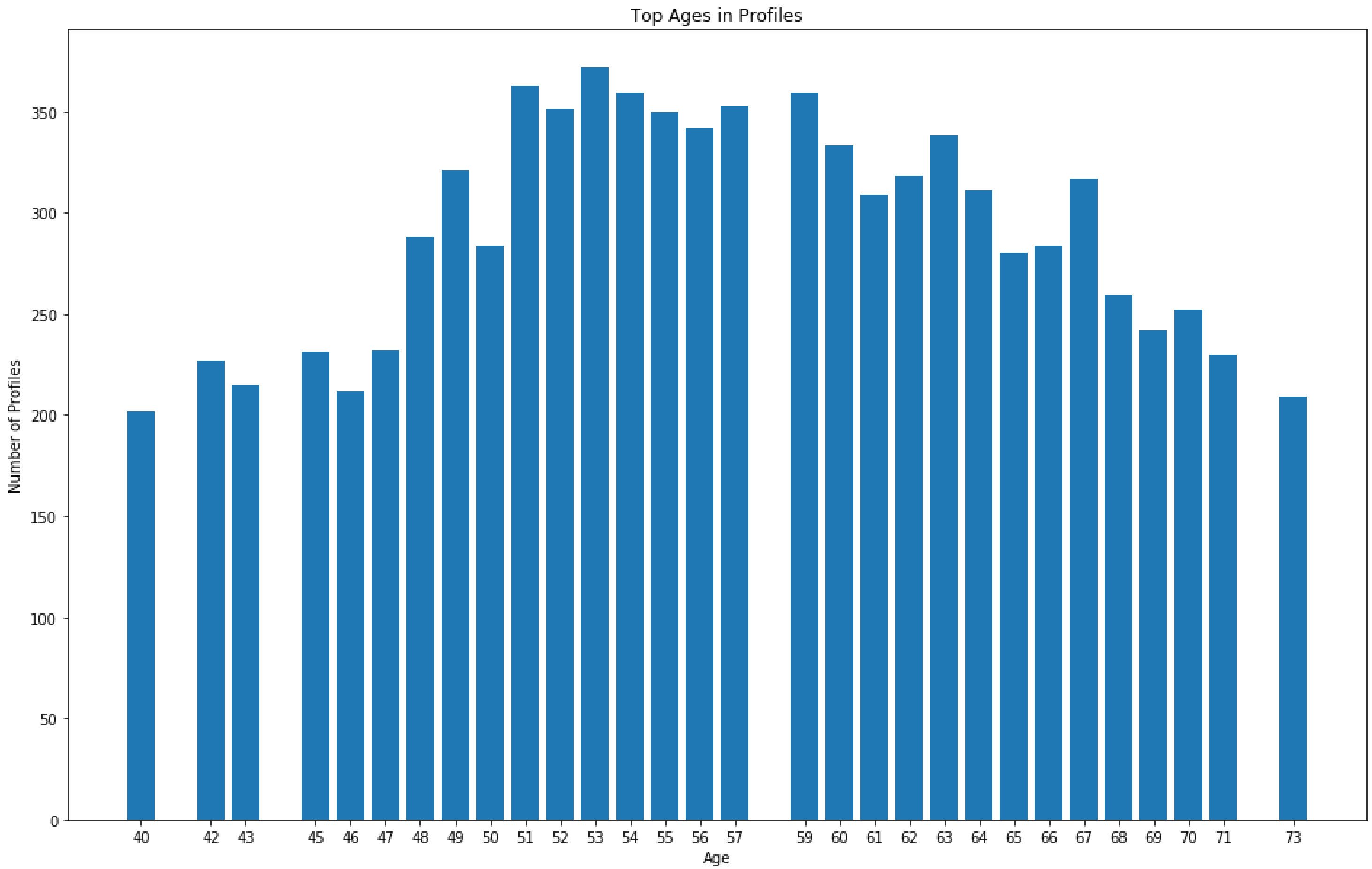
Exploratory Visualization

Exploratory Visualization



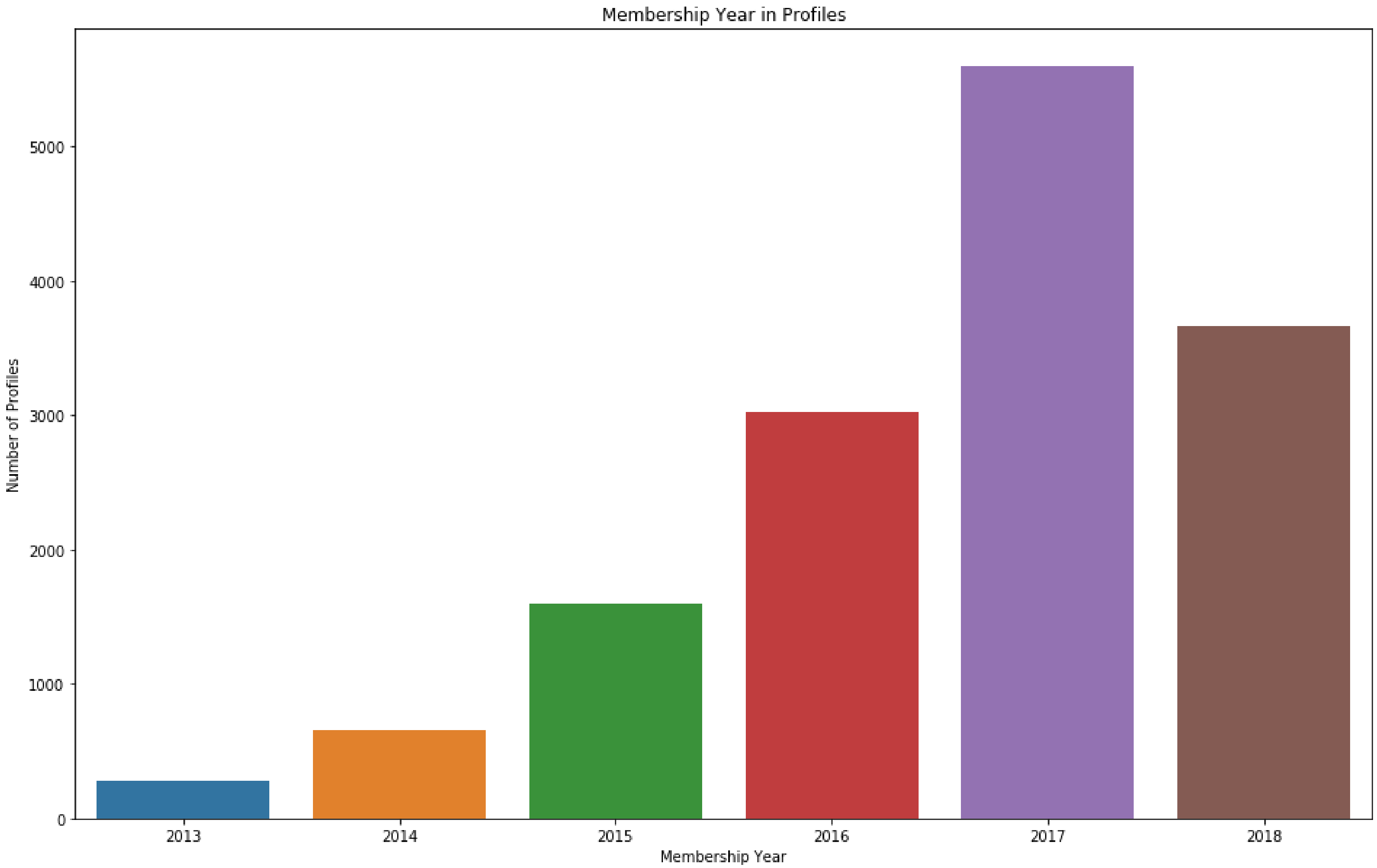
Exploratory Visualization before Merging the data sets

Profile Data Visualization:



Exploratory Visualization before Merging the data sets

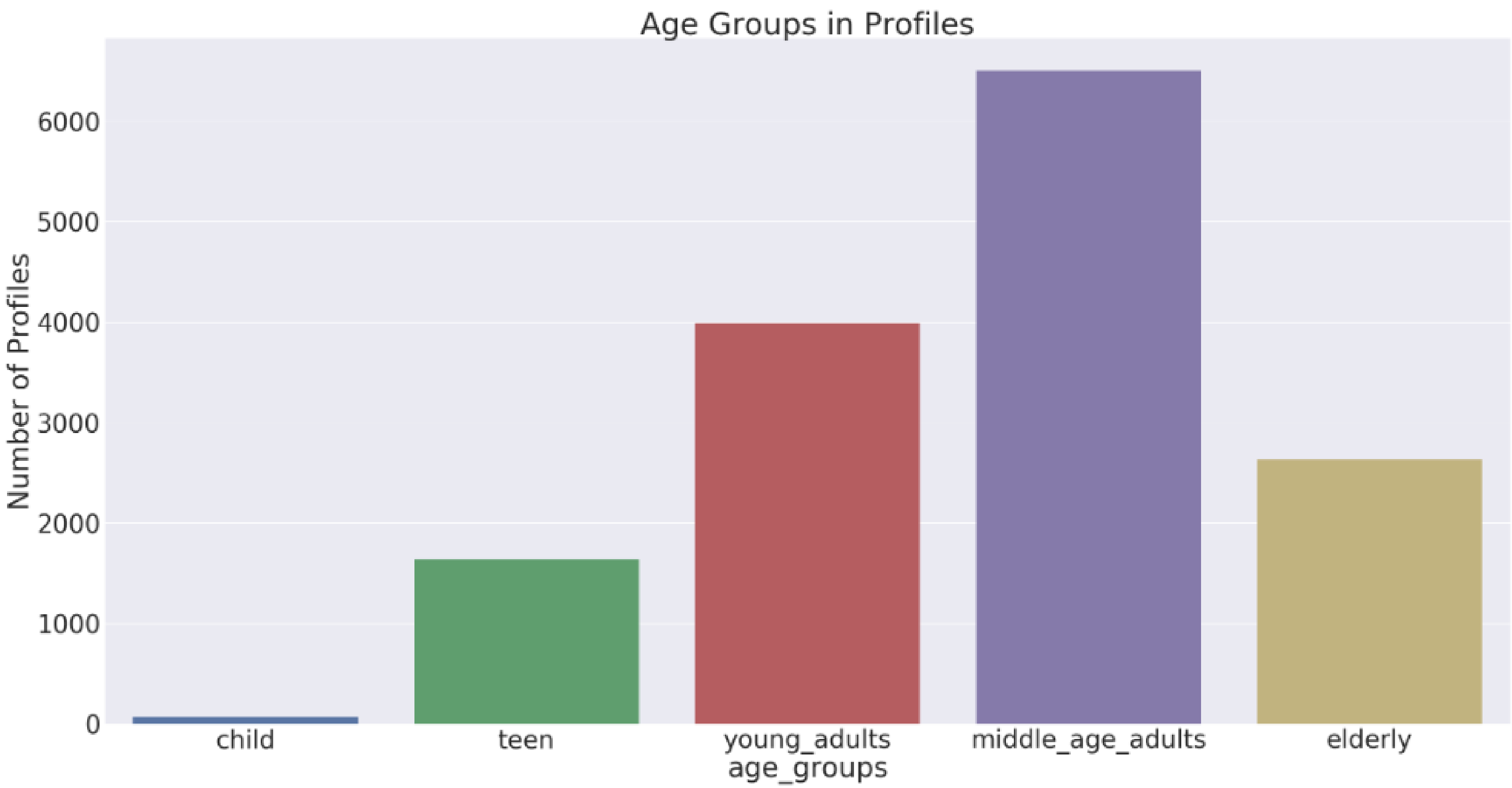
Profile Data Visualization:



The relation between member ship year and number of Profiles

Exploratory Visualization before Merging the data sets

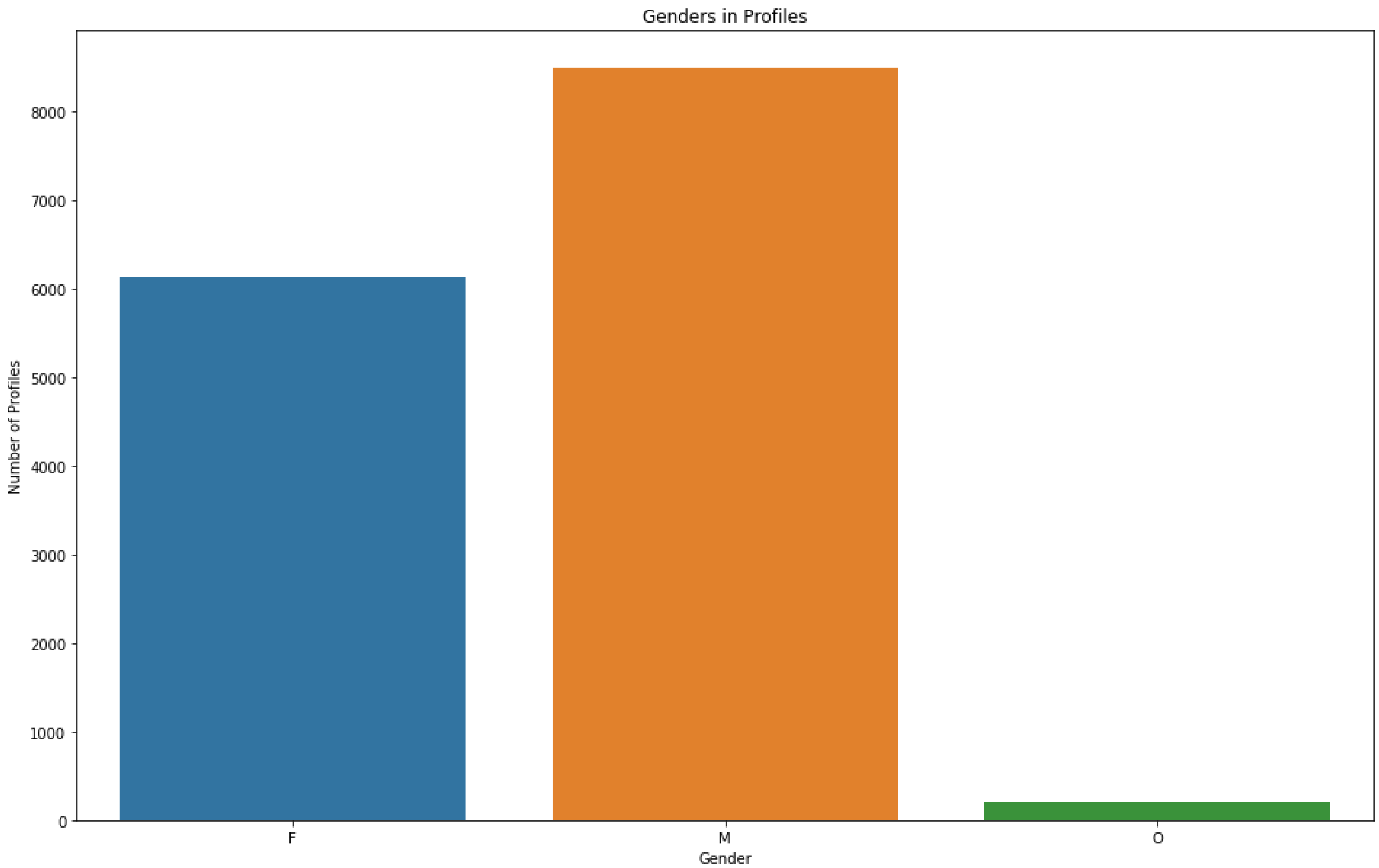
Profile Data Visualization:



The relation between age groups and number of Profiles

Exploratory Visualization before Merging the data sets

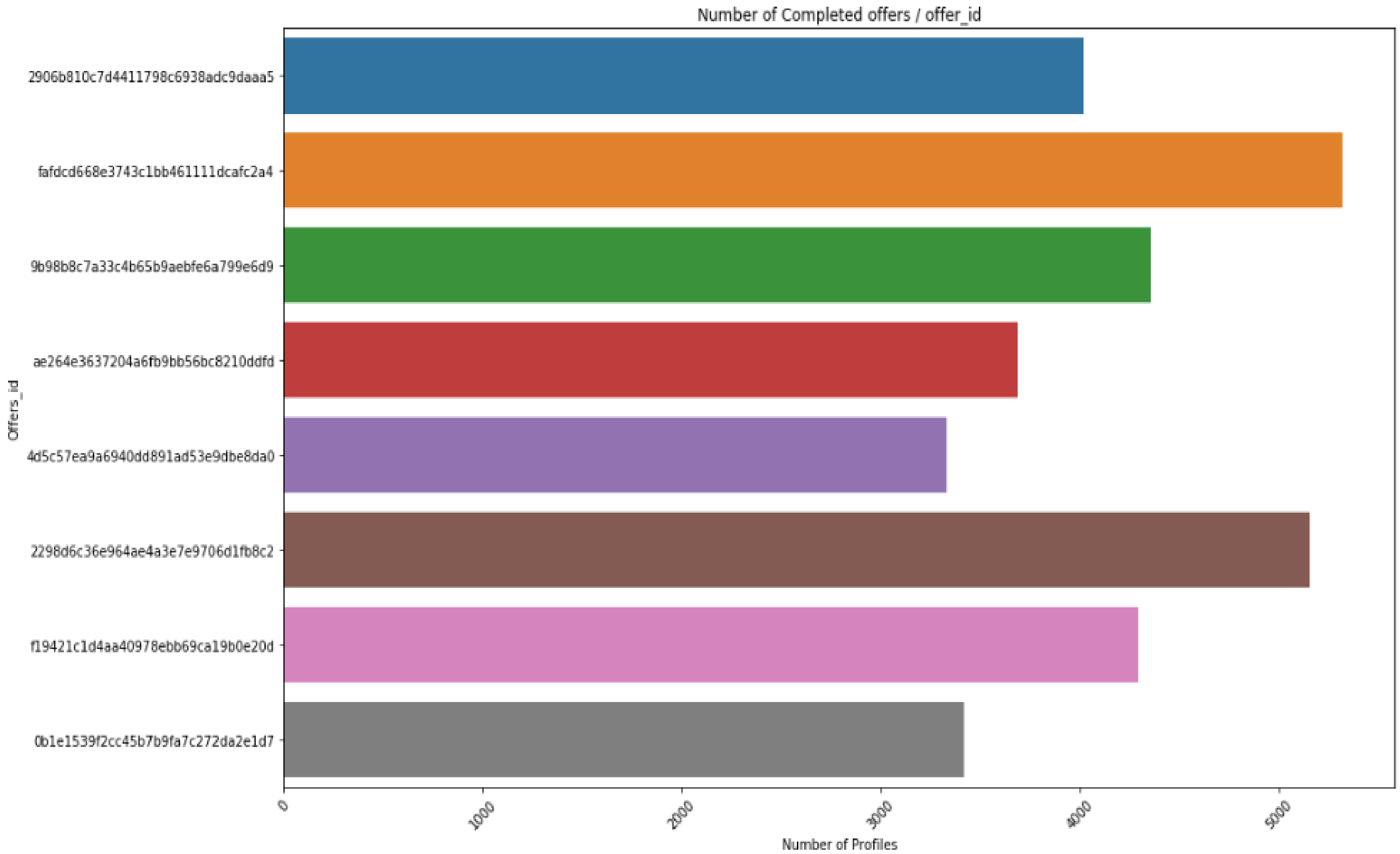
Profile Data Visualization:



The relation between gender and number of Profiles

Exploratory Visualization before Merging the data sets

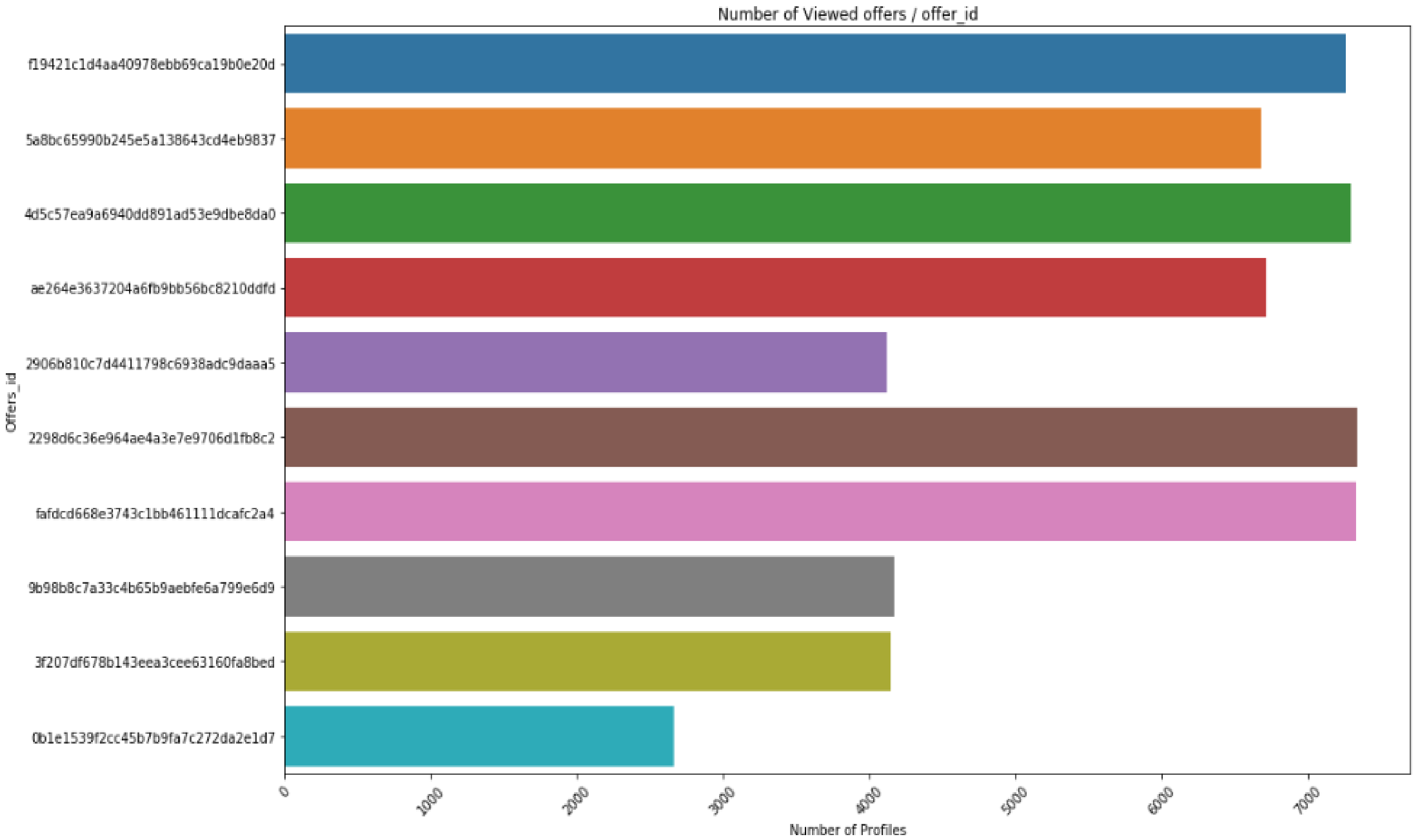
Transcript Data Visualization:



The relation between offer id's of Completed offers and number of Profiles

Exploratory Visualization
before Merging the data sets

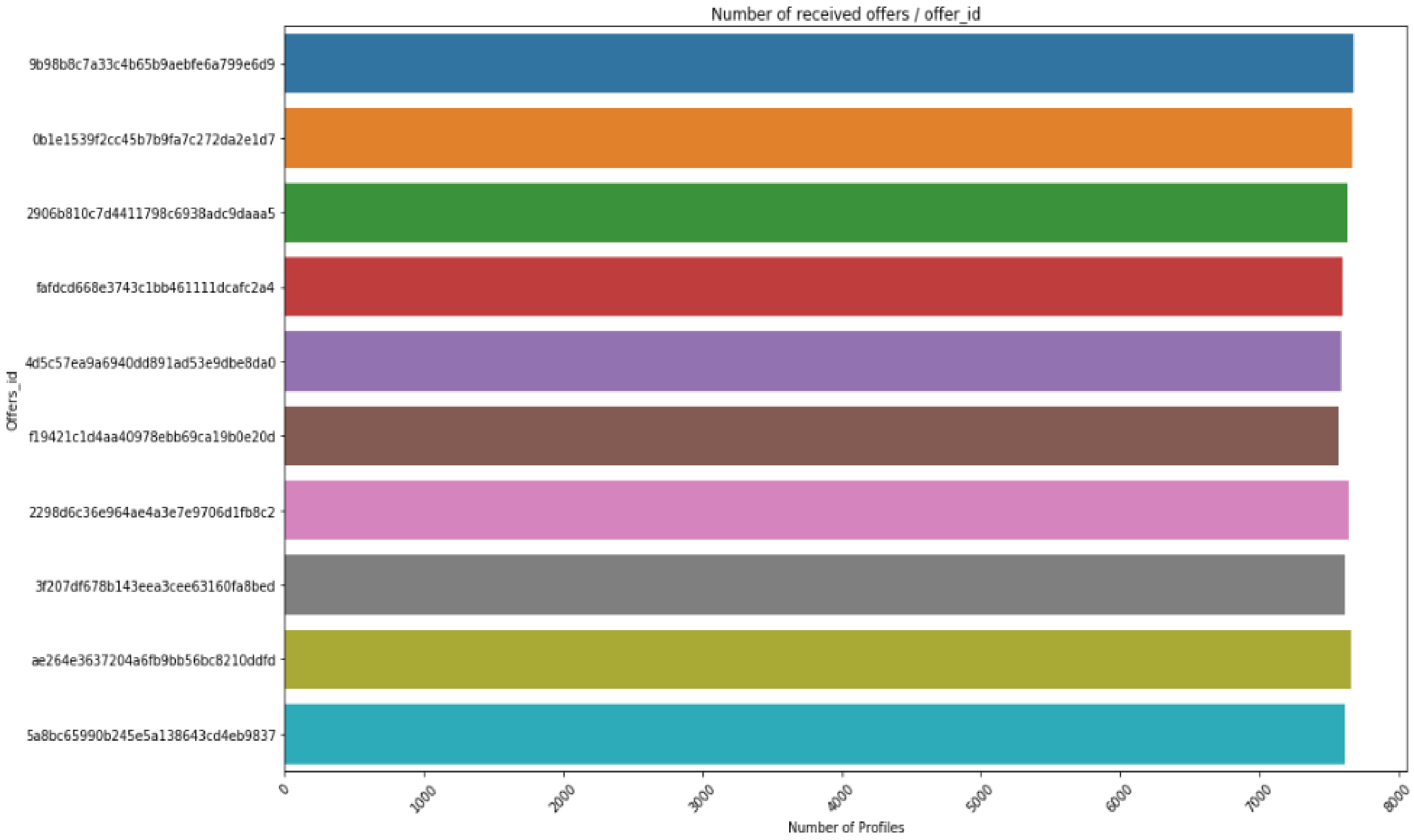
Transcript Data Visualization:



The relation between offer id's of viewed offers and number of Profiles

Exploratory Visualization before Merging the data sets

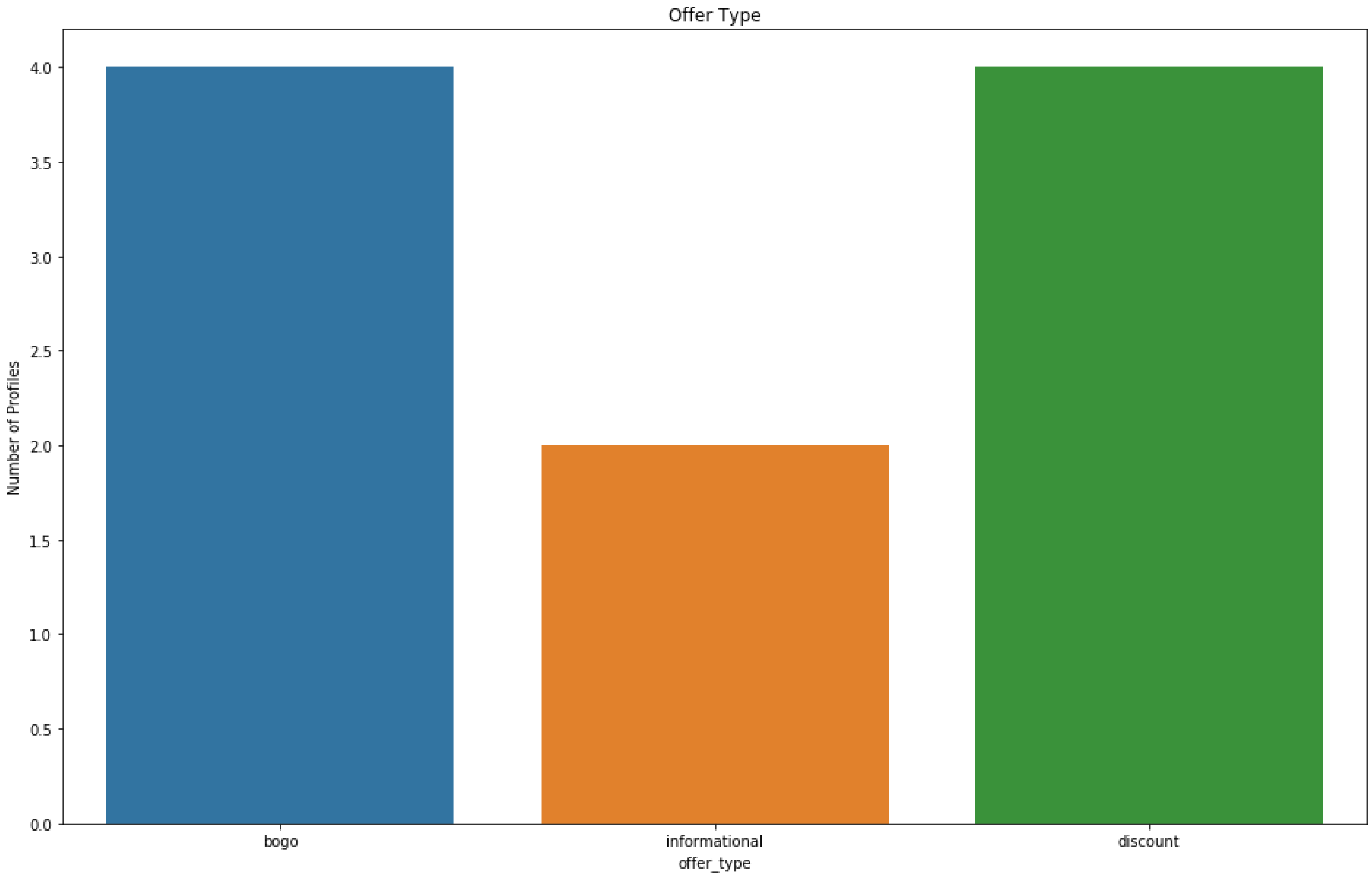
Transcript Data Visualization:



The relation between offer id's of received offers and number of Profiles

Exploratory Visualization before Merging the data sets

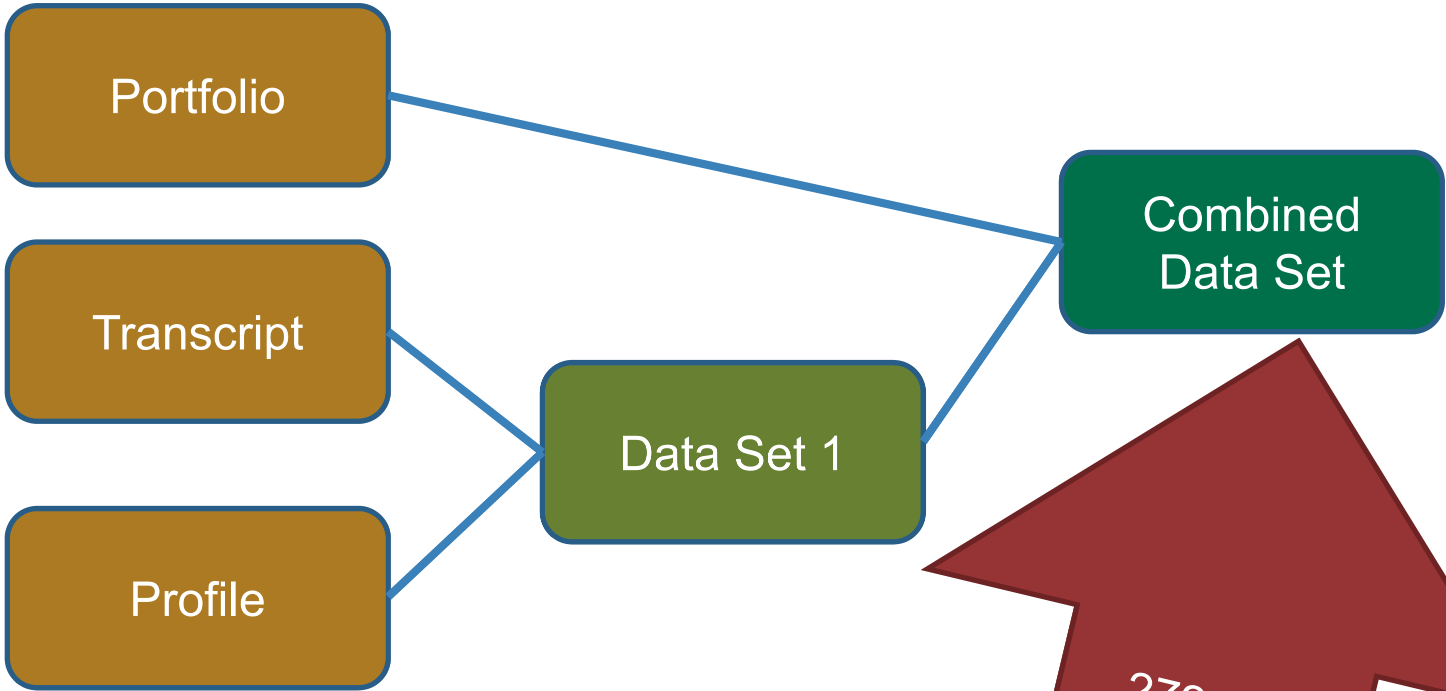
Portfolio Data Visualization:

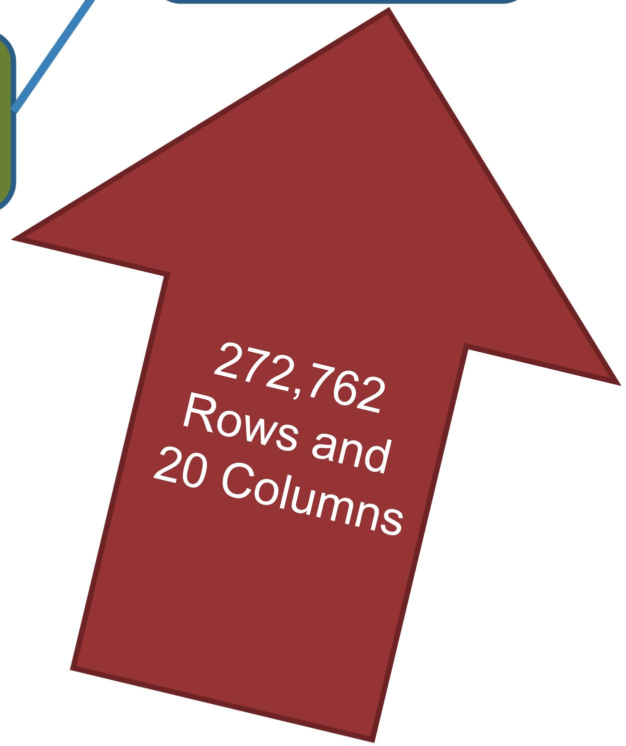


The relation between offer type and number of Profiles

Exploratory Visualization after Merging the data sets

Combined the data Sets(Combining Process):

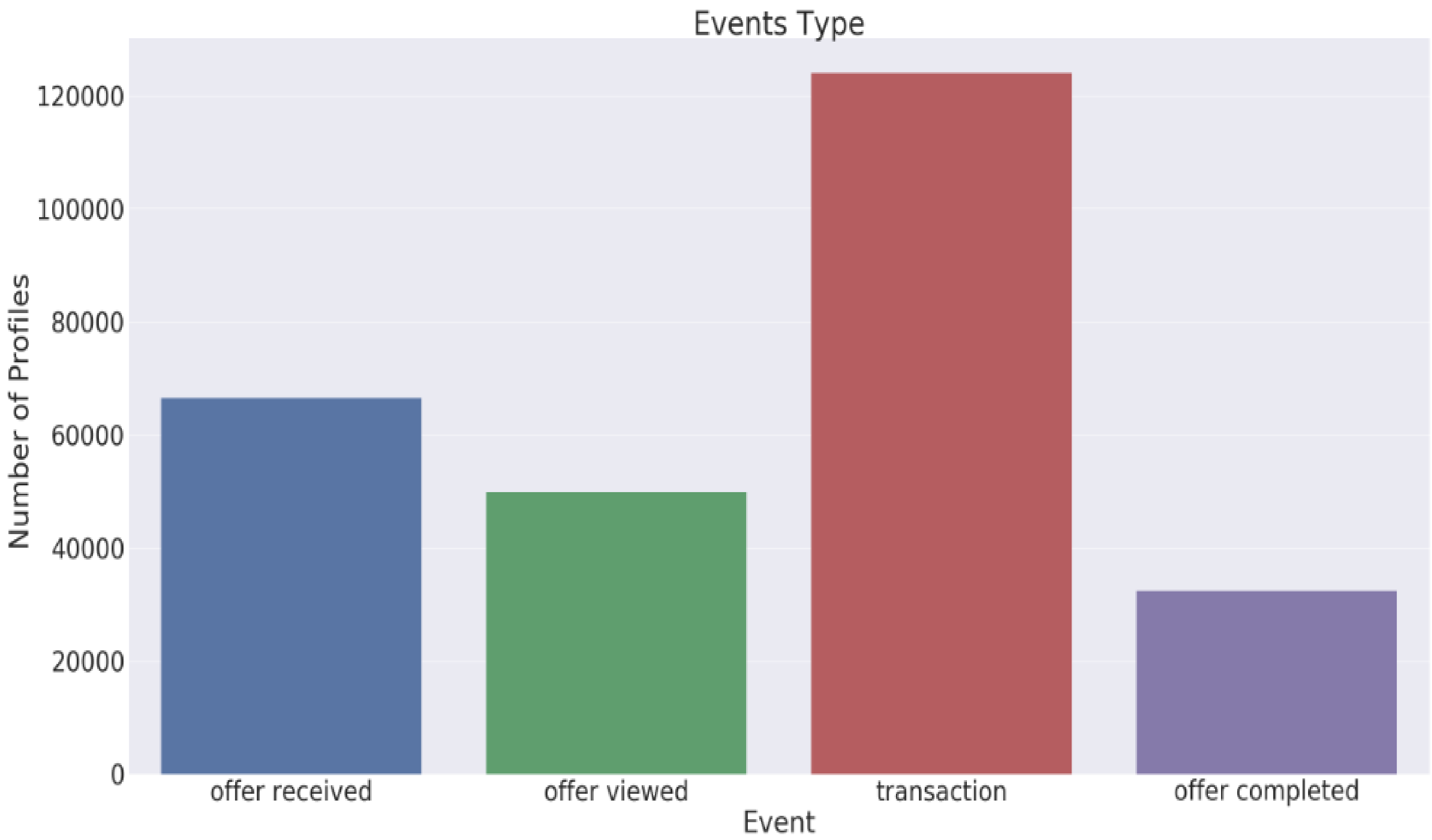





```
Combined_all_data.info()

<class 'pandas.core.frame.DataFrame'>
Int64Index: 272762 entries, 0 to 272761
Data columns (total 20 columns):
event                     272762 non-null object
customer                  272762 non-null object
time                      272762 non-null int64
offer_id                  272762 non-null object
amount                    272762 non-null object
age                       272762 non-null int64
became_member_on          272762 non-null datetime64[ns]
gender                    272762 non-null object
income                    272762 non-null float64
age_groups                272762 non-null category
member_launch_Cum_days    272762 non-null int64
member_launch_year        272762 non-null int64
difficulty                148805 non-null float64
duration                  148805 non-null float64
offer_type                148805 non-null object
reward                    148805 non-null float64
web                       148805 non-null float64
email                     148805 non-null float64
mobile                    148805 non-null float64
social                    148805 non-null float64
dtypes: category(1), datetime64[ns](1), float64(8), int64(4), object(6)
memory usage: 41.9+ MB
```

Exploratory Visualization
after Merging the data sets

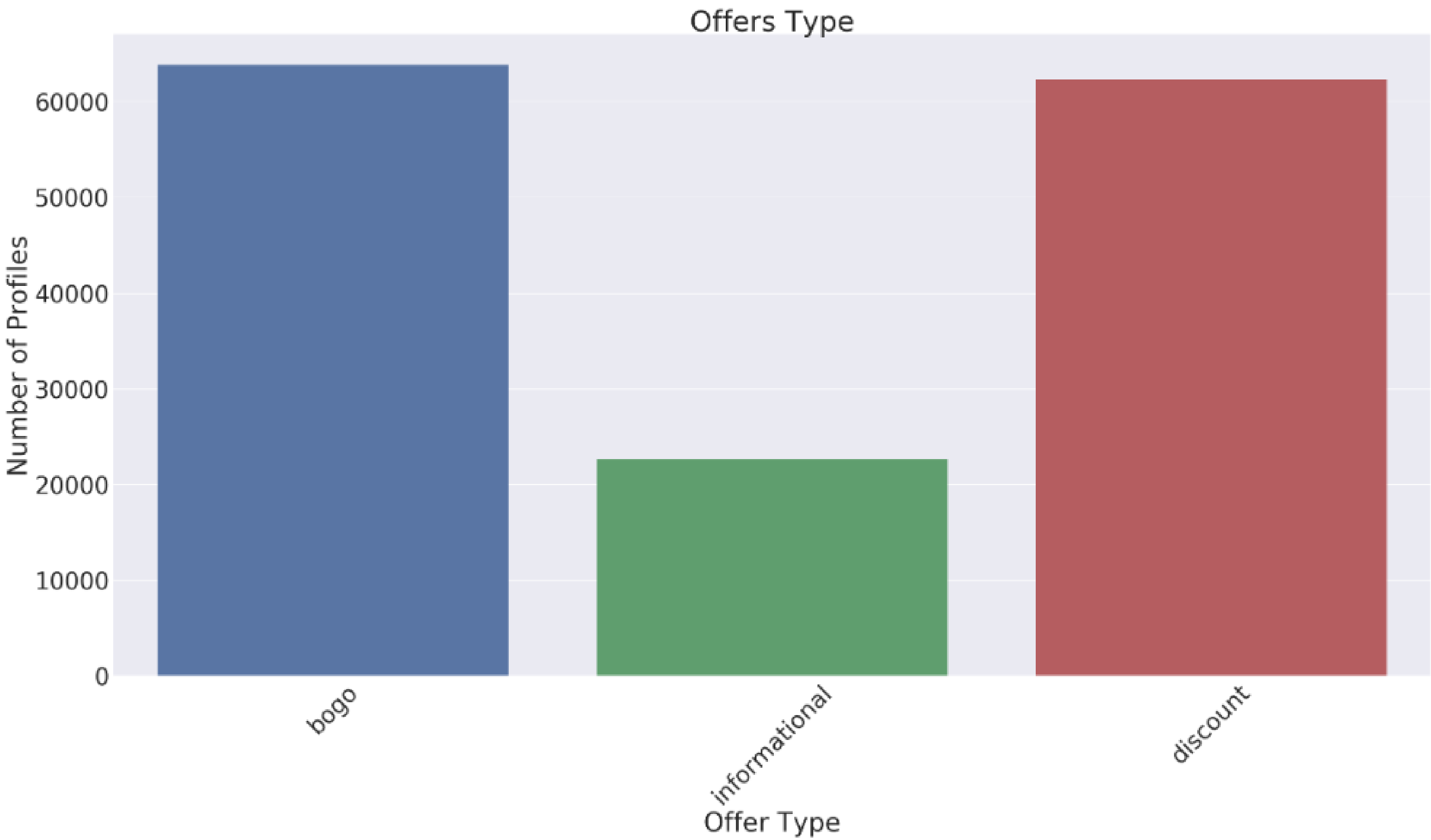
Combined data Sets:



The relation between events type and number of Profiles

Exploratory Visualization after Merging the data sets

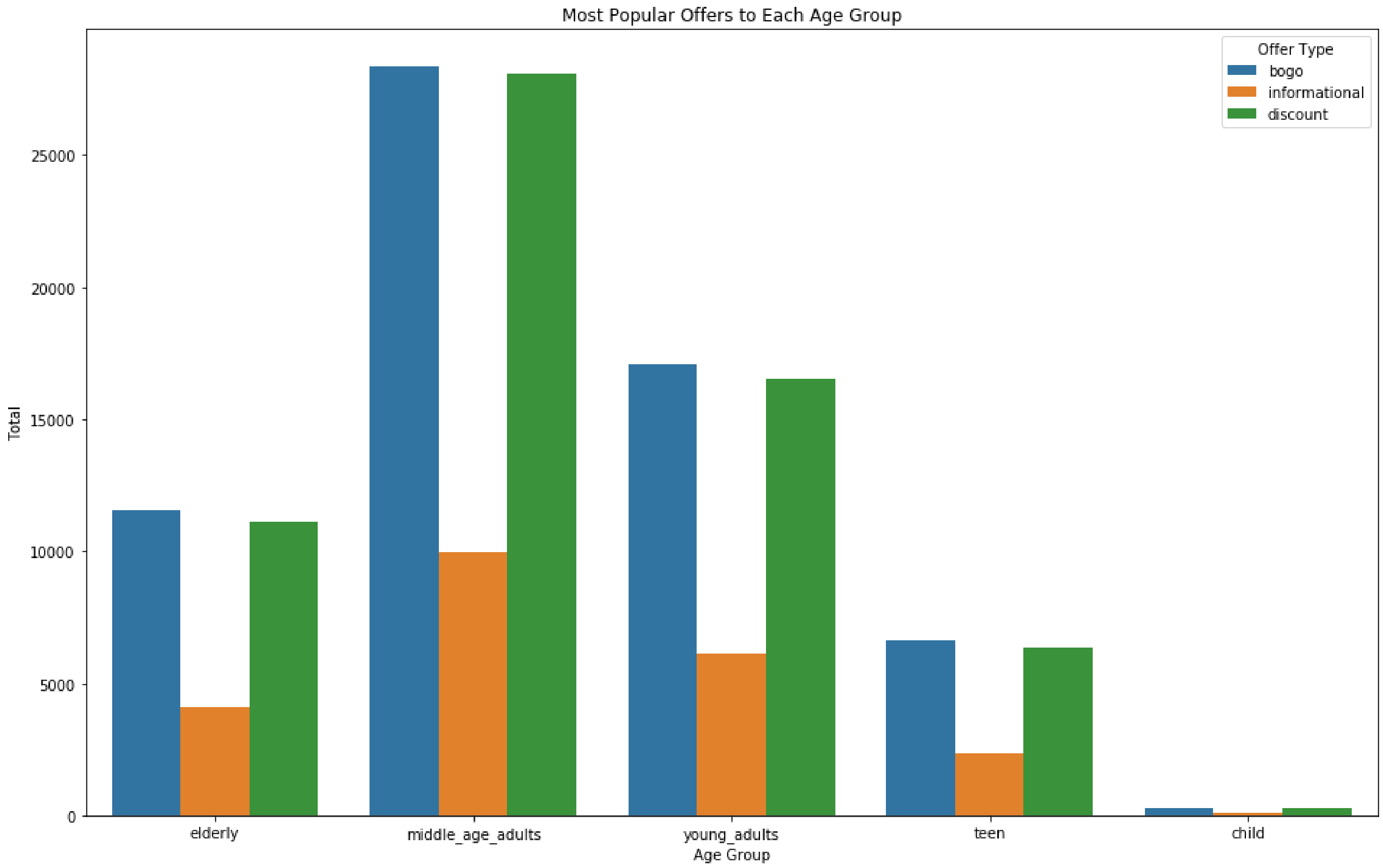
Combined data Sets:



The relation between offer type and number of Profiles

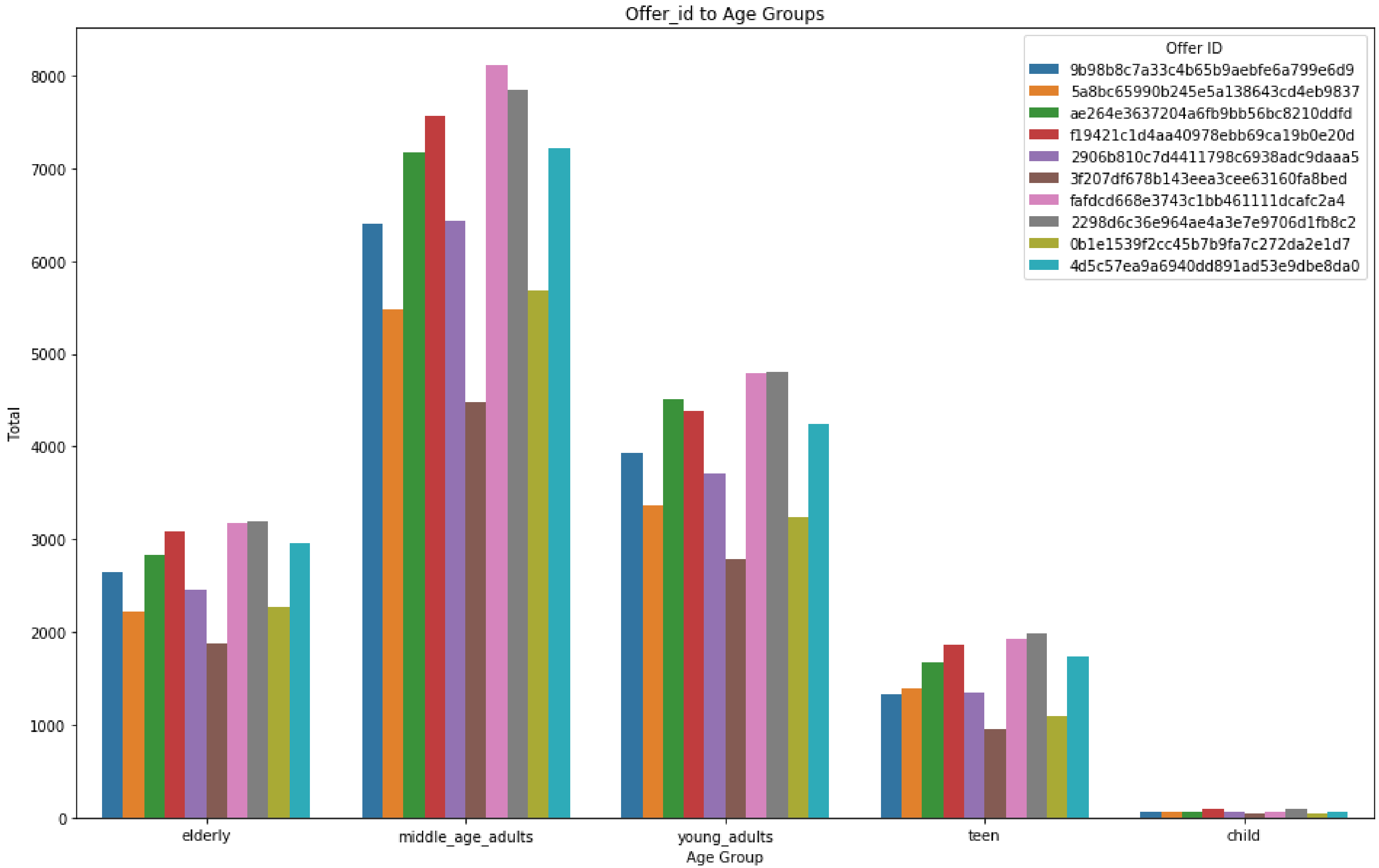
Exploratory Visualization after Merging the data sets

Combined data Sets:



Exploratory Visualization after Merging the data sets

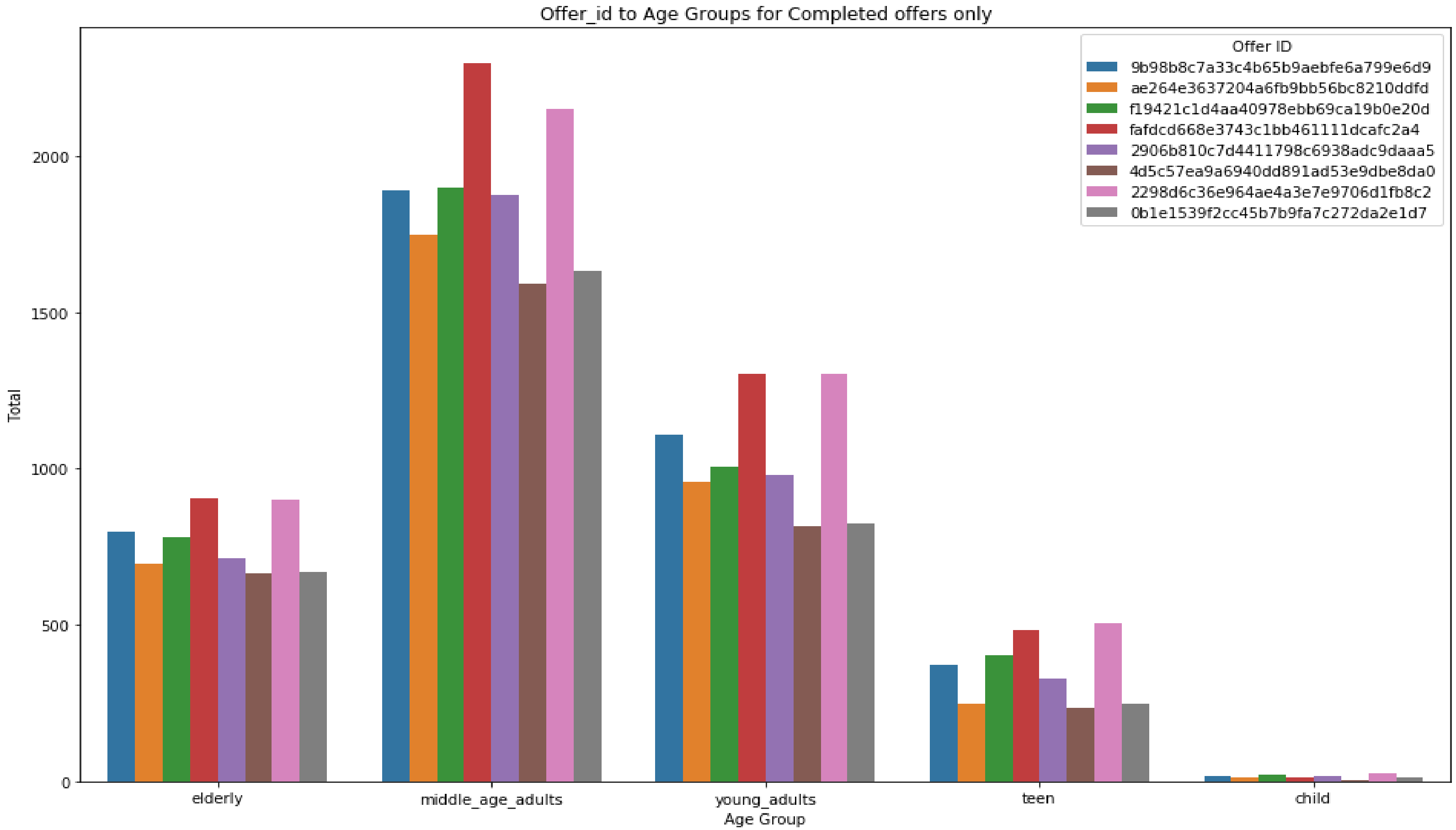
Combined data Sets:



As Shown , the most of offers come from middle age adults

Exploratory Visualization after Merging the data sets

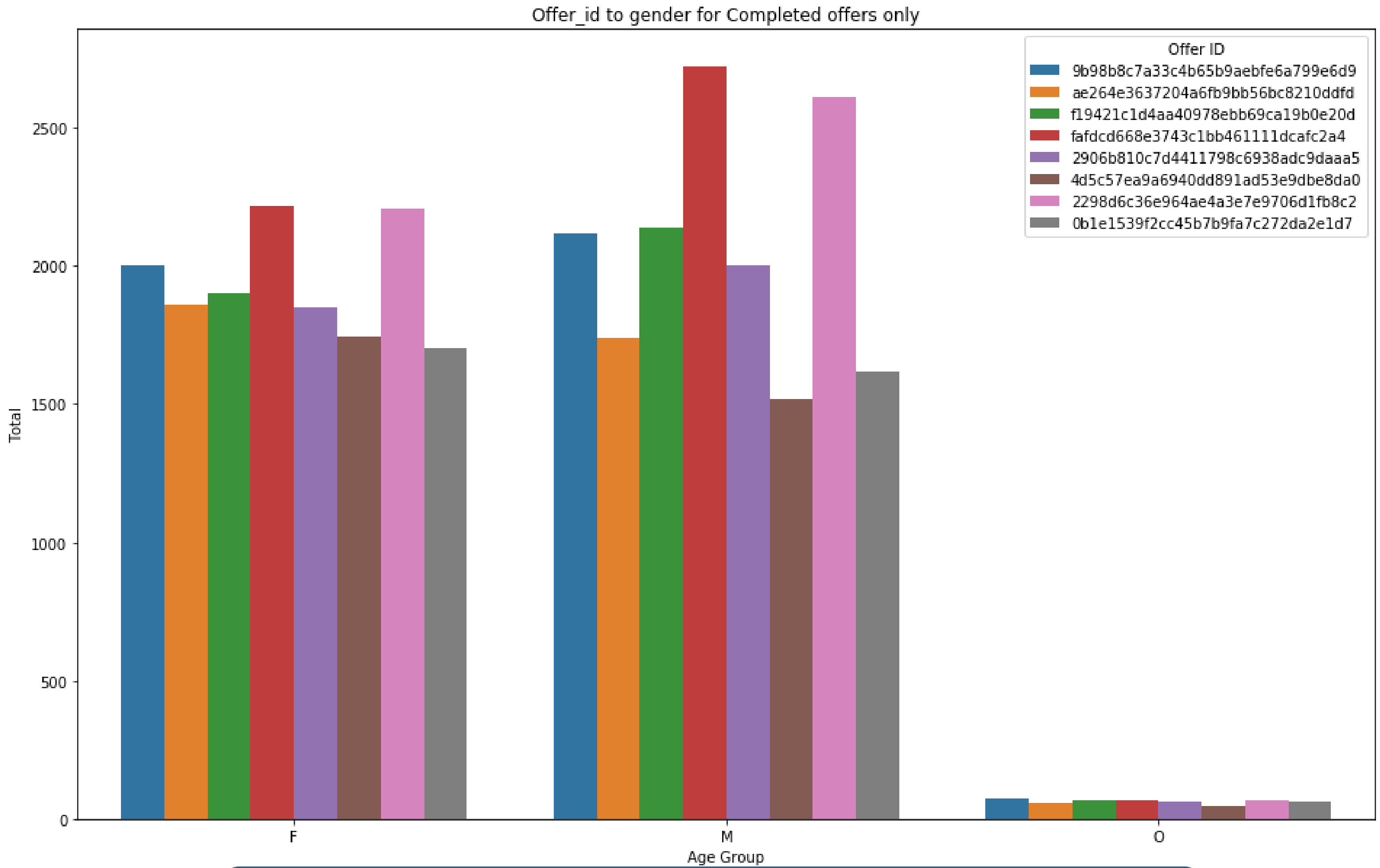
<u>Combined data Sets:</u>



As Shown , the most of Completed offers come from middle age adults

Exploratory Visualization after Merging the data sets

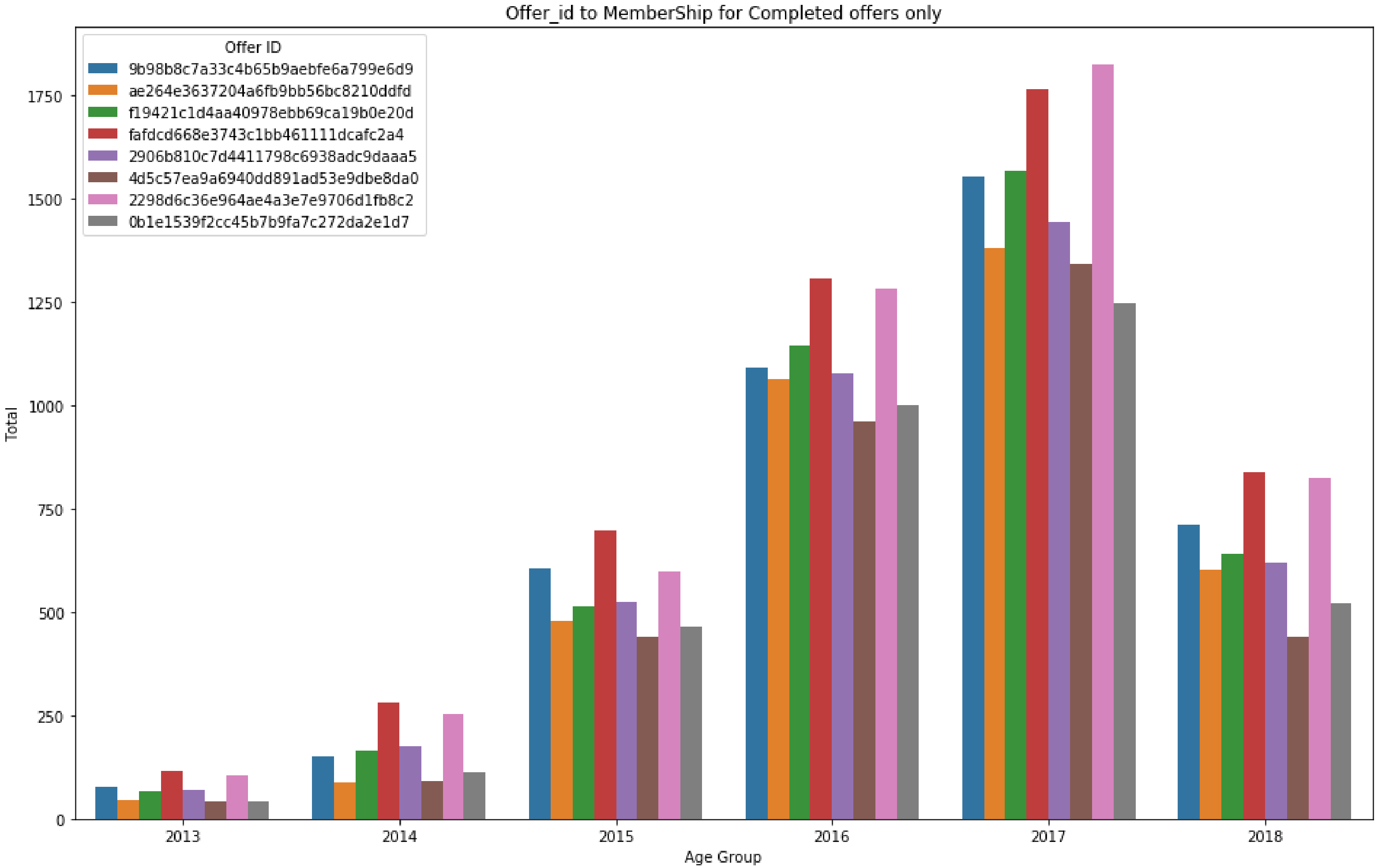
Combined data Sets:



As Shown , the most of completed offers come from Males

Exploratory Visualization after Merging the data sets

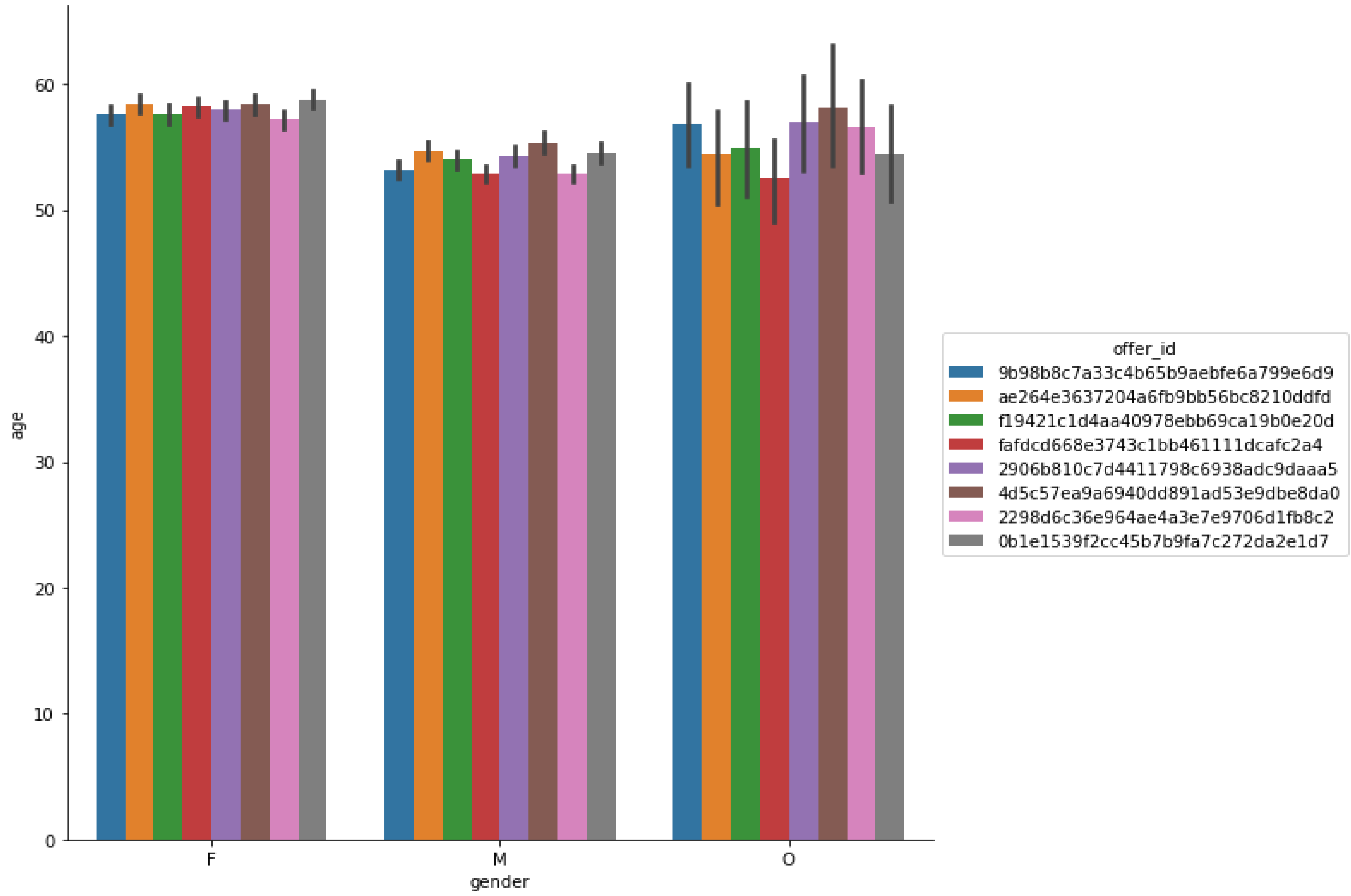
Combined data Sets:



As Shown , the most of completed offers come from Customers with membership starts in 2017

Exploratory Visualization after Merging the data sets

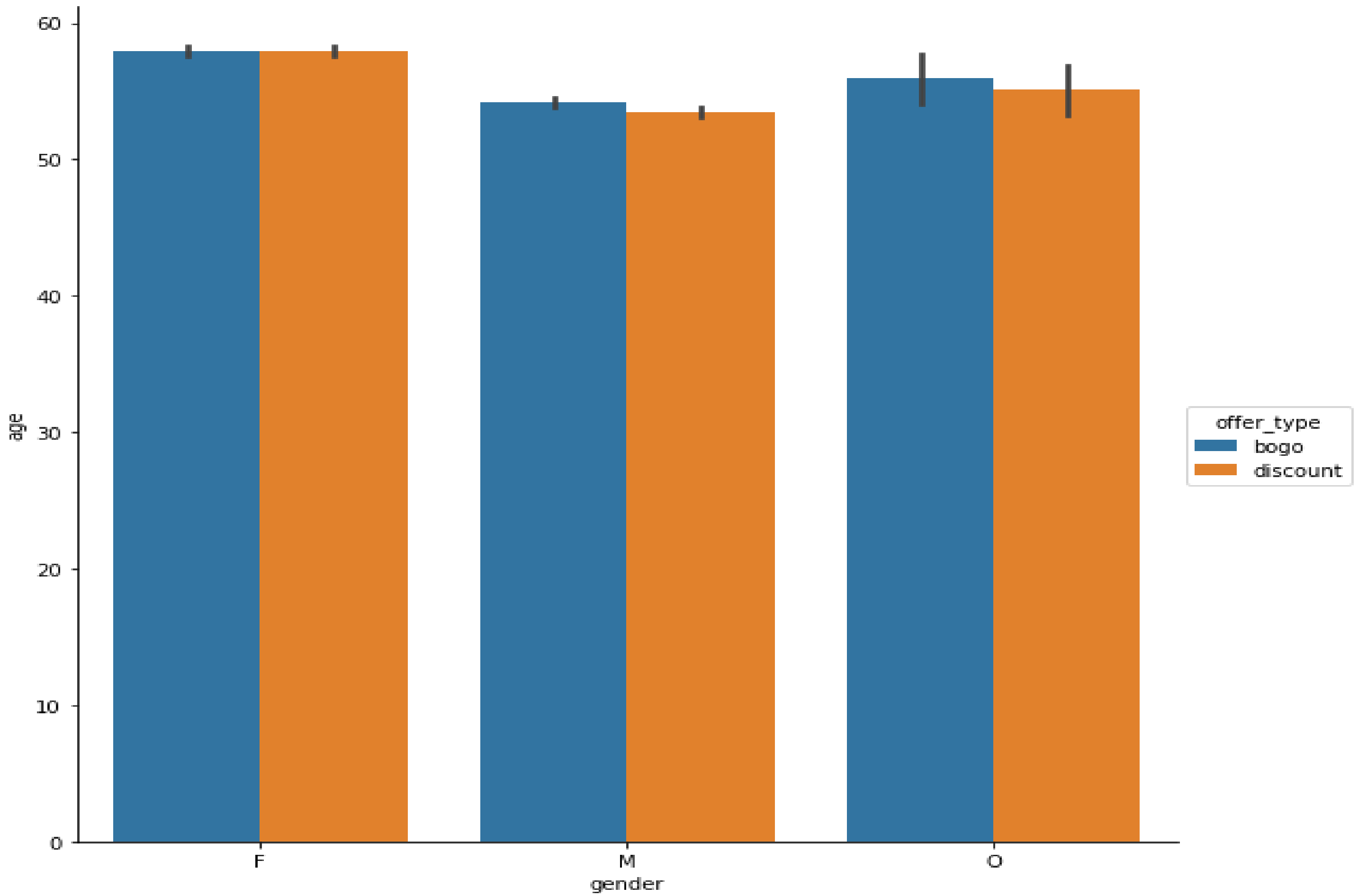
Combined data Sets:



Offers Distribution according to gender and age

Exploratory Visualization after Merging the data sets

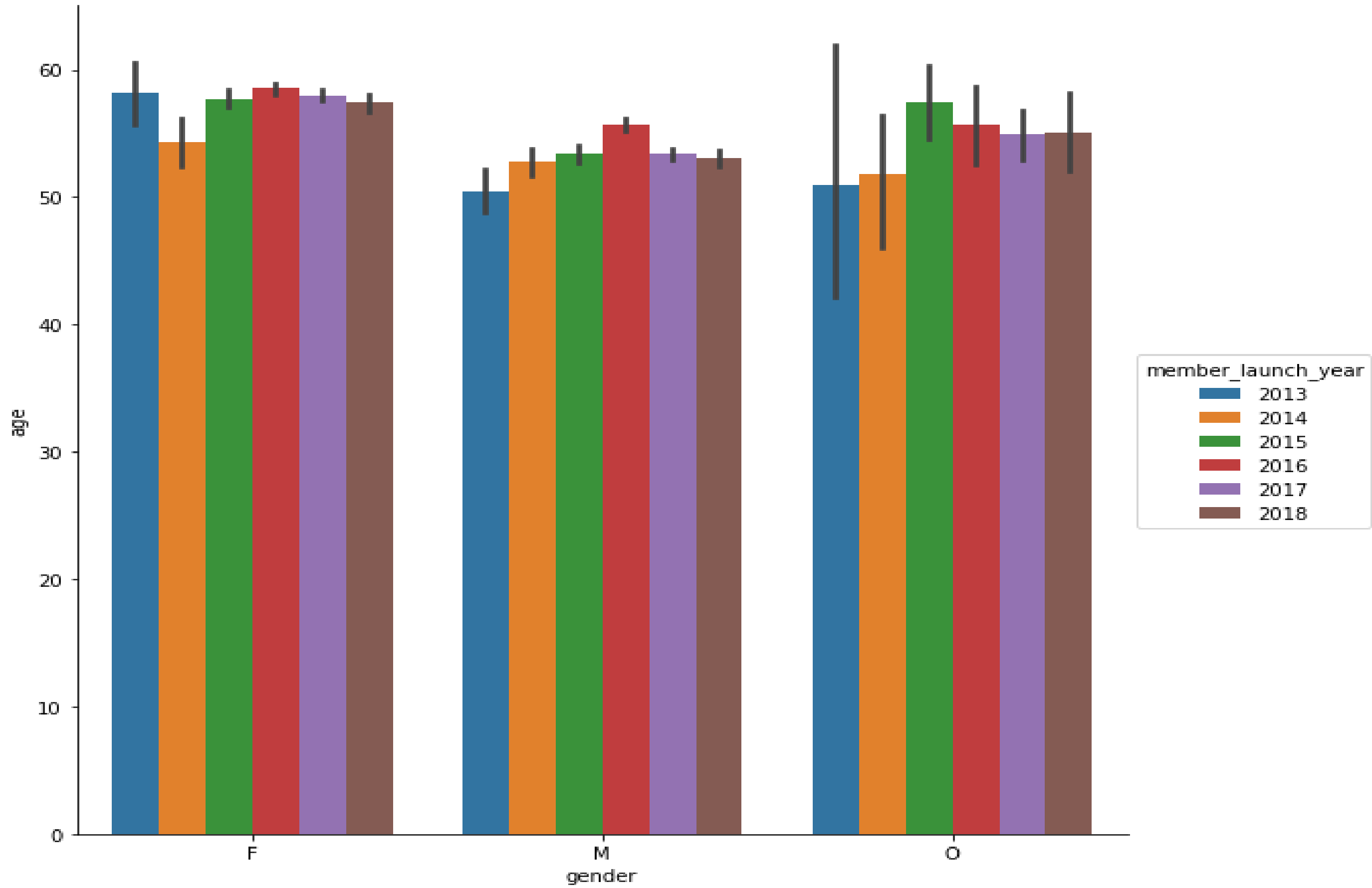
<u>Combined data Sets:</u>



Offers type Distribution according to gender and age

Exploratory Visualization after Merging the data sets

Combined data Sets:



Member launch year Distribution according to gender and age

Exploratory Visualization after Merging the data sets

Combined Data Statistics

Events Type VS Gender

For Females:
Number of offer received: 27456, 43.1% of total offers.
Number of offer viewed: 20786, 32.6% of total offers.
Number of offer completed: 15477 , 56.4% of received offers.

For Males:
Number of offer received: 38129, 46.0% of total offers.
Number of offer viewed: 28301, 34.1% of total offers.
Number of offer completed: 16466, 43.2% of received offers.

Exploratory Visualization after Merging the data sets

Combined Data Statistics

TOP-10 Customers
[#1]
Person: 3c8d541112a74af99e88abbd0692f00e
Number of Completed Offers: 5 offers
Amount: $1609.0
[#2]
Person: f1d65ae63f174b8f80fa063adcaa63b7
Number of Completed Offers: 6 offers
Amount: $1366.0
[#3]
Person: ae6f43089b674728a50b8727252d3305
Number of Completed Offers: 3 offers
Amount: $1328.0
[#4]
Person: 626df8678e2a4953b9098246418c9cfa
Number of Completed Offers: 4 offers
Amount: $1321.0
[#5]
Person: 73afdeca19e349b98f09e928644610f8
Number of Completed Offers: 5 offers
Amount: $1320.0
[#6]
Person: 52959f19113e4241a8cb3bef486c6412
Number of Completed Offers: 5 offers
Amount: $1293.0
[#7]
Person: ad1f0a409ae642bc9a43f31f56c130fc
Number of Completed Offers: 3 offers
Amount: $1258.0
[#8]
Person: d240308de0ee4cf8bb6072816268582b
Number of Completed Offers: 5 offers
Amount: $1252.0
[#9]
Person: 946fc0d3ecc4492aa4cc06cf6b1492c3
Number of Completed Offers: 4 offers
Amount: $1232.0
[#10]
Person: 6406abad8e2c4b8584e4f68003de148d
Number of Completed Offers: 3 offers
Amount: $1212.0

Top 10 Customers related to Offers Completion

Exploratory Visualization after Merging the data sets

Combined Data Statistics

The Consumed time to Complete offer VS Genders

The Maximum value to Complete offer for Females: 428.0 Hours and the Value by days is : 17.8 days
The Maximum value to Complete offer for Males: 434.0 Hours and the Value by days is : 18.1 days

Exploratory Visualization after Merging the data sets

Combined Data Statistics

Offer IDs (10 offers) VS events

offer ID:0b1e1539f2cc45b7b9fa7c272da2e1d7 Total number of offers: 12327

offer ID:0b1e1539f2cc45b7b9fa7c272da2e1d7 Total number of Completed offers: 3386 and Percentage is: 27.468159325058817 %

offer ID:0b1e1539f2cc45b7b9fa7c272da2e1d7 Total number of Viewed offers: 2215 and Percentage is: 17.96868662286039 %

offer ID:0b1e1539f2cc45b7b9fa7c272da2e1d7 Total number of received offers: 6726 and Percentage is: 54.5631540520808 %

offer ID:2298d6c36e964ae4a3e7e9706d1fb8c2 Total number of offers: 17920

offer ID:2298d6c36e964ae4a3e7e9706d1fb8c2 Total number of Completed offers: 4886 and Percentage is: 27.265625 %

offer ID:2298d6c36e964ae4a3e7e9706d1fb8c2 Total number of Viewed offers: 6379 and Percentage is: 35.597098214285715 %

offer ID:2298d6c36e964ae4a3e7e9706d1fb8c2 Total number of received offers: 6655 and Percentage is: 37.137276785714285 %

offer ID:2906b810c7d4411798c6938adc9daaa5 Total number of offers: 14002

offer ID:2906b810c7d4411798c6938adc9daaa5 Total number of Completed offers: 3911 and Percentage is: 27.93172403942294 %

offer ID:2906b810c7d4411798c6938adc9daaa5 Total number of Viewed offers: 3460 and Percentage is: 24.710755606341948 %

offer ID:2906b810c7d4411798c6938adc9daaa5 Total number of received offers: 6631 and Percentage is: 47.35752035423511 %

Exploratory Visualization after Merging the data sets

Combined Data Statistics

offer ID:3f207df678b143eea3cee63160fa8bed Total number of offers: 10144

offer ID:3f207df678b143eea3cee63160fa8bed Total number of Viewed offers: 3487 and Percentage is: 34.375 %

offer ID:3f207df678b143eea3cee63160fa8bed Total number of received offers: 6657 and Percentage is: 65.625 %

offer ID:4d5c57ea9a6940dd891ad53e9dbe8da0 Total number of offers: 16232

offer ID:4d5c57ea9a6940dd891ad53e9dbe8da0 Total number of Completed offers: 3310 and Percentage is: 20.39181862986693 %

offer ID:4d5c57ea9a6940dd891ad53e9dbe8da0 Total number of Viewed offers: 6329 and Percentage is: 38.99088220798423 %

offer ID:4d5c57ea9a6940dd891ad53e9dbe8da0 Total number of received offers: 6593 and Percentage is: 40.61729916214884 %

offer ID:5a8bc65990b245e5a138643cd4eb9837 Total number of offers: 12516

offer ID:5a8bc65990b245e5a138643cd4eb9837 Total number of Viewed offers: 5873 and Percentage is: 46.92393736017897 %

offer ID:5a8bc65990b245e5a138643cd4eb9837 Total number of received offers: 6643 and Percentage is: 53.07606263982103 %

offer ID:9b98b8c7a33c4b65b9aebfe6a799e6d9 Total number of offers: 14372

offer ID:9b98b8c7a33c4b65b9aebfe6a799e6d9 Total number of Completed offers: 4188 and Percentage is: 29.13999443362093 %

offer ID:9b98b8c7a33c4b65b9aebfe6a799e6d9 Total number of Viewed offers: 3499 and Percentage is: 24.345950459226273 %

offer ID:9b98b8c7a33c4b65b9aebfe6a799e6d9 Total number of received offers: 6685 and Percentage is: 46.5140551071528 %

Exploratory Visualization after Merging the data sets

Combined Data Statistics

offer ID:ae264e3637204a6fb9bb56bc8210ddfd Total number of offers: 16241

offer ID:ae264e3637204a6fb9bb56bc8210ddfd Total number of Completed offers: 3657 and Percentage is: 22.51708638630626 %

offer ID:ae264e3637204a6fb9bb56bc8210ddfd Total number of Viewed offers: 5901 and Percentage is: 36.333969583153745 %

offer ID:ae264e3637204a6fb9bb56bc8210ddfd Total number of received offers: 6683 and Percentage is: 41.148944030539994 %

offer ID:f19421c1d4aa40978ebb69ca19b0e20d Total number of offers: 16989

offer ID:f19421c1d4aa40978ebb69ca19b0e20d Total number of Completed offers: 4103 and Percentage is: 24.150921184295722 %

offer ID:f19421c1d4aa40978ebb69ca19b0e20d Total number of Viewed offers: 6310 and Percentage is: 37.14167991053034 %

offer ID:f19421c1d4aa40978ebb69ca19b0e20d Total number of received offers: 6576 and Percentage is: 38.70739890517393 %

offer ID:fafdcd668e3743c1bb461111dcafc2a4 Total number of offers: 18062

offer ID:fafdcd668e3743c1bb461111dcafc2a4 Total number of Completed offers: 5003 and Percentage is: 27.699036651533604 %

offer ID:fafdcd668e3743c1bb461111dcafc2a4 Total number of Viewed offers: 6407 and Percentage is: 35.47226220795039 %

offer ID:fafdcd668e3743c1bb461111dcafc2a4 Total number of received offers: 6652 and Percentage is: 36.828701140516 %

03

Algorithms and Techniques

Algorithms and Techniques

As we are implementing a Classification Problem , we will implement the models in the following slides, and by comparing the results and our Evaluation metrics to our Benchmark model , we can know which is the best model to be implemented to our Problem.

Admittedly , we will concentrate on the Gradient Boosting Models like XGBoost, Cat Boost and LightGBM which Often provides predictive accuracy that cannot be beat , Lots of flexibility - can optimize on different loss functions and provides several hyperparameters tuning options that make the function fit very flexible , No data pre-processing required - often works great with categorical and numerical values as is and Handles missing data .

Algorithms and Techniques

1.Amazon Sage maker XG-Boost built in Algorithm:

XGBoost (extreme gradient boosting) is a popular and efficient open-source implementation of the gradient-boosted trees algorithm. *Gradient boosting* is a machine learning algorithm that attempts to accurately predict target variables by combining the estimates of a set of simpler, weaker models. By applying gradient boosting to decision tree models in a highly scalable manner, XGBoost does remarkably well in machine learning competitions. It also robustly handles a variety of data types, relationships, and distributions. It provides a large number of hyperparameters—variables that can be tuned to improve model performance. This flexibility makes XGBoost a solid choice for various machine learning problems.

Algorithms and Techniques

LightGBM Model:

LightGBM is a gradient boosting framework that uses tree based learning algorithms. It is designed to be distributed and efficient with the following advantages:

-Faster training speed and higher efficiency.

-Lower memory usage.

-Better accuracy.

-Support of parallel and GPU learning.

-Capable of handling large-scale data.

Algorithms and Techniques

CatBoost Model:

CatBoost is a recently open-sourced machine learning algorithm from Yandex. It can easily integrate with deep learning frameworks like Google's TensorFlow and Apple's Core ML. It can work with diverse data types to help solve a wide range of problems that businesses face today. To top it up, it provides best-in-class accuracy.

It is especially powerful in two ways:

It yields state-of-the-art results without extensive data training typically required by other machine learning methods, and Provides powerful out-of-the-box support for the more descriptive data formats that accompany many business problems.

Algorithms and Techniques

Random Forest Classifier: ensemble learning method for classification and other tasks that operates by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees.

Decision Tree Classifier : A Decision Tree is a simple representation for classifying examples. It is a Supervised Machine Learning where the data is continuously split according to a certain parameter

K-neighbours Classifier

04

Benchmark Model

Benchmark Model

- We will use Logistic Regression model as a Benchmark in which to compare our models 's performance to , because it is fast and simple to implement.
- We will implement the roc_auc_score , Precision and Recall Metrics to Compare other Models 's Results.

C METHODOLOGY

-- 1 Data Pre-processing

-- 2 Implementation

-- 3 Refinement

A
B
C
D

01

Data Preprocessing

Data Preprocessing

Data set Preparation for Models training

A)Dividing our Combined Data to three data sets :

1-received : extracting the items with event= offer received.
2-Viewed : extracting the items with event = offer viewed.
3-completed : extracting the items with event = offer completed.
4-transaction : extracting the items with event = transaction .

B)(1st output)extracting the persons who completes the received offers ,two new columns to be added to updated data set :

-(forecast_finish) column which equals to (received offer time + offer duration) .
-(finish) column which equals to (forecast_finish) value and received time value in case of the offer not completed or equals to completion time in case of offer completed.
-(completed) column which equals to (1) in case of offer completed and equals to (0) in case of offer not completed.

C)(2nd output) extracting the person who completed the received offer (1st ouput) after viewing the offer within the offer period , three columns to be added

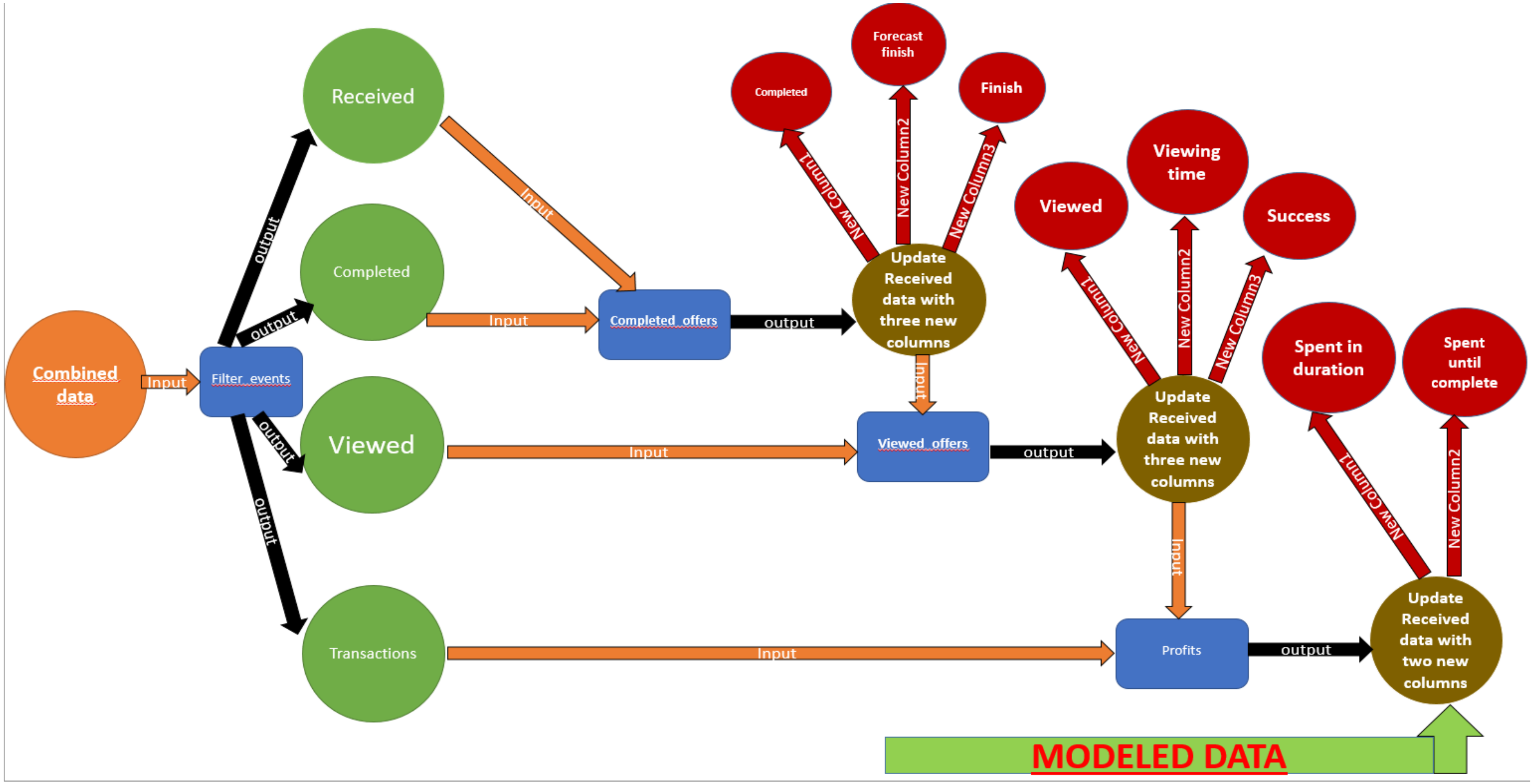
-(success) Column which equals to (1) in case of offer completed after viewing the offer other wise equals to (0).
-(viewing_time) Column which equals to viewed offer time
-(Viewed) Column which equals to either (1) or (0).

D)(3rd output) profits calculation for the amount of money which is spent within the offer forecast completion time assuming that all transaction executed within the offer duration are using the offers

Finally we will get our Modelled data which will be used in our models

Data Preprocessing

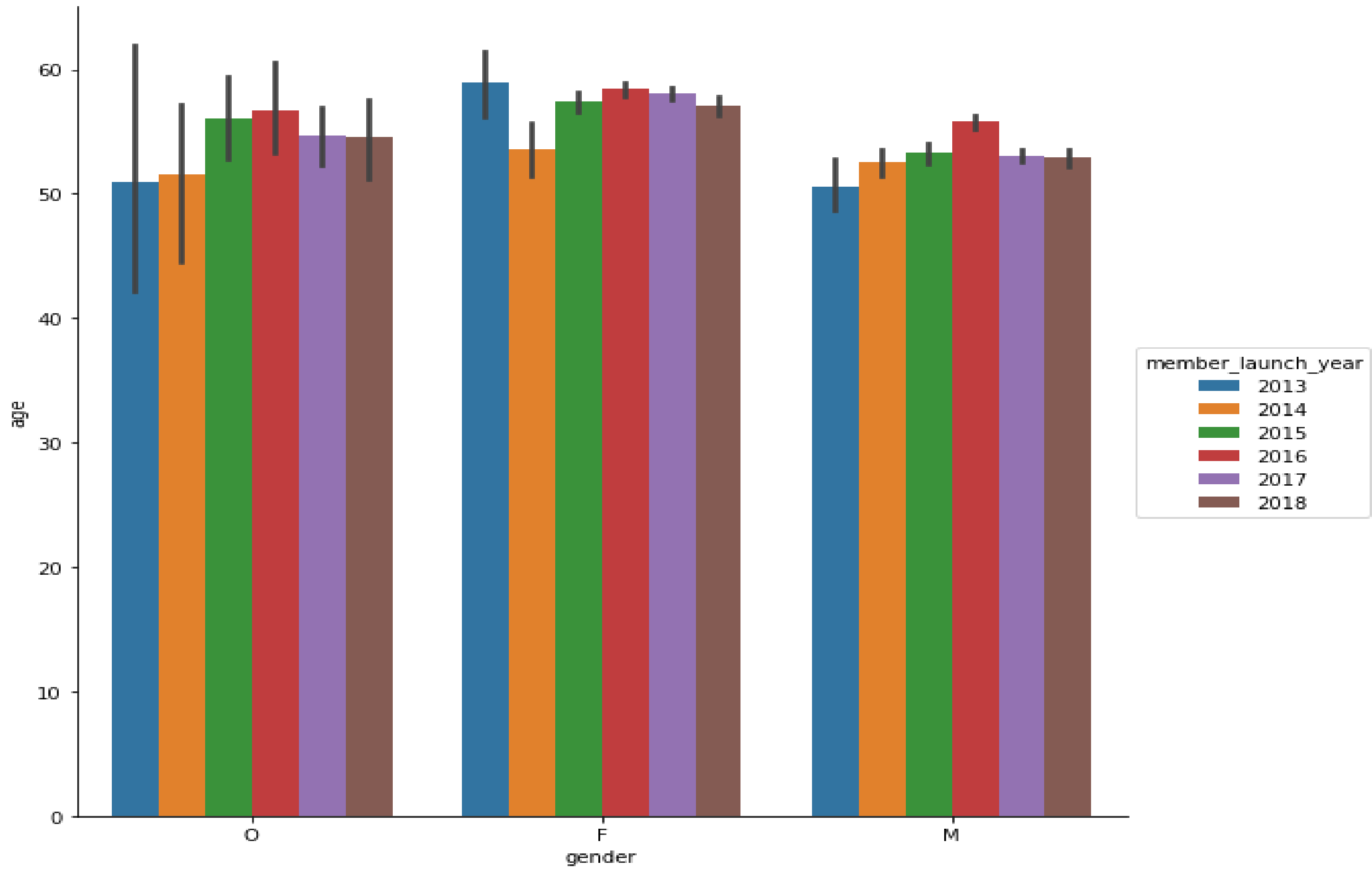
Flow Chart Describes the Pre-Processing Process to generate the Modelled Data



The Success Column (our output label) equals to "1" in case of offer Completed after receiving and viewing or " 0" otherwise

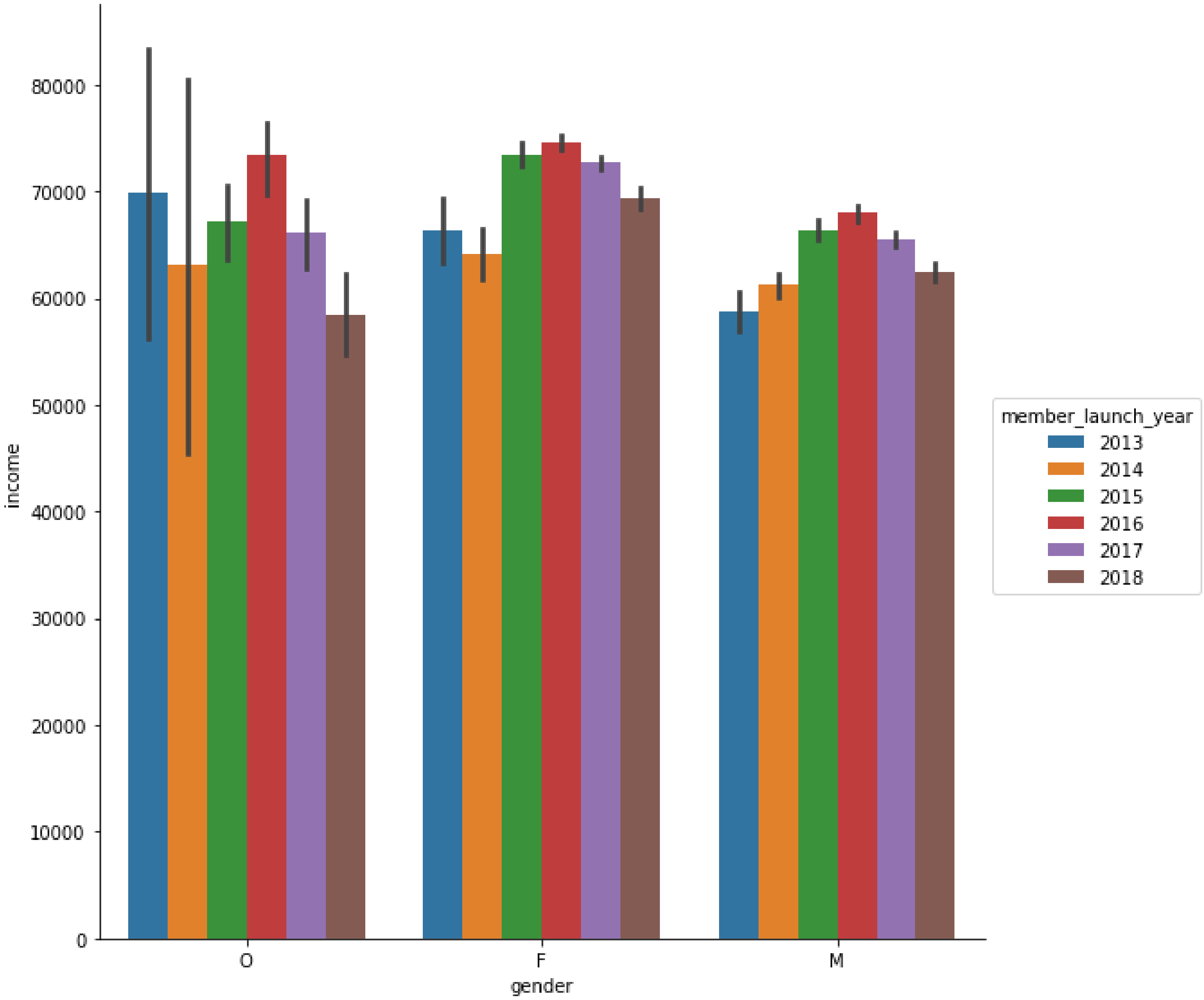
Data Preprocessing

Exploratory Visualization of Modelled data

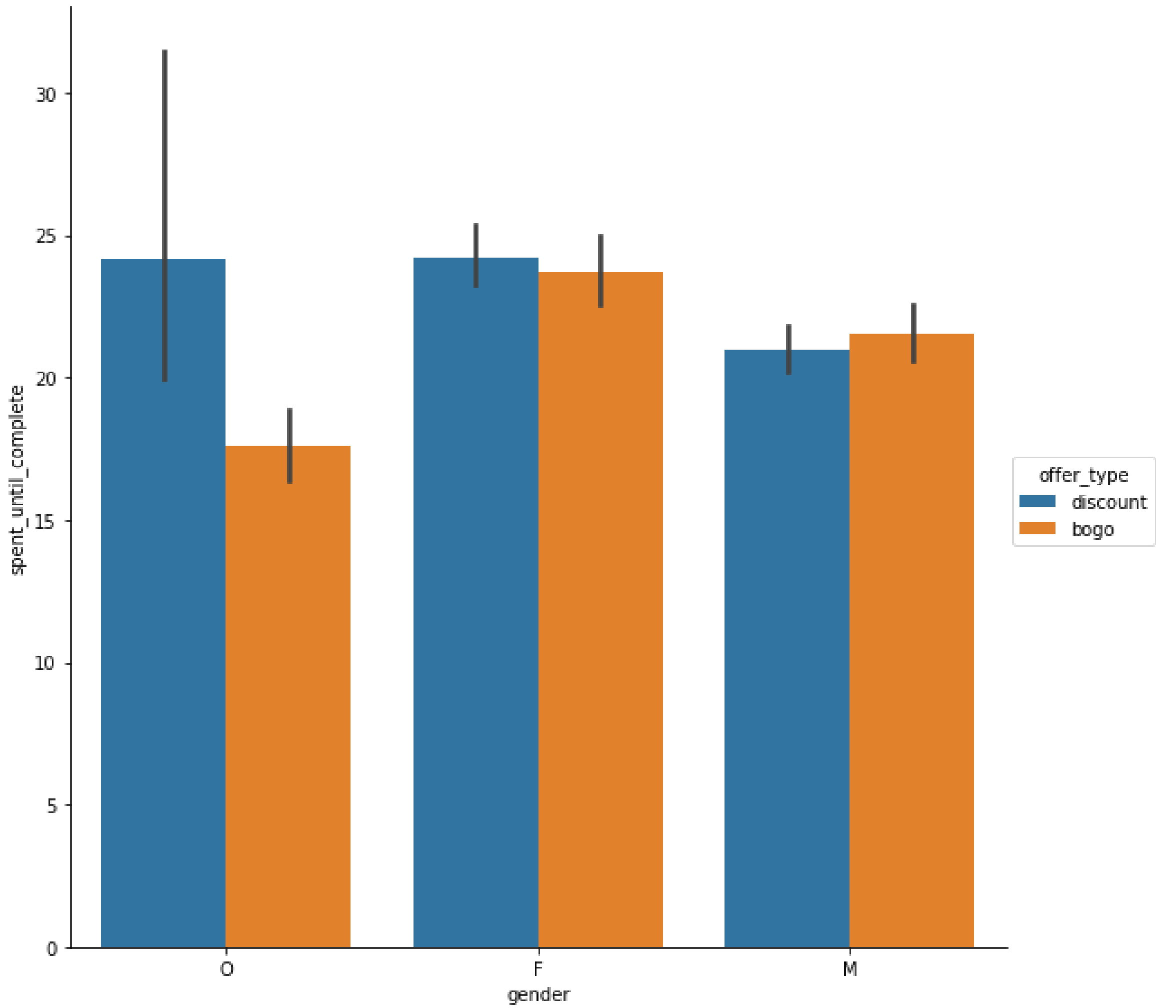


The relation between gender and age with customer member ship launching year for Successful offers only

Data Preprocessing

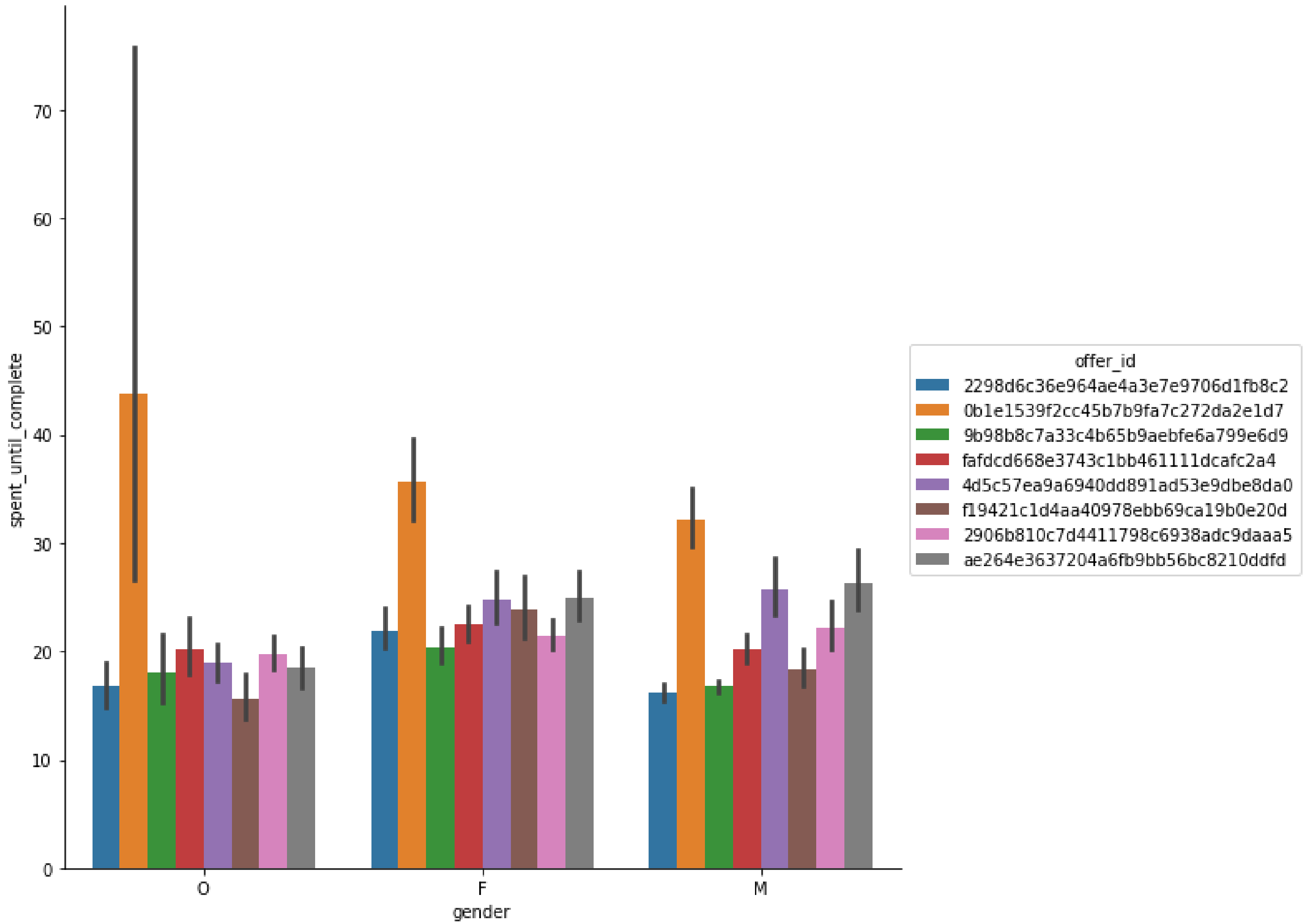


Data Preprocessing



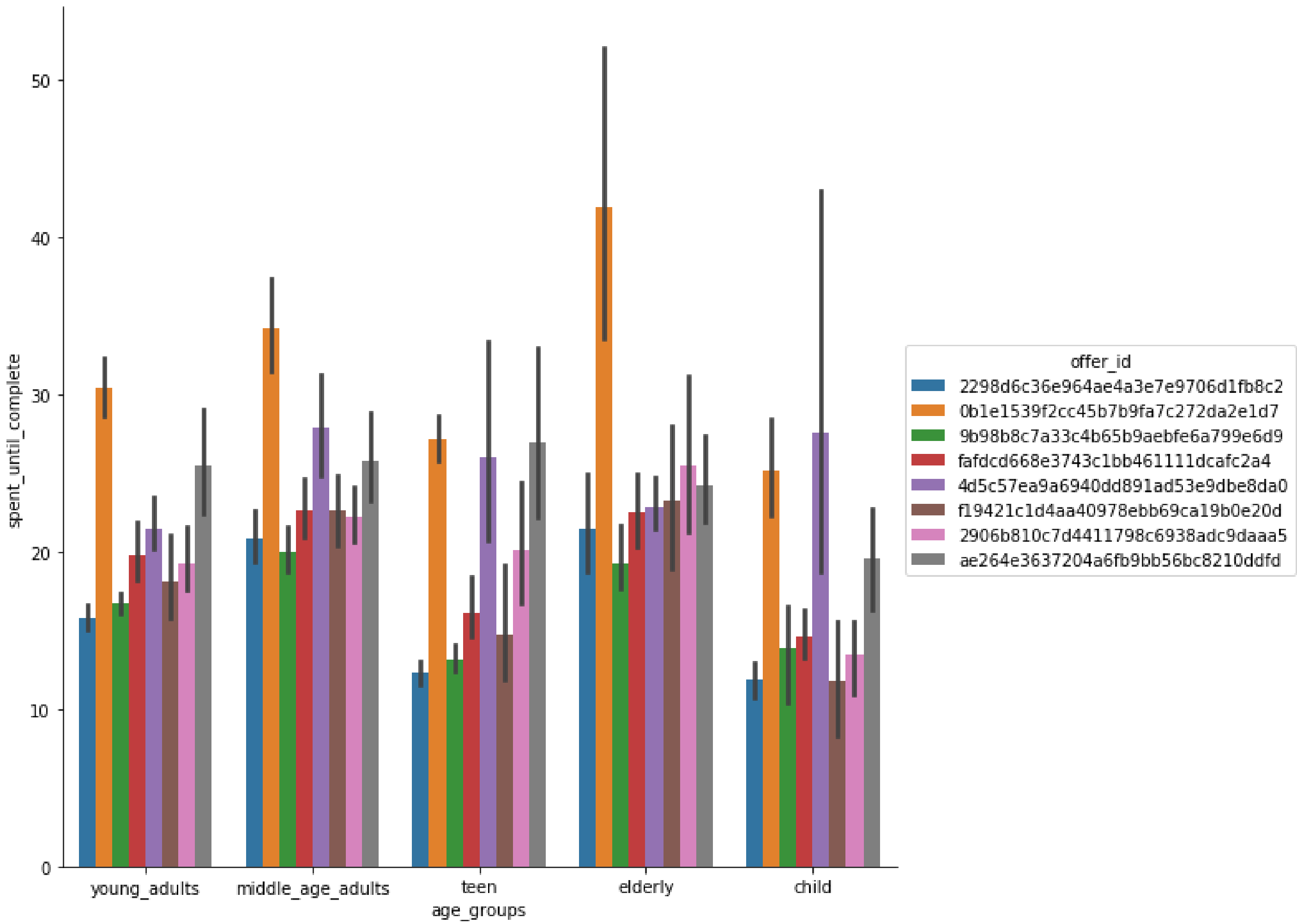
The relation between gender and Customer spent until offer Completion with offer type for Successful offers only

Data Preprocessing



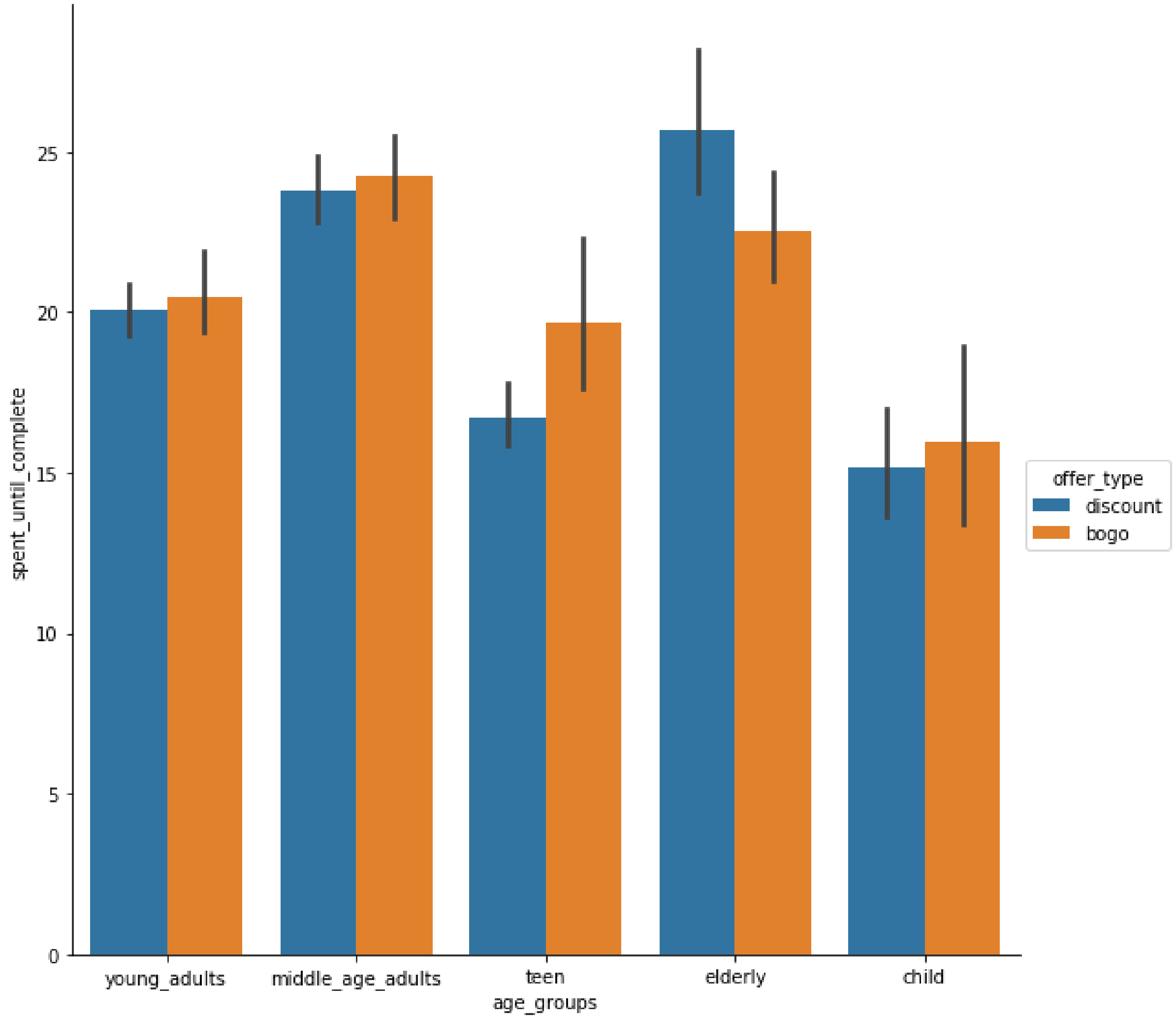
The relation between gender and Customer spent until offer Completion with offer id for Successful offers only

Data Preprocessing



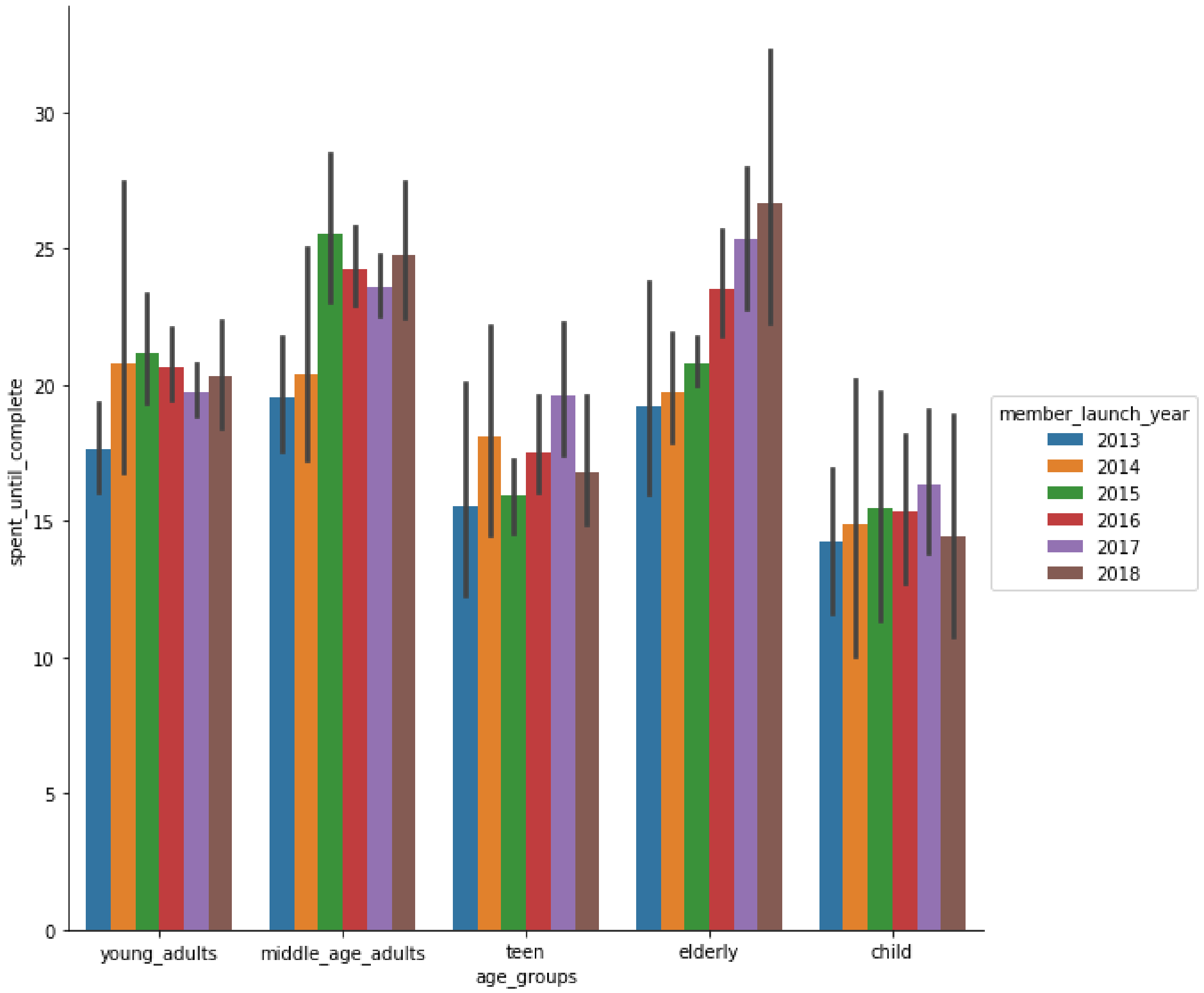
The relation between age_groups and Customer spent until offer Completion with offer id for Successful offers only

Data Preprocessing



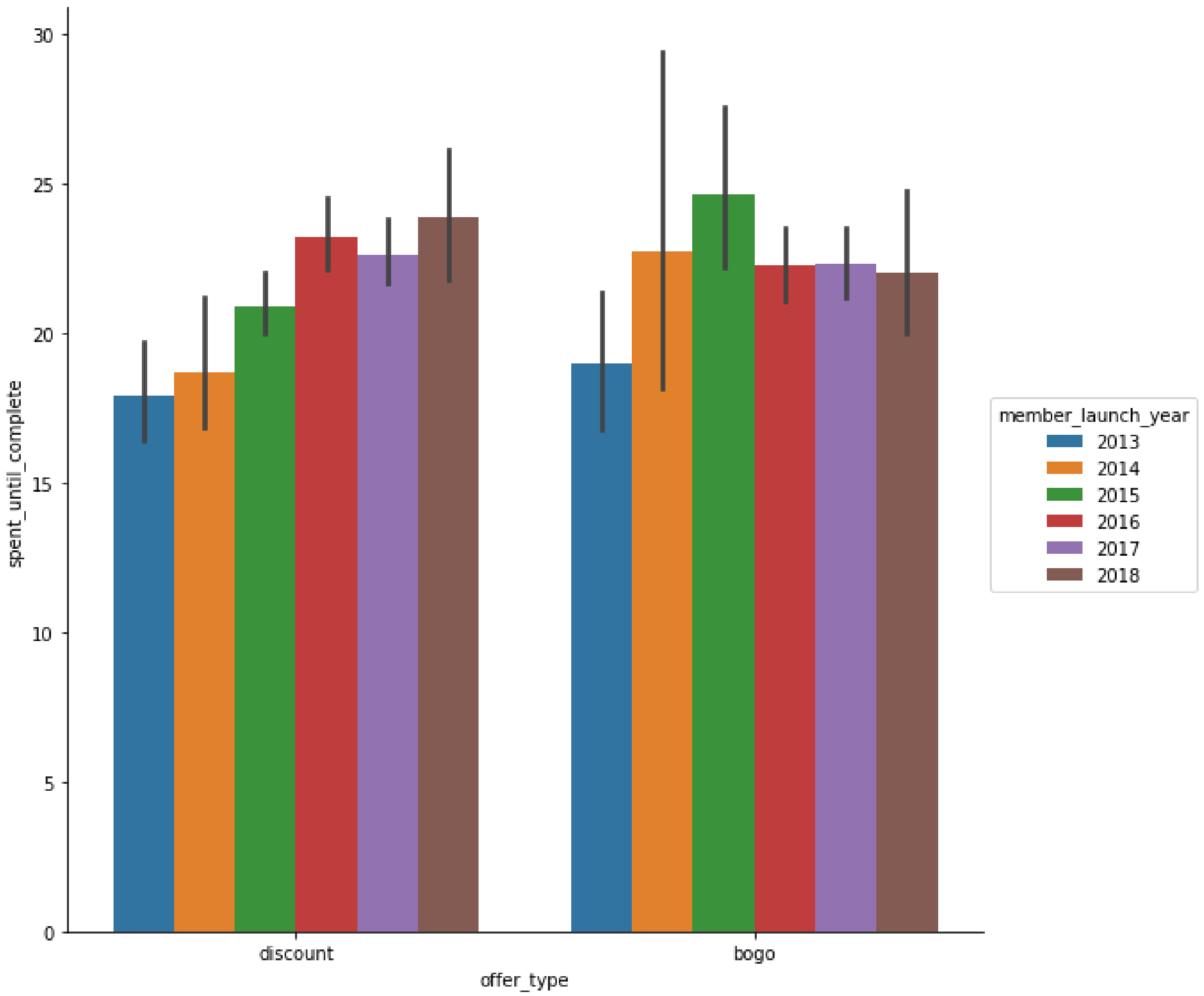
The relation between age_groups and Customer spent until offer Completion with offer type for Successful offers only

Data Preprocessing



The relation between age_groups and Customer spent until offer Completion with member launch year for Successful offers only

Data Preprocessing



The relation between offer type and Customer spent until offer Completion with member launch year for Successful offers only

Data Preprocessing

Modelled Data statistics Explanation

For Females:
Number of offer Succeeded: 11107 , 40.5% of Female received offers.
Number of offer Succeeded: 11107 , 16.7% of Total received offers.

For Males:
Number of offer Succeeded: 12413 , 32.6% of Male received offers.
Number of offer Succeeded: 12413 , 18.7% of Total received offers.

For Females:
Total Spent until offer Complete: 266395.9199999997 , 53.8% of Total Female received offers.

For Males:
Total Spent until offer Complete: 263321.7999999994 , 50.6% of Total Male received offers.

For Females:
Total Spent until offer Complete: 266395.9199999997 , 25.9% of Total received offers.

For Males:
Total Spent until offer Complete: 263321.7999999994 , 25.6% of Total received offers.

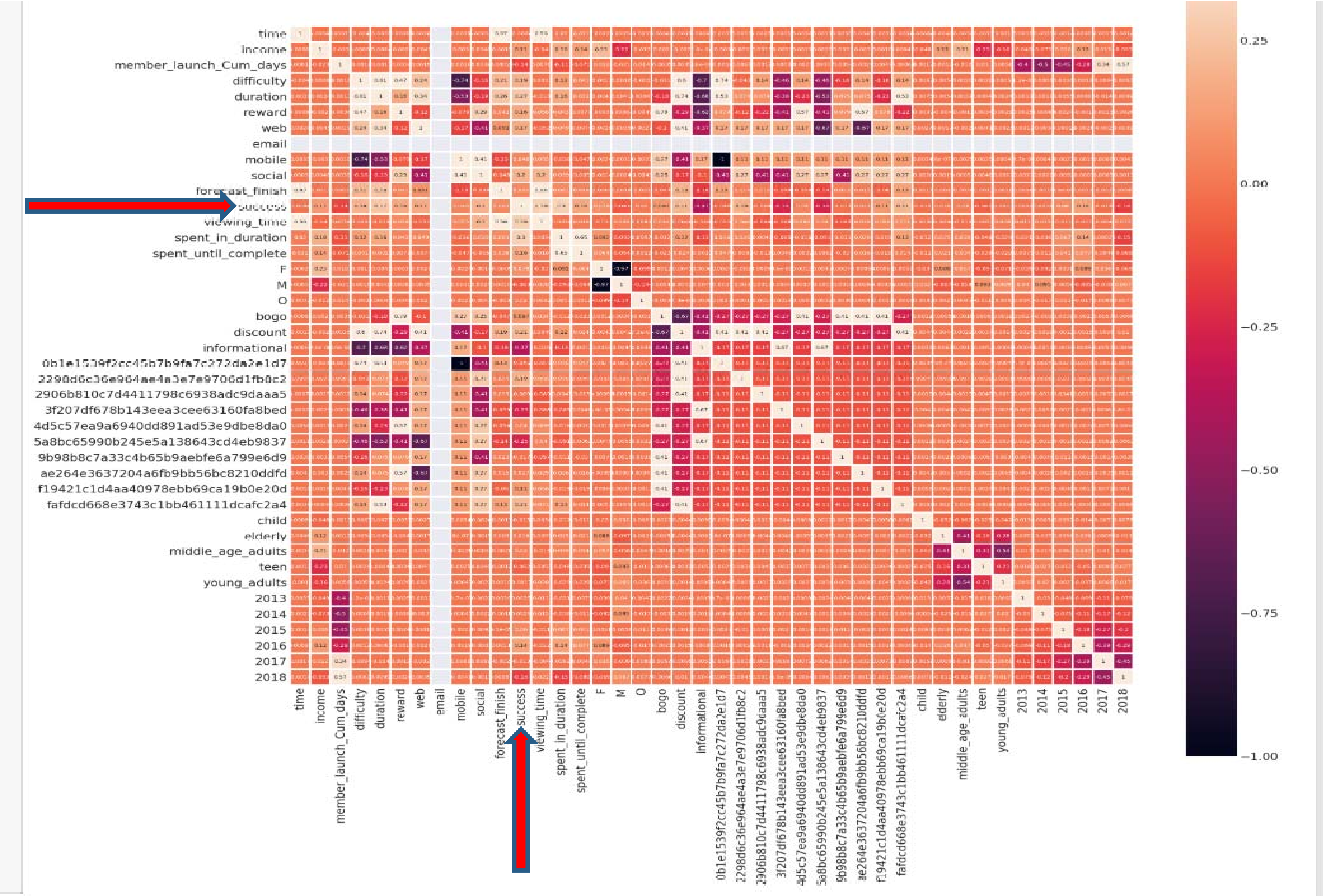
Data Preprocessing

TOP-10 Customers
[#1]
Person: 0cc6e8553c844c02ab525bc466aa569b
Number of Success Offers: 4 offers
Spent until complete: $1754.0
[#2]
Person: 2fc5fa0b50f944e398b903b0be851678
Number of Success Offers: 3 offers
Spent until complete: $1532.0
[#3]
Person: 8d31a8a4b5d24b10a54da118855f7132
Number of Success Offers: 4 offers
Spent until complete: $1489.0
[#4]
Person: a2633655a62e4287a3b651d926a774a6
Number of Success Offers: 6 offers
Spent until complete: $1439.0
[#5]
Person: e72ad19d4f6c4827b69b55c5e3a55bba
Number of Success Offers: 4 offers
Spent until complete: $1271.0
[#6]
Person: 4d4216b868fe43ddb9c9f0b77212c0cb
Number of Success Offers: 6 offers
Spent until complete: $1164.0
[#7]
Person: bfce6d50205a4f6982d87ce80e5d5356
Number of Success Offers: 4 offers
Spent until complete: $1161.0
[#8]
Person: dce784e26f294101999d000fad9089bb
Number of Success Offers: 4 offers
Spent until complete: $1074.0
[#9]
Person: 5dfdad4241764dfe959f51b7460e42b1
Number of Success Offers: 4 offers
Spent until complete: $1029.0
[#10]
Person: 454b00bdd77c4f588eb9f6cafd81dc5d
Number of Success Offers: 1 offers
Spent until complete: $1016.0

Top-10 customers Whom they have successful offers and the amount they have spent until offer completion

Data Preprocessing

Heat map for the Modelled data



Data Preprocessing

Best Features for the Modelled data

getting the best features for the modelled data

```
#Correlation with output variable
cor_target = abs(C_mat["success"])
#Selecting highly correlated features
relevant_features = cor_target[cor_target>0.15].sort_values()
relevant_features
```

```
spent_until_complete                0.155922
2018                                0.158226
reward                              0.163495
web                                 0.167122
2298d6c36e964ae4a3e7e9706d1fb8c2    0.186246
difficulty                          0.190465
social                              0.197064
discount                            0.209410
fafdcd668e3743c1bb461111dcafc2a4    0.210728
5a8bc65990b245e5a138643cd4eb9837    0.249718
3f207df678b143eea3cee63160fa8bed    0.250011
duration                            0.265026
viewing_time                        0.291399
spent_in_duration                   0.299669
informational                       0.374796
success                             1.000000
```

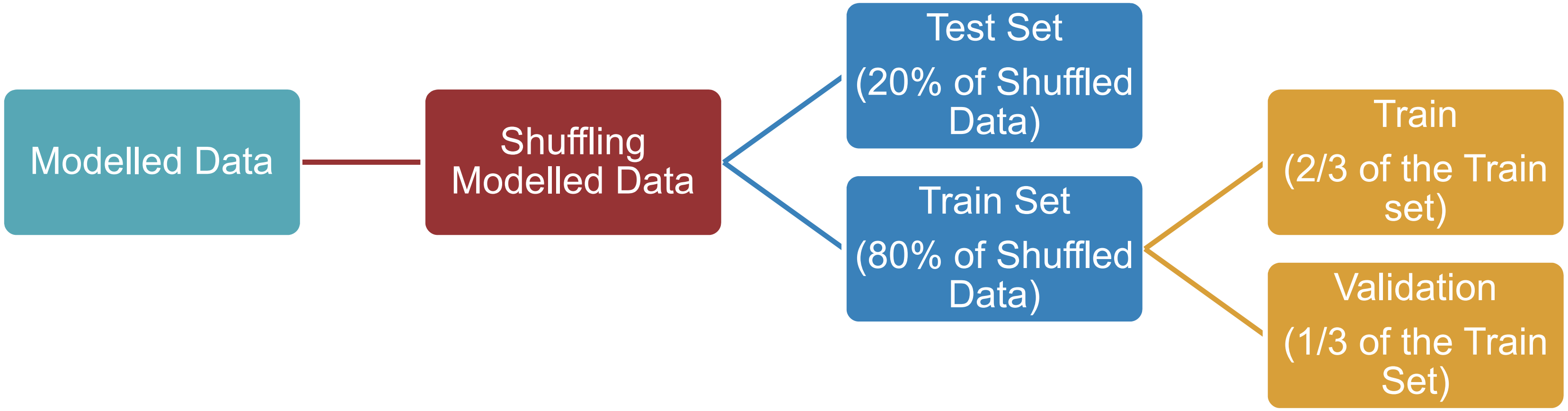
Data Preprocessing

The Modelled data which will be utilized in our Model training and testing :shape (66501 rows x 42 columns)

- Input features : 41 Features
- Output Label: 1 Column ("success") It will be either("1") or ("0").
- We will follow the below Process for Dividing the Modelled Data to training , validation and testing Sets.

```
modeled_data.info(0)

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 66501 entries, 0 to 66500
Data columns (total 42 columns):
time                                 66501 non-null float64
income                               66501 non-null float64
member_launch_Cum_days               66501 non-null float64
difficulty                           66501 non-null float64
duration                             66501 non-null float64
reward                               66501 non-null float64
web                                  66501 non-null float64
email                                66501 non-null float64
mobile                               66501 non-null float64
social                               66501 non-null float64
forecast_finish                      66501 non-null float64
success                              66501 non-null int64
viewing_time                         66501 non-null float64
spent_in_duration                    66501 non-null float64
spent_until_complete                 66501 non-null float64
F                                    66501 non-null uint8
M                                    66501 non-null uint8
O                                    66501 non-null uint8
bogo                                 66501 non-null uint8
discount                             66501 non-null uint8
informational                        66501 non-null uint8
0b1e1539f2cc45b7b9fa7c272da2e1d7     66501 non-null uint8
2298d6c36e964ae4a3e7e9706d1fb8c2     66501 non-null uint8
2906b810c7d4411798c6938adc9daaa5     66501 non-null uint8
3f207df678b143eea3cee63160fa8bed     66501 non-null uint8
4d5c57ea9a6940dd891ad53e9dbe8da0     66501 non-null uint8
5a8bc65990b245e5a138643cd4eb9837     66501 non-null uint8
9b98b8c7a33c4b65b9aebfe6a799e6d9     66501 non-null uint8
ae264e3637204a6fb9bb56bc8210ddfd     66501 non-null uint8
f19421c1d4aa40978ebb69ca19b0e20d     66501 non-null uint8
fafdcd668e3743c1bb461111dcafc2a4     66501 non-null uint8
child                                66501 non-null uint8
elderly                              66501 non-null uint8
middle_age_adults                    66501 non-null uint8
teen                                 66501 non-null uint8
young_adults                         66501 non-null uint8
2013                                 66501 non-null uint8
2014                                 66501 non-null uint8
2015                                 66501 non-null uint8
2016                                 66501 non-null uint8
2017                                 66501 non-null uint8
2018                                 66501 non-null uint8
dtypes: float64(14), int64(1), uint8(27)
memory usage: 9.3 MB
```

Data Preprocessing



02

Implementation

Implementation

Firstly - after the Preparation of our training and testing data sets -We Will implement our Benchmark model (Logistic regression Model) and calculating our Metrics that we have discussed before.

LOGISTIC REGRESSION MODEL(BENCHMARK MODEL)

In [334]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
from sklearn.metrics import confusion_matrix, classification_report
from sklearn.metrics import precision_score,recall_score,f1_score
from sklearn.metrics import roc_auc_score
classifier = LogisticRegression(random_state = 0)
classifier.fit(X_train, y_train)

# Predicting the Test set results
y_pred_LOG = classifier.predict(X_test)

print( 'roc_auc_score:' , roc_auc_score(y_test, y_pred_LOG))
print('Precision Metric:',precision_score(y_test, y_pred_LOG))
print('Recall Metric:',recall_score(y_test, y_pred_LOG))
```

```
/home/ec2-user/anaconda3/envs/mxnet_p36/lib/python3.6/site-packages/sk
t solver will be changed to 'lbfgs' in 0.22. Specify a solver to silen
  FutureWarning)
```

```
roc_auc_score: 0.855732621614278
Precision Metric: 0.8207803046108909
Recall Metric: 0.8129778879933871
```

Roc_auc_score:0.856
Precision: 0.821
Recall: 0.813

Implementation

Secondly , we will follow our Models training and testing :

Random Forrest Classifier:

```
from sklearn.ensemble import RandomForestClassifier
import math
rf = RandomForestClassifier(max_depth=10, random_state=0)

rf.fit(X_train, y_train)
y_pred_RF = rf.predict(X_test)
print( 'roc_auc_score:' , roc_auc_score(y_test, np.around(y_pred_RF)))
print('Precision Metric:',precision_score(y_test, np.around(y_pred_RF)))
print('Recall Metric:',recall_score(y_test, np.around(y_pred_RF)))

#print(accuracy_score(y_test, np.around(y_pred_RF)))
#print(confusion_matrix(y_test, np.around(y_pred_RF)))
#print('-'*100)
#print(classification_report(y_test, np.around(y_pred_RF)))
```

```
/home/ec2-user/anaconda3/envs/mxnet_p36/lib/python3.6/site-packages/sklearn/
value of n_estimators will change from 10 in version 0.20 to 100 in 0.22.
  "10 in version 0.20 to 100 in 0.22.", FutureWarning)
```

```
roc_auc_score: 0.9702271326258832
Precision Metric: 0.9338452451269935
Recall Metric: 0.9801611903285803
```

Roc_auc_score:0.970
Precision: 0.933
Recall: 0.980

Implementation

Decision Tree Classifier:

```
from sklearn.tree import DecisionTreeClassifier

dt = DecisionTreeClassifier()

dt.fit(X_train, y_train)
y_pred_tree = dt.predict(X_test)
print( 'roc_auc_score:' , roc_auc_score(y_test, y_pred_tree))
print('Precision Metric:',precision_score(y_test, y_pred_tree))
print('Recall Metric:',recall_score(y_test, y_pred_tree))

#print(accuracy_score(y_test, y_pred_tree))
#print(confusion_matrix(y_test, y_pred_tree))
#print('-'*100)
#print(classification_report(y_test, y_pred_tree))
```

```
roc_auc_score: 0.962364001246666
Precision Metric: 0.9518694484610618
Recall Metric: 0.9522628642281463
```

Roc_auc_score:0.962
Precision: 0.952
Recall: 0.952

Implementation

K-neighbors Classifier:

```
from sklearn.neighbors import KNeighborsClassifier

knn = KNeighborsClassifier()

knn.fit(X_train, y_train)
# Predicting the Test set results
y_pred_knn = knn.predict(X_test)

print( 'roc_auc_score:' , roc_auc_score(y_test, y_pred_knn))
print('Precision Metric:',precision_score(y_test, y_pred_knn))
print('Recall Metric:',recall_score(y_test, y_pred_knn))

#print(accuracy_score(y_test, y_pred_knn))
#print(confusion_matrix(y_test, y_pred_knn))
#print('-'*100)
#print(classification_report(y_test, y_pred_knn))
```

```
roc_auc_score: 0.7806063590805208
Precision Metric: 0.718865598027127
Recall Metric: 0.7228766274023558
```

Roc_auc_score:0.780
Precision: 0.719
Recall: 0.723

Implementation

Amazon Sage maker XG-Boost built in Algorithm :

```
# As stated above, we use this utility method to construct the image name for the training container.
container = get_image_uri(session.boto_region_name, 'xgboost','0.90-1')

xgb = sagemaker.estimator.Estimator(container, # The location of the container we wish to use
                                    role,                                    # What is our current IAM Role
                                    train_instance_count=1,                  # How many compute instances
                                    train_instance_type='ml.m4.xlarge',      # What kind of compute instances
                                    output_path='s3://{}/{}/output'.format(session.default_bucket(), prefix),
                                    sagemaker_session=session)

# And then set the algorithm specific parameters.
xgb.set_hyperparameters(max_depth=2,
                        eta=0.02,
                        gamma=2.6,
                        min_child_weight=2,
                        subsample=0.65,
                        silent=0,
                    #   alpha=1.5,
                     #  colsample_bylevel=0.5,
                      #  colsample_bynode=0.5,
                       # colsample_bytree=0.5,
                        max_delta_step=3,
                        objective='binary:logistic',
                        early_stopping_rounds=100,
                        num_round=500)
```

```
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
from sklearn.metrics import confusion_matrix, classification_report
from sklearn.metrics import precision_score,recall_score,f1_score
from sklearn.metrics import roc_auc_score,roc_curve,auc
false_positive_rate, true_positive_rate, thresholds = roc_curve(y_test, predictions)
print( 'roc_auc Metric:' , roc_auc_score(y_test, predictions))
print( 'AUC Metric:' , auc(false_positive_rate, true_positive_rate))
print('Precision Metric:',precision_score(y_test, predictions))
print('Recall Metric:',recall_score(y_test, predictions))
#print(accuracy_score(y_test, predictions))
#print(confusion_matrix(y_test, predictions))
#print('-'*100)
#print(classification_report(y_test, predictions))
```

```
roc_auc Metric: 0.956562528252559
AUC Metric: 0.956562528252559
Precision Metric: 0.9495728276724318
Recall Metric: 0.9417234965902046
```

Roc_auc_score:0.957
Precision: 0.950
Recall: 0.942

Implementation

LightGBM Model :

```
from sklearn.model_selection import train_test_split, GridSearchCV
import lightgbm as lgb
train_data=lgb.Dataset(X_train, label=y_train)

#Select Hyper-Parameters
params = {'boosting_type': 'gbdt',
          'max_depth' : -1,
          'objective': 'binary',
          'nthread': 5,
          'num_leaves': 64,
          'learning_rate': 0.07,
          'max_bin': 512,
          'subsample_for_bin': 200,
          'subsample': 1,
          'subsample_freq': 1,
          'colsample_bytree': 0.8,
          'reg_alpha': 1.2,
          'reg_lambda': 1.2,
          'min_split_gain': 0.5,
          'min_child_weight': 1,
          'min_child_samples': 5,
          'scale_pos_weight': 1,
          'num_class' : 1,
          'metric' : 'auc'
          }
```

```
print('Fitting with params: ')
print(params)

#Train model on selected parameters and number of iterations
lgbm = lgb.train(params,
                 train_data,
                 280,
                 #early_stopping_rounds= 40,
                 verbose_eval= 4
                 )

#Predict on test set
predictions_lgbm_prob = lgbm.predict(X_test)
predictions_lgbm_01 = np.where(predictions_lgbm_prob > 0.5, 1, 0) #Turn probability to 0-1 binary output
```

Implementation

LightGBM Model :

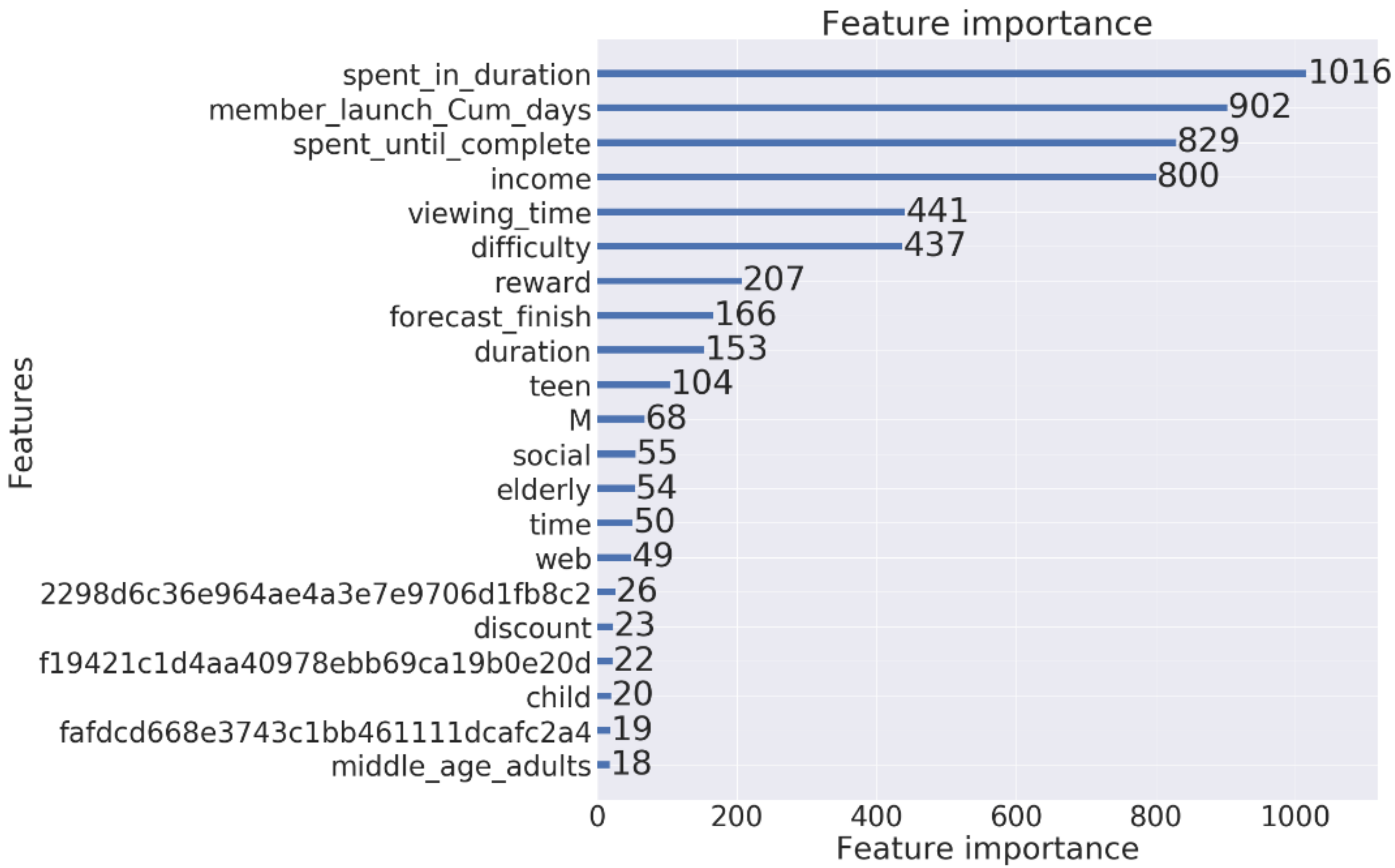
```
from sklearn.metrics import confusion_matrix,accuracy_score, roc_curve, auc
plt.figure(figsize=(16, 10))
lgb.plot_importance(lgbm, max_num_features=21, importance_type='split')

#Print accuracy
auc_lgbm = roc_auc_score(y_test,predictions_lgbm_01)
print('roc_auc_score of Light GBM model:', auc_lgbm)
print('Precision Metric:',precision_score(y_test, predictions_lgbm_01))
print('Recall Metric:',recall_score(y_test, predictions_lgbm_01))
#Print Area Under Curve
plt.figure(figsize=(16, 10))
false_positive_rate, recall, thresholds = roc_curve(y_test, predictions_lgbm_prob)
roc_auc = auc(false_positive_rate, recall)
plt.title('Receiver Operating Characteristic (ROC)')
plt.plot(false_positive_rate, recall, 'b', label = 'AUC = %0.3f' %roc_auc)
plt.legend(loc='lower right')
plt.plot([0,1], [0,1], 'r--')
plt.xlim([0.0,1.0])
plt.ylim([0.0,1.0])
plt.ylabel('Recall')
plt.xlabel('Fall-out (1-Specificity)')
plt.show()

#roc_auc_score(y_test, predictions)

#print('AUC score:', roc_auc)
print('roc_auc_score:', roc_auc_score(y_test, predictions_lgbm_01))

#Print Confusion Matrix
plt.figure(figsize=(16, 10))
cm = confusion_matrix(y_test, predictions_lgbm_01)
labels = ['No Default', 'Default']
plt.figure(figsize=(8,6))
sns.heatmap(cm, xticklabels = labels, yticklabels = labels, annot = True, fmt='d', cmap="Blues", vmin = 0.2);
plt.title('Confusion Matrix')
plt.ylabel('True Class')
plt.xlabel('Predicted Class')
plt.show()
```

```
roc_auc_score of Light GBM model: 0.9736142527264956
Precision Metric: 0.9632065775950668
Recall Metric: 0.9683818970861748
```

Roc_auc_score:0.974
Precision: 0.963
Recall: 0.968

Implementation

LightGBM Model(Features importance) :



Implementation

CatBoost Model :

```
from catboost import CatBoostClassifier
model_cat = CatBoostClassifier(iterations=4000, learning_rate=0.005, l2_leaf_reg=5, depth=4, rsm=0.98, loss_function= 'Logloss',
model_cat.fit(X_train,y_train,eval_set=(X_val,Y_val))
preds = model_cat.predict_proba(X_test)
pred = np.where(preds > 0.5, 1, 0) #Turn probability to 0-1 binary output
predss= pred[:,1]
```

```
#Print accuracy
#CatBoostClassifier.plot_importance(model_cat, max_num_features=21, importance_type='split')
auc_cat = roc_auc_score(y_test,preds_cat)
print('roc_auc_score of CATBOOST model:', auc_cat)
print('Precision Metric:',precision_score(y_test, preds_cat))
print('Recall Metric:',recall_score(y_test, preds_cat))

#Print Area Under Curve
plt.figure(figsize=(16, 10))
false_positive_rate, recall, thresholds = roc_curve(y_test, preds_cat)
roc_auc = auc(false_positive_rate, recall)
plt.title('Receiver Operating Characteristic (ROC)')
plt.plot(false_positive_rate, recall, 'b', label = 'AUC = %0.3f' %roc_auc)
plt.legend(loc='lower right')
plt.plot([0,1], [0,1], 'r--')
plt.xlim([0.0,1.0])
plt.ylim([0.0,1.0])
plt.ylabel('Recall')
plt.xlabel('Fall-out (1-Specificity)')
plt.show()

#print('AUC score:', roc_auc)
print('roc_auc_score:', roc_auc_score(y_test,preds_cat))
#Print Confusion Matrix
plt.figure(figsize=(16, 10))
cm = confusion_matrix(y_test, preds_cat)
labels = ['No Default', 'Default']
plt.figure(figsize=(8,6))
sns.heatmap(cm, xticklabels = labels, yticklabels = labels, annot = True, fmt='d', cmap="Blues", vmin = 0.2);
plt.title('Confusion Matrix')
plt.ylabel('True Class')
plt.xlabel('Predicted Class')
plt.show()
```

```
roc_auc_score of CATBOOST model: 0.974086648947443
Precision Metric: 0.9634346754313886
Recall Metric: 0.9692085141558173
```

Roc_auc_score:0.974
Precision: 0.963
Recall: 0.969

03

Refinement

Refinement

We will work for improvement of our Models , and we will concentrate on XGB , LGB and CatBoost by tuning the hyper parameters.

Amazon Sage maker XG-Boost built in Algorithm-Hyper parameter Tuning :

```python
#Hyper parametrer tuning
# First, make sure to import the relevant objects used to construct the tuner
from sagemaker.tuner import IntegerParameter, ContinuousParameter, HyperparameterTuner


xgb_hyperparameter_tuner = HyperparameterTuner(estimator = xgb, # The estimator object to use as the basis for the training jobs
                                               objective_metric_name = 'validation:auc', # The metric used to compare trained mo
                                               objective_type = 'Maximize', # Whether we wish to minimize or maximize the metric
                                               max_jobs = 6, # The total number of models to train
                                               max_parallel_jobs = 3, # The number of models to train in parallel
                                               hyperparameter_ranges = {
                                                    'max_depth': IntegerParameter(2, 4),

                                                    'eta'      : ContinuousParameter(0.02, 0.04),
                                                  #  'colsample_bylevel'      : ContinuousParameter(0.1, 1),
                                                   # 'colsample_bynode'      : ContinuousParameter(0.1, 1),
                                                    #'colsample_bytree'      : ContinuousParameter(0.1, 1),
                                                    #'max_delta_step': IntegerParameter(1, 3),

                                                    #'alpha'      : ContinuousParameter(1.1,1.6),
                                                    'min_child_weight': IntegerParameter(1, 2),
                                                    'num_round': IntegerParameter(400, 700),
                                                    'subsample': ContinuousParameter(0.71, 0.73),
                                                    'gamma': ContinuousParameter(3.60, 3.65),
                                               })
```

Best Parameters

Best training job hyperparameters

Name	Type	Value
_tuning_objective_metric	FreeText	validation:auc
early_stopping_rounds	FreeText	100
eta	Continuous	0.03969777649103867
gamma	Continuous	3.6167721097638235
max_delta_step	FreeText	3
max_depth	Integer	3
min_child_weight	Integer	1
num_round	Integer	630
objective	FreeText	binary:logistic
silent	FreeText	0
subsample	Continuous	0.7124805853162347

Refinement

Amazon Sage maker XG-Boost built in Algorithm- (metrics with best parameters):

```
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
from sklearn.metrics import confusion_matrix, classification_report
from sklearn.metrics import precision_score,recall_score,f1_score
from sklearn.metrics import roc_auc_score,roc_curve,auc
false_positive_rate, true_positive_rate, thresholds = roc_curve(y_test, predictions)
print( 'roc_auc Metric:' , roc_auc_score(y_test, predictions))
print( 'AUC Metric:' , auc(false_positive_rate, true_positive_rate))
print('Precision Metric:',precision_score(y_test, predictions))
print('Recall Metric:',recall_score(y_test, predictions))
#print(accuracy_score(y_test, predictions))
#print(confusion_matrix(y_test, predictions))
#print('-'*100)
#print(classification_report(y_test, predictions))
```

```
roc_auc Metric: 0.9746620670340336
AUC Metric: 0.9746620670340336
Precision Metric: 0.962318246979316
Recall Metric: 0.9710684025625129
```

Roc_auc_score:0.974
Precision: 0.962
Recall: 0.971

Better results than before

UDACITY

Refinement

LightGBM Model -Hyper parameter Tuning :

```python
# Create parameters to search
gridParams = {
    'learning_rate': [0.01,0.05,0.1,0.15],
    'n_estimators': [16,32,48,52],
    'num_leaves': [15,30, 35,45,50, 60],
    'boosting_type' : ['gbdt'],
    'objective' : ['binary'],
    'random_state' : [501],
    'colsample_bytree' : [0.50,0.55,0.60,0.65, 0.70,0.75],
    'subsample' : [0.4,0.5,0.6,0.7]
   # 'metric' : 'auc'
    #'reg_alpha' : [1, 1.2],
    #'reg_lambda' : [ 1.2, 1.4],
    }

# Create classifier to use
mdl = lgb.LGBMClassifier(boosting_type= 'gbdt',
          objective = 'binary',
          n_jobs = 5,
          silent = True,
          metric='auc',
          max_depth = params['max_depth'],
          max_bin = params['max_bin'],
          subsample_for_bin = params['subsample_for_bin'],
          subsample = params['subsample'],
          subsample_freq = params['subsample_freq'],
          min_split_gain = params['min_split_gain'],
          min_child_weight = params['min_child_weight'],
          min_child_samples = params['min_child_samples'],
          scale_pos_weight = params['scale_pos_weight'])

# View the default model params:
mdl.get_params().keys()

# Create the grid
grid = GridSearchCV(mdl, gridParams, verbose=2, cv=4, n_jobs=-1)

# Run the grid
grid.fit(X_train, y_train)

# Print the best parameters found
print(grid.best_params_)
print(grid.best_score_)
```

Best Parameters

```
{'boosting_type': 'gbdt', 'colsample_bytree': 0.75, 'learning_rate': 0.1, 'n_estimators': 48, 'num_leaves': 35, 'objective': 'b
inary', 'random_state': 501, 'subsample': 0.5}
0.9741050387161935
```

Refinement

LightGBM Model -Hyper parameter Tuning (metrics with best parameters):

```python
from sklearn.metrics import confusion_matrix,accuracy_score, roc_curve, auc

lgb.plot_importance(lgbm, max_num_features=21, importance_type='split',figsize= (20,20))

#Print accuracy
auc_lgbm = roc_auc_score(y_test,predictions_lgbm_01)
print('roc_auc_score of Light GBM model:', auc_lgbm)
print('Precision Metric:',precision_score(y_test, predictions_lgbm_01))
print('Recall Metric:',recall_score(y_test, predictions_lgbm_01))
#Print Area Under Curve
plt.figure(figsize=(16, 10))
false_positive_rate, recall, thresholds = roc_curve(y_test, predictions_lgbm_prob)
roc_auc = auc(false_positive_rate, recall)
plt.title('Receiver Operating Characteristic (ROC)')
plt.plot(false_positive_rate, recall, 'b', label = 'AUC = %0.3f' %roc_auc)
plt.legend(loc='lower right')
plt.plot([0,1], [0,1], 'r--')
plt.xlim([0.0,1.0])
plt.ylim([0.0,1.0])
plt.ylabel('Recall')
plt.xlabel('Fall-out (1-Specificity)')
plt.show()

#roc_auc_score(y_test, predictions)

#print('AUC score:', roc_auc)
print('roc_auc_score:', roc_auc_score(y_test, predictions_lgbm_01))

#Print Confusion Matrix
plt.figure(figsize=(16, 10))
cm = confusion_matrix(y_test, predictions_lgbm_01)
labels = ['No Default', 'Default']
plt.figure(figsize=(8,6))
sns.heatmap(cm, xticklabels = labels, yticklabels = labels, annot = True, fmt='d', cmap="Blues", vmin = 0.2);
plt.title('Confusion Matrix')
plt.ylabel('True Class')
plt.xlabel('Predicted Class')
plt.show()
```

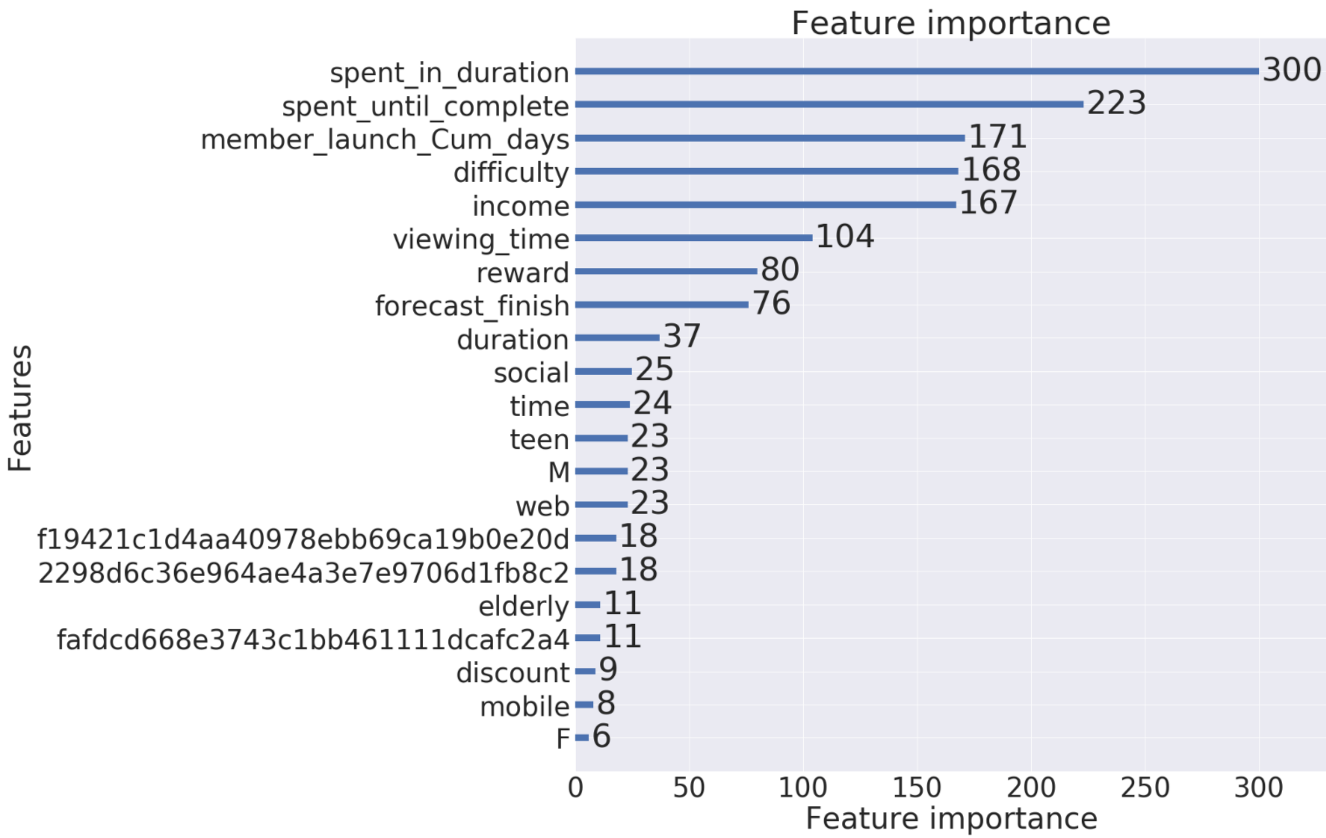
```
roc_auc_score of Light GBM model: 0.9746763047364562
Precision Metric: 0.960016319869441
Recall Metric: 0.9725149824343873
```

Better results than before

Roc_auc_score:0.974
Precision: 0.960
Recall: 0.973

Refinement

LightGBM Model -Hyper parameter Tuning(Features importance) :



Refinement

CatBoost Model -Hyper parameter Tuning :

```
from sklearn.model_selection import RandomizedSearchCV
grid = {
    'learning_rate': [0.03,0.05, 0.1],
    'depth':[3,4, 6,8],
    'l2_leaf_reg': [ 3, 5, 7, 9,11]
}
randm = RandomizedSearchCV(estimator=model_cat, param_distributions = grid,
                               cv = 4, n_iter = 10, n_jobs=-1)
randm.fit(X_train, y_train,eval_set=(X_val,Y_val))

    # Results from Random Search
print("\n========================================================")
print(" Results from Random Search " )
print("========================================================")

print("\n The best estimator across ALL searched params:\n",
          randm.best_estimator_)

print("\n The best score across ALL searched params:\n",
          randm.best_score_)

print("\n The best parameters across ALL searched params:\n",
          randm.best_params_)

print("\n ========================================================")
```

Best Parameters

```
The best parameters across ALL searched params:
{'learning_rate': 0.1, 'l2_leaf_reg': 7, 'depth': 6}
```

Refinement

CatBoost Model -Hyper parameter Tuning (metrics with best parameters): :

```
#Print accuracy
#CatBoostClassifier.plot_importance(model_cat, max_num_features=21, importance_type='split')
auc_cat = roc_auc_score(y_test,preds_cat)
print('roc_auc_score of CATBOOST model:', auc_cat)
print('Precision Metric:',precision_score(y_test, preds_cat))
print('Recall Metric:',recall_score(y_test, preds_cat))

#Print Area Under Curve
plt.figure(figsize=(16, 10))
false_positive_rate, recall, thresholds = roc_curve(y_test, preds_cat)
roc_auc = auc(false_positive_rate, recall)
plt.title('Receiver Operating Characteristic (ROC)')
plt.plot(false_positive_rate, recall, 'b', label = 'AUC = %0.3f' %roc_auc)
plt.legend(loc='lower right')
plt.plot([0,1], [0,1], 'r--')
plt.xlim([0.0,1.0])
plt.ylim([0.0,1.0])
plt.ylabel('Recall')
plt.xlabel('Fall-out (1-Specificity)')
plt.show()

#print('AUC score:', roc_auc)
print('roc_auc_score:', roc_auc_score(y_test,preds_cat))
#Print Confusion Matrix
plt.figure(figsize=(16, 10))
cm = confusion_matrix(y_test, preds_cat)
labels = ['No Default', 'Default']
plt.figure(figsize=(8,6))
sns.heatmap(cm, xticklabels = labels, yticklabels = labels, annot = True, fmt='d', cmap="Blues", vmin = 0.2);
plt.title('Confusion Matrix')
plt.ylabel('True Class')
plt.xlabel('Predicted Class')
plt.show()
```

```
roc_auc_score of CATBOOST model: 0.9741754331106635
Precision Metric: 0.9644106150997737
Recall Metric: 0.9687952056209961
```

Roc_auc_score:0.974
Precision: 0.964
Recall: 0.968

Better results than before

D RESULTS

-- 1 Model Evaluation and Validation

-- 2 Justification

01

Model Evaluation and Validation

Model Evaluation and Validation

Now , after finalizing the models hyper parameters tuning and getting the best parameters , we will Evaluate the all Models together to get the best Model according to roc_auc_Score:

	classifiertype	roc_auc_score	Precision Metric	Recall Metric
0	LGB	0.974676	0.960016	0.972515
1	XGBoost	0.974662	0.962318	0.971068
2	CatBoost	0.974175	0.964411	0.968795
3	Random_Forrest_classifier	0.970227	0.933845	0.980161
4	Decision_Tree_Classifier	0.962039	0.950124	0.952676
5	Logistic_regression	0.855733	0.820780	0.812978
6	KNeighborsClassifier	0.780606	0.718866	0.722877

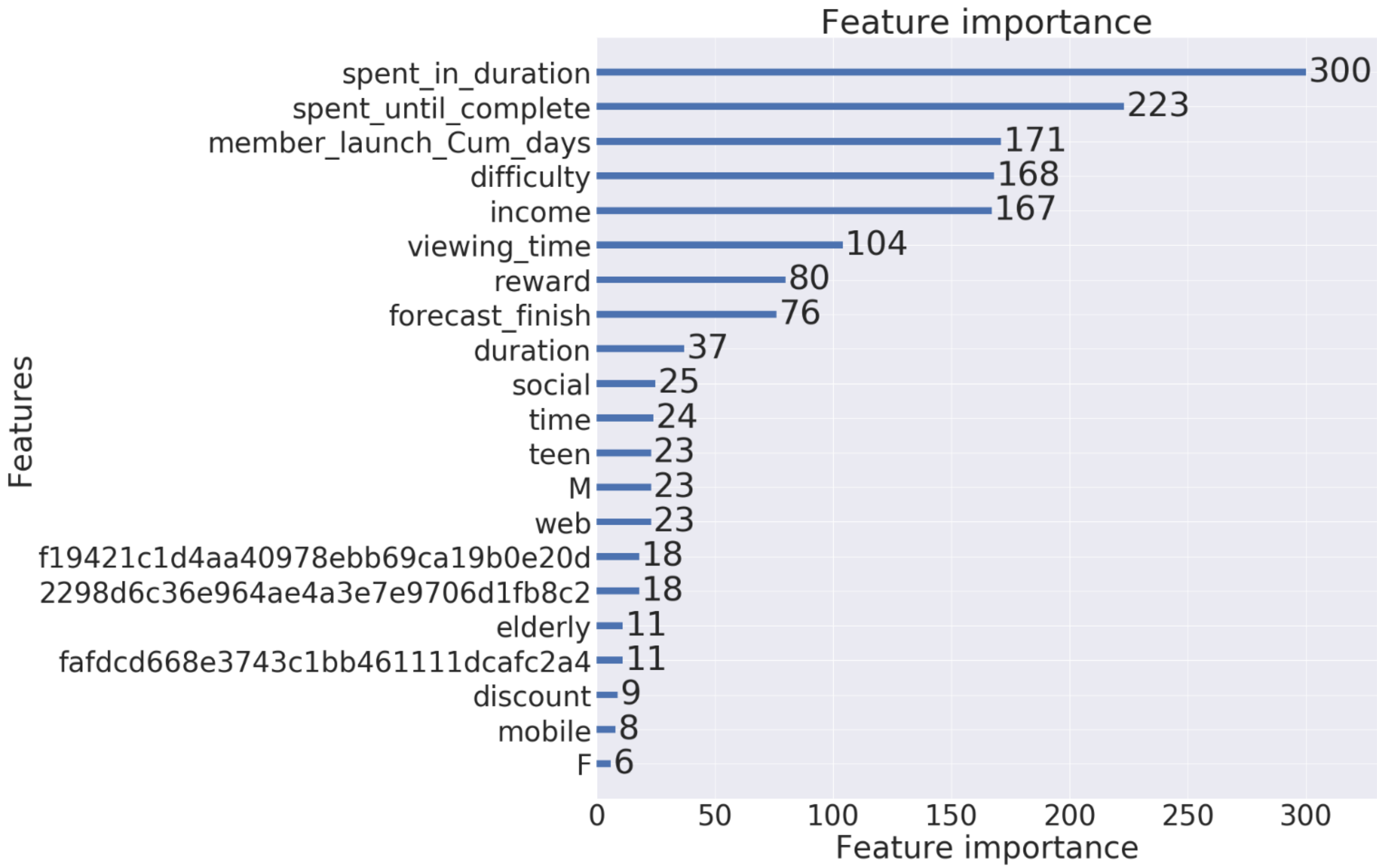
As shown in the abovementioned figure , the best three models are the Boosting models LGB,XGB and CatBoost and their results are so incredibly close.

02

Justification

Justification

Eventually , we can say that the Boosting Models have the Best results in our Problem comparing to our Benchmark Model (Logistic regression) , especially LightGBM Model with the below features



Justification

we can achieve more improvement for our Boosting Models , by using ensembling stacking:
https://www.kaggle.com/arthurtok/introduction-to-ensembling-stacking-in-python

I feel that we can get better results by applying that approach.

References:

1-https://towardsdatascience.com/handling-imbalanced-datasets-in-machine-learning-7a0e84220f28
2-https://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_auc_score.html
3-https://aws.amazon.com/blogs/machine-learning/simplify-machine-learning-with-xgboost-and-amazon-sagemaker/
4-http://uc-r.github.io/gbm_regression
5-https://www.analyticsvidhya.com/blog/2017/08/catboost-automated-categorical-data/
6-https://en.wikipedia.org/wiki/Receiver_operating_characteristic
7-https://www.kaggle.com/arthurtok/introduction-to-ensembling-stacking-in-python
8-https://stackoverflow.com/questions/10373660/converting-a-pandas-groupby-object-to-dataframe

Perfection is not attainable, but if we chase perfection we can catch excellence.

[Vince Lombardi]