# Analysing and Predicting Starbucks Customers' Reponses towards 10 kinds of Promotion Offers


Introduction

Data Set

Data Cleaning and Preprocessing

Exploration analysis

- Q1 Offers Distribution
- Q2 Interest Distribution towards different offers
- Q3 Difficulty Distribution of different offers
- Q4 Index IIR: is the offer significant popular?
- Summary for Data Exploration

Feature Extraction

Model

- Issue1: Offer is going to be sent to a customer, will this offer effective?
  - Issue1 Summary:
- Issue2: Offer is already sent to a customer, is this offer effective?
  - Issue2 Summary:
- Issue3: Given basic infos of a customer, how to recommend an offer with the most effctivity?
  - Issue3 Summary:
- Additional Issue: Neural Network for regeression
  - Additional Issue summary:

Results: deployment of Issue2

Conclusion

References


## Introduction

With the advent of the era of big data, companies are more and more inclined to analyze customer consumption behaviors, in order to formulate specific marketing strategies to promote consumers to complete the transactions.

Customer groups of different ages and different incomes obviously have different consumption habits. Taking age as an example, young people are more susceptible to online and social media advertisements. And more, because of their relatively less savings, they will be more concerned about the price  and the promotion.
In this context, Starbucks has developed an experimental system that simulates user consumption data, and analyzes the data to find the patterns of customer consumption, so as to conduct more targeted promotions and optimize revenue.

Once every few days, Starbucks sends out an offer to users of the mobile app. An offer can be merely an advertisement for a drink or an actual offer such as a discount or BOGO (buy one get one free).
Some users might not receive any offer during certain weeks.
Not all users receive the same offer.

This data set contains simulated data that mimics customer behavior on the Starbucks rewards mobile app, it's a simplified version of the real Starbucks app because the underlying simulator only has one product whereas Starbucks actually sells dozens of products.

The task is to combine transaction, demographic and offer data to determine which demographic groups respond best to which offer type.

More precisely, I aim to answer two questions:

1. What are the main factors that driver customers or groups to complete an offer?
2. Given offer characteristics and user demographics, can we predict whether the customer will complete the offer effectively? What's more, how much money will the customer pay for?

---

# Data Set

- portfolio.json
  containing offer ids and meta data about each offer (duration, type, etc.)

| Columns | Data Type | Explanation | Total Count | NaN Count |
|---|---|---|---|---|
| id | str | id of offer | 10 | – |

| Columns | Data Type | Explanation | Total Count | NaN Count |
|---|---|---|---|---|
| offer_type | str | type of offer<br>**values:**<br>'bogo','discount','informational' | 10 | – |
| difficulty | int | the minimum consumption to complete the offer | 10 | – |
| reward | int | reward after completing the offer | 10 | – |
| duration | int | the valid duration of the offer | 10 | – |
| channels | str list | the channel to send the offer | 10 | – |

- profile.json
  demographic data for each customer

| Columns | Data Type | Explanation | Total Count | NaN Count |
|---|---|---|---|---|
| age | int | the age of customer | 14825 | 2175 |
| became_member_on | int | the enroll date of customer<br>e.g. 20170101 | 17000 | – |
| gender | str | the gender of customer<br>**values:**'male','female','other' | 17000 | – |
| id | str | the id of customer | 17000 | – |
| income | float | the income of customer | 14825 | 2175 |

- transcript.json
  records for transactions, offers received, offers viewed, and offers completed. It shows user purchases made on the app including the timestamp of purchase and the amount of money spent on a purchase.

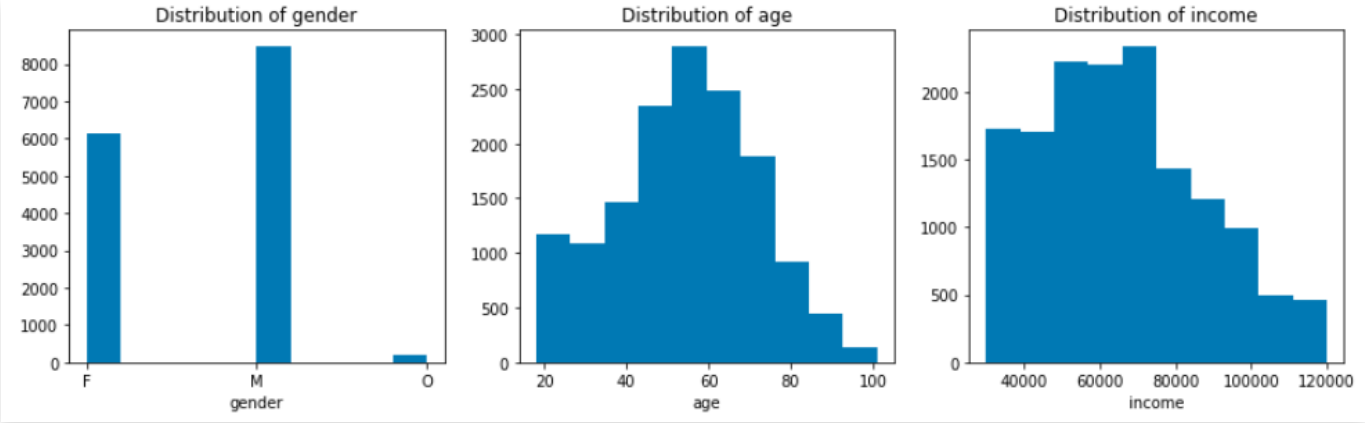
| Columns | Data Type | Explanation | Total Count | NaN Count |
|---|---|---|---|---|
| person | str | the id of customer | 306534 | – |

| Columns | Data Type | Explanation | Total Count | NaN Count |
|---|---|---|---|---|
| event | str | the description of transcript **values:**'offer received','offer viewed','transaction','offer completed' | 306534 | – |
| time | int | the happend time of event(hour) | 306534 | – |
| value | str dict | some includes id of offer, some includes amount of transaction | 306534 | – |

---

## Data Cleaning and Preprocessing

At the begining, I try to know the basic informations about **NaN values**, they all bounded with the unusual age value "118". After all these NaN records have been deleted, Here comes the distributions of gender, age and income.

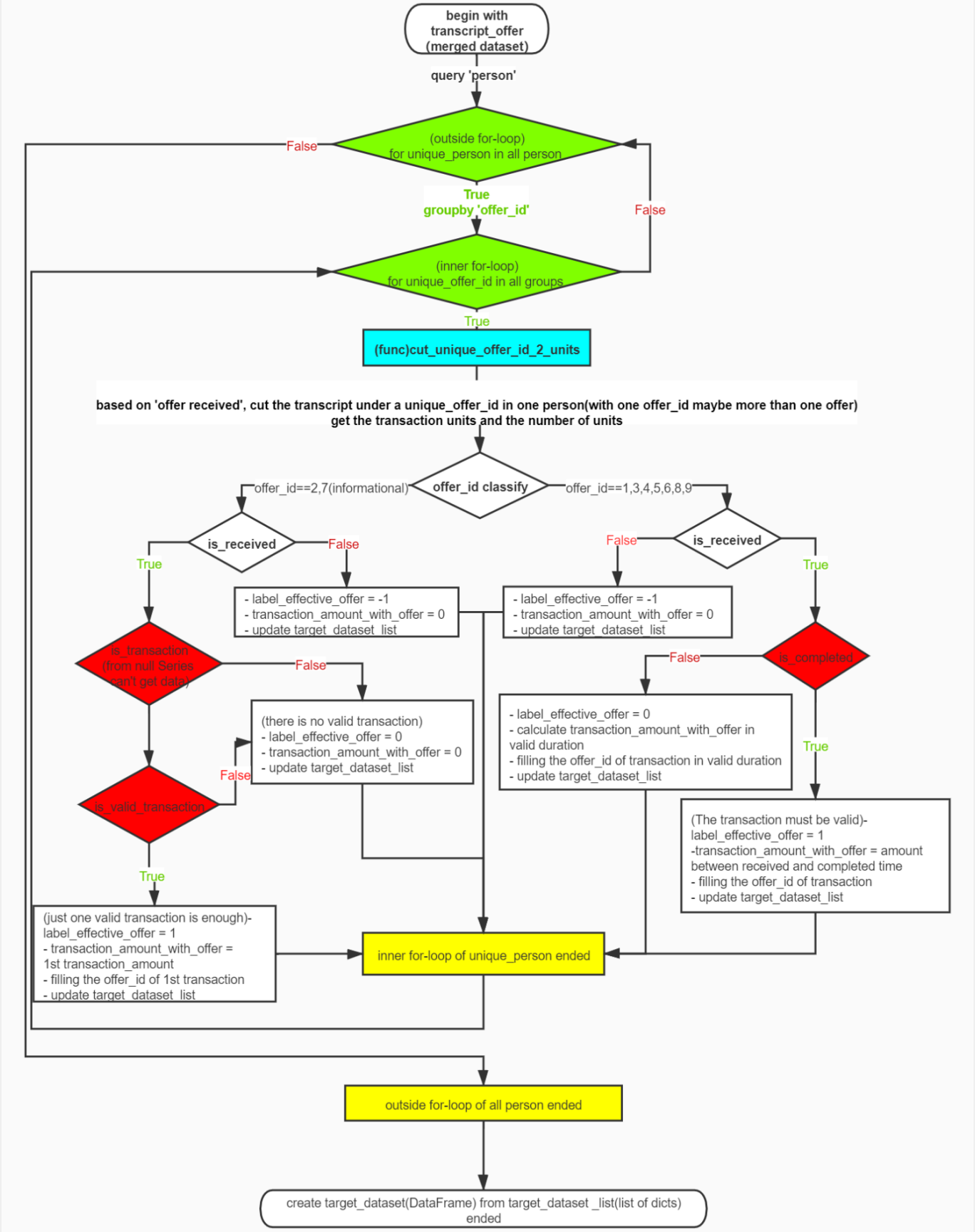



Immediately after, I wrangled the data `transcript` by extracting the value column to offer id and amount columns. What's more, I merged the 'duration' and 'offer_type' infos bounded with the offer id for further data preprocessing. Below is description of the wrangled 'transcript':

| Column | Type | Explanation | Total Count | NaN Count |
|---|---|---|---|---|
| person | int | Id of customer | 272762 | – |
| | | | | |

| | | | | |
|---|---|---|---|---|
| event | str object | State of the record:<br>'offer received', 'offer viewed',<br>'transaction', 'offer completed' | 272762 | – |
| time | float | Happend time of this record | 272762 | – |
| amount | float | How much money paid under this record<br>Notice: exist only when 'event' value is transaction | 272762 | – |
| offer_id | str object | The offer_id bound with this record<br>Notice: '–1' means there is no offer | 272762 | 123957 of '–1' value |
| duration | float | The valid duration of the offer | 272762 | 123957 |
| offer_type | str object | The offer type | 272762 | 123957 |

By using the data set above, based on a self designed program flow chart(File: data_preprocessing_class.py), I extract the transactions infomation direct to the related person and offer.


begin with transcript_offer (merged dataset)
query 'person'
(outside for-loop) for unique_person in all person
False
True
groupby 'offer_id'
False
(inner for-loop) for unique_offer_id in all groups
True
(func)cut_unique_offer_id_2_units
based on 'offer received', cut the transcript under a unique_offer_id in one person(with one offer_id maybe more than one offer) get the transaction units and the number of units
offer_id classify
offer_id==2,7(informational)
offer_id==1,3,4,5,6,8,9
is_received
True
False
- label_effective_offer = -1
- transaction_amount_with_offer = 0
- update target_dataset_list
is_transaction (from null Series can't get data)
False
(there is no valid transaction)
- label_effective_offer = 0
- transaction_amount_with_offer = 0
- update target_dataset_list
is_valid_transaction
False
True
(just one valid transaction is enough)-
label_effective_offer = 1
- transaction_amount_with_offer = 1st transaction_amount
- filling the offer_id of 1st transaction
- update target_dataset_list
is_received
False
True
- label_effective_offer = -1
- transaction_amount_with_offer = 0
- update target_dataset_list
is_completed
False
True
- label_effective_offer = 0
- calculate transaction_amount_with_offer in valid duration
- filling the offer_id of transaction in valid duration
- update target_dataset_list
(The transaction must be valid)-
label_effective_offer = 1
-transaction_amount_with_offer = amount between received and completed time
- filling the offer_id of transaction
- update target_dataset_list
inner for-loop of unique_person ended
outside for-loop of all person ended
create target_dataset(DataFrame) from target_dataset _list(list of dicts) ended

Meantime, I wrangle the data type of some features (e.g. the 'value' column in `portfolio` is a list, should be unfolded), and transform some features to normal form(e.g. transform the member enroll date from 'int' to 'date').

Then I divide all the customers into 12 segments according to 'age' and 'income', which tends to show a group characteristics.

What's more important, considering all kinds of response situations to offer, I divide all transactions to 4 groups:

| | |
|---|---|
| 1. none_offer | never received offer |
| 2. no_care_offer | received, but don't care about the offer |
| 3. tried_offer | tried to do some transaction, but not complete within the duration of offer |
| 4. effective_offer | complete the offer |

Finally, I get an ideal wrangled data set with label of segments and response groups described as follows(File: model_dataset_raw):

| Column | Type | Explanation | Total Count | NaN Count |
|---|---|---|---|---|
| person | int | id of customer | 66506 | – |
| offer_id | str object | values: '–1', '0'–'9'<br>represent 10 offers, '–1'means no offer received | 66506 | – |
| time_received | float | time when offer received<br>'NaN' represents not received | 66506 | 5 |
| time_viewed | float | time when offer viewed<br>'NaN' represents not viewed | 66506 | 16646 |
| time_transaction | str object | time then transaction(s) takes place<br>'' represents there is no transaction<br>',3.0,5.0' means there are two transactions under this offer, one is at time 3.0, another in at time 5.0 | 66506 | 8754 |
| time_completed | float | time when offer completed<br>'NaN' represents not completed | 66506 | 26099 |
| amount_wi | float | How much money has been paid under this | 66506 | – |

| | | | | |
|---|---|---|---|---|
| th_offer | | offer<br>'0.0' represent no transaction | | |
| label_effective_offer | int | the label to mark the completing level of offer<br>**More details See Notice below** | 66506 | – |
| reward | float | Reward after completing the offer | 66501 | 5 |
| difficulty | float | The minimum consumption to complete the offer | 66501 | 5 |
| duration | float | The valid duration of the offer<br>'NaN' implies the offer_id is '–1' | 66501 | 5 |
| offer_type | str object | 'bogo', 'discount','informational' | 66501 | 5 |
| email | float | One Channel to send offer | 66501 | 5 |
| mobile | float | One Channel to send offer | 66501 | 5 |
| social | float | One Channel to send offer | 66501 | 5 |
| web | float | One Channel to send offer | 66501 | 5 |
| gender | str object | 'male','female','other' | 66506 | – |
| age | int | Age of the customer | 66506 | – |
| income | float | Yearly income of the customer | 66506 | – |
| member_days | int | The days from enroll date to 2019.01.01 | 66506 | – |
| label_group | str object | 4 groups of resonse to offers<br>Values **see Notice2. below** | 66506 | – |
| label_seg | int | 1–12: 12 segments based on age and income | 66506 | – |

**Notice 1.: label_effective_offer** (Label describes the completed level of offer)
**(Attention: there is no infomation about 'offer viewed')**

| Values | Meaning |
|---|---|
| 1 | for informational offer there is at least one transaction within duration;<br>for other offer there should be 'offer completed' |
| 0 | for informational offer there is no valid transaction within the duration but 'offer received'; |

| | |
|---|---|
| | for other offers there is no 'offer completed', but within duration there maybe some amount, although the amount of transactions not fulfil requirements |
| –1 | the initial label, when there is no 'offer received', the label keeps '–1' |
| –2 | represent some people: they only have transactions within all the experimantal time , no offer was sent to them |

**Notice 2.: label_group** (4 groups of resonse to offers)

| Group | received | viewed | valid completed | transaction amount | Scenario | Logical expression |
|---|---|---|---|---|---|---|
| 1.none_offer | 0 | 0 | 0 | | haven't received the offer | label_effective_offer.isin([–1, –2]) & time_viewed == NaN |
| 2.no_care | 1 | 0 | – | | received but not viewed. regarded as "don't care" | label_effective_offer.isin([0, 1]) & time_viewed == NaN |
| | 1 | 1 | 0 | =0.0 | received, viewed but no transaction | label_effective_offer == 0 & amount == 0.0 & time_viewed.notnull() |
| | 1 | 1 | 1 viewed after completed | | received, but completed unintentionally, namely viewing after completed | label_effective_offer == 1 & time_viewed > time_completed |
| 3.tried | 1 | 1 | 0 | >0.0 | received, viewed, have transaction, | label_effective_offer == 0 |

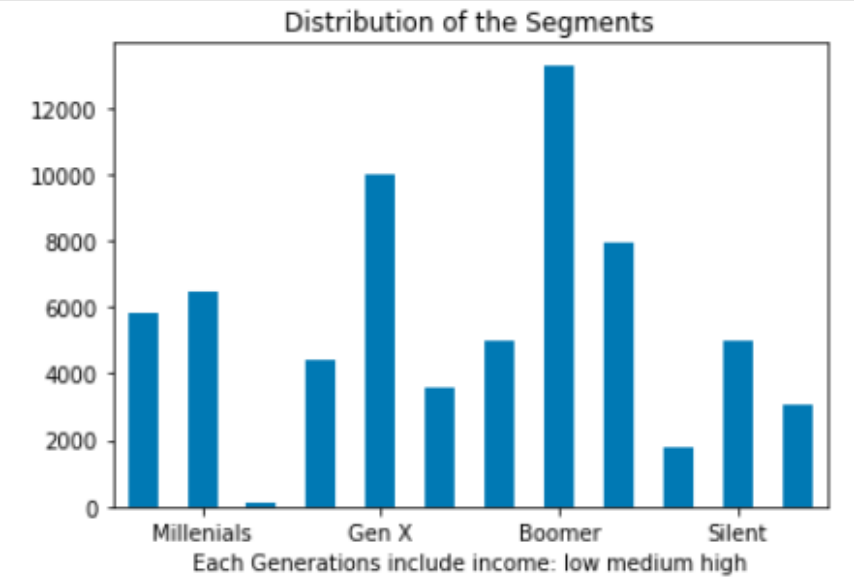
| | | | | | | |
|---|---|---|---|---|---|---|
| | | | | | but amount less than 'difficulty' | & amount > 0.0 & time_viewed.notnull() |
| 4.effctive_offer | 1 | 1 | 1 viewed before completed | | viewed before completed, effctive offer | label_effective_offer == 1 & time_viewed < time_completed |

**Notice 3.: label_seg** (12 segments based on age and income)

| Segment # | Age Group (edge included) (Experiment in 2018) | Income |
|---|---|---|
| 1 | Millenials(–21 & 22–37) | low |
| 2 | Millenials(–21 & 22–37) | medium |
| 3 | Millenials(–21 & 22–37) | high |
| 4 | Gen X(38–53) | low |
| 5 | Gen X(38–53) | medium |
| 6 | Gen X(38–53) | high |
| 7 | Baby Boomer(54–72) | low |
| 8 | Baby Boomer(54–72) | medium |
| 9 | Baby Boomer(54–72) | high |
| 10 | Silent(73–90 & 91+) | low |
| 11 | Silent(73–90 & 91+) | medium |
| 12 | Silent(73–90 & 91+) | high |

Income Level:

| Income | Values($) |
|---|---|
| low | 30,000–50,000 |
| medium | 50,001–82,500 |
| high | 82,501–120,000 |

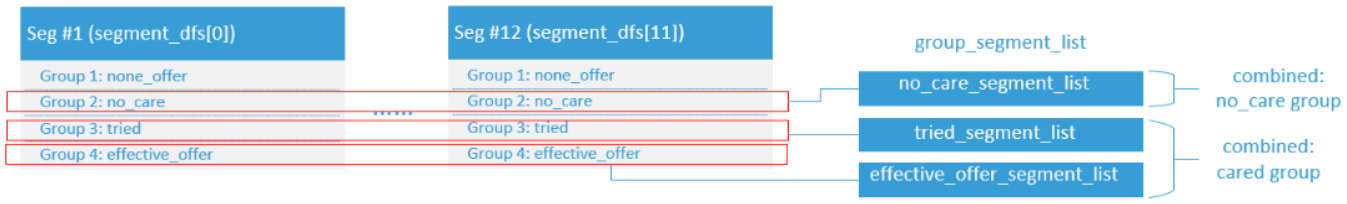



**Notice 4.: offer_id** (10 Kinds of offer)

| offer_id | type | duration | requirement | reward |
|---|---|---|---|---|
| 0 | bogo | 7 | 10 | 10 |
| 1 | bogo | 5 | 10 | 10 |
| 2 | infomational | 4 | – | – |
| 3 | bogo | 7 | 5 | 5 |
| 4 | discount | 10 | 20 | 5 |
| 5 | discount | 7 | 7 | 3 |
| 6 | discount | 10 | 10 | 2 |
| 7 | informational | 3 | – | – |
| 8 | bogo | 5 | 5 | 5 |
| 9 | discount | 7 | 10 | 2 |

# Exploration analysis

Below is the structure of the analysis data:



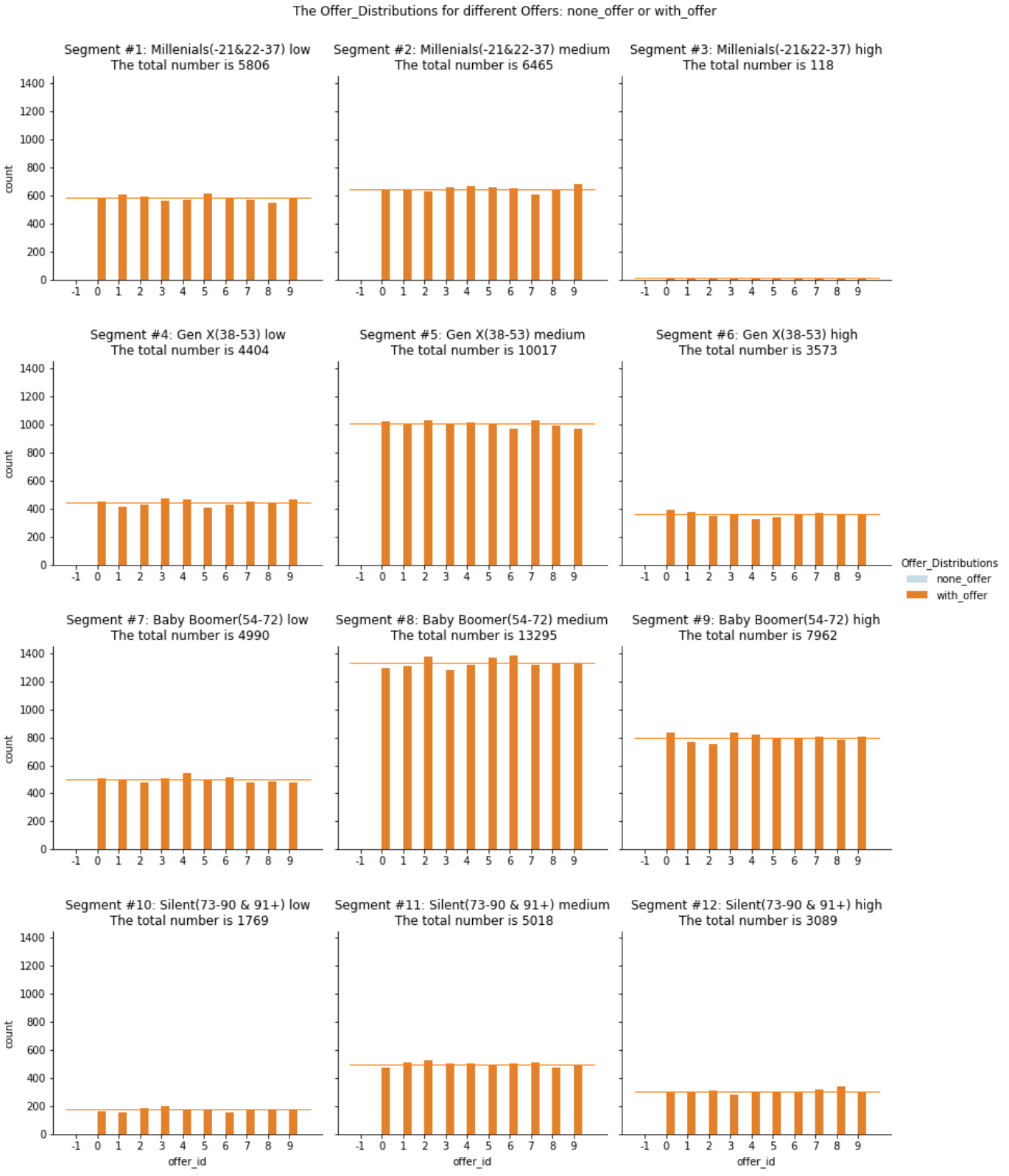

The data set has been divided into 12 Segments based on age and income, and each segment has 4 response groups.

With different combination of the 4 response groups, there are 3 questions to be explored.

| | Combined group set 1 | Combined group set 2 |
|---|---|---|
| 1 Offers Distribution | **received none offer** *Group(s): none_offer* | **received offer** *Group(s): no_care, tried, effective_offer* |
| 2 Interest Distribution towards different offers | **don't care** *Group(s): no_care* | **care** *Group(s): tried, effective_offer* |
| 3 Difficulty Distribution of different offers | **tried but not completed** *Group(s): tried* | **effectively completed** *Group(s): effective_offer* |

What's more, I will use the index IIR to discuss, whether the offer is significantly popular by customer.

## Q1 Offers Distribution


The Offer_Distributions for different Offers: none_offer or with_offer
Segment #1: Millenials(-21&22-37) low
The total number is 5806
Segment #2: Millenials(-21&22-37) medium
The total number is 6465
Segment #3: Millenials(-21&22-37) high
The total number is 118
Segment #4: Gen X(38-53) low
The total number is 4404
Segment #5: Gen X(38-53) medium
The total number is 10017
Segment #6: Gen X(38-53) high
The total number is 3573
Segment #7: Baby Boomer(54-72) low
The total number is 4990
Segment #8: Baby Boomer(54-72) medium
The total number is 13295
Segment #9: Baby Boomer(54-72) high
The total number is 7962
Segment #10: Silent(73-90 & 91+) low
The total number is 1769
Segment #11: Silent(73-90 & 91+) medium
The total number is 5018
Segment #12: Silent(73-90 & 91+) high
The total number is 3089
count
offer_id
Offer_Distributions
none_offer
with_offer

```
# the whole group of offer not received
is_none_offer = (summary_dataset.offer_id == '-1')
summary_dataset[is_none_offer]
```

| ... | email | mobile | social | web | gender | age | income | member_days | Offer_Distributions | label_seg |
|---|---|---|---|---|---|---|---|---|---|---|
| ... | NaN | NaN | NaN | NaN | F | 66 | 34000.0 | 459 | none_offer | 7 |
| ... | NaN | NaN | NaN | NaN | F | 72 | 35000.0 | 444 | none_offer | 7 |
| ... | NaN | NaN | NaN | NaN | F | 54 | 72000.0 | 725 | none_offer | 8 |
| ... | NaN | NaN | NaN | NaN | F | 55 | 88000.0 | 868 | none_offer | 9 |
| ... | NaN | NaN | NaN | NaN | M | 91 | 70000.0 | 1184 | none_offer | 11 |

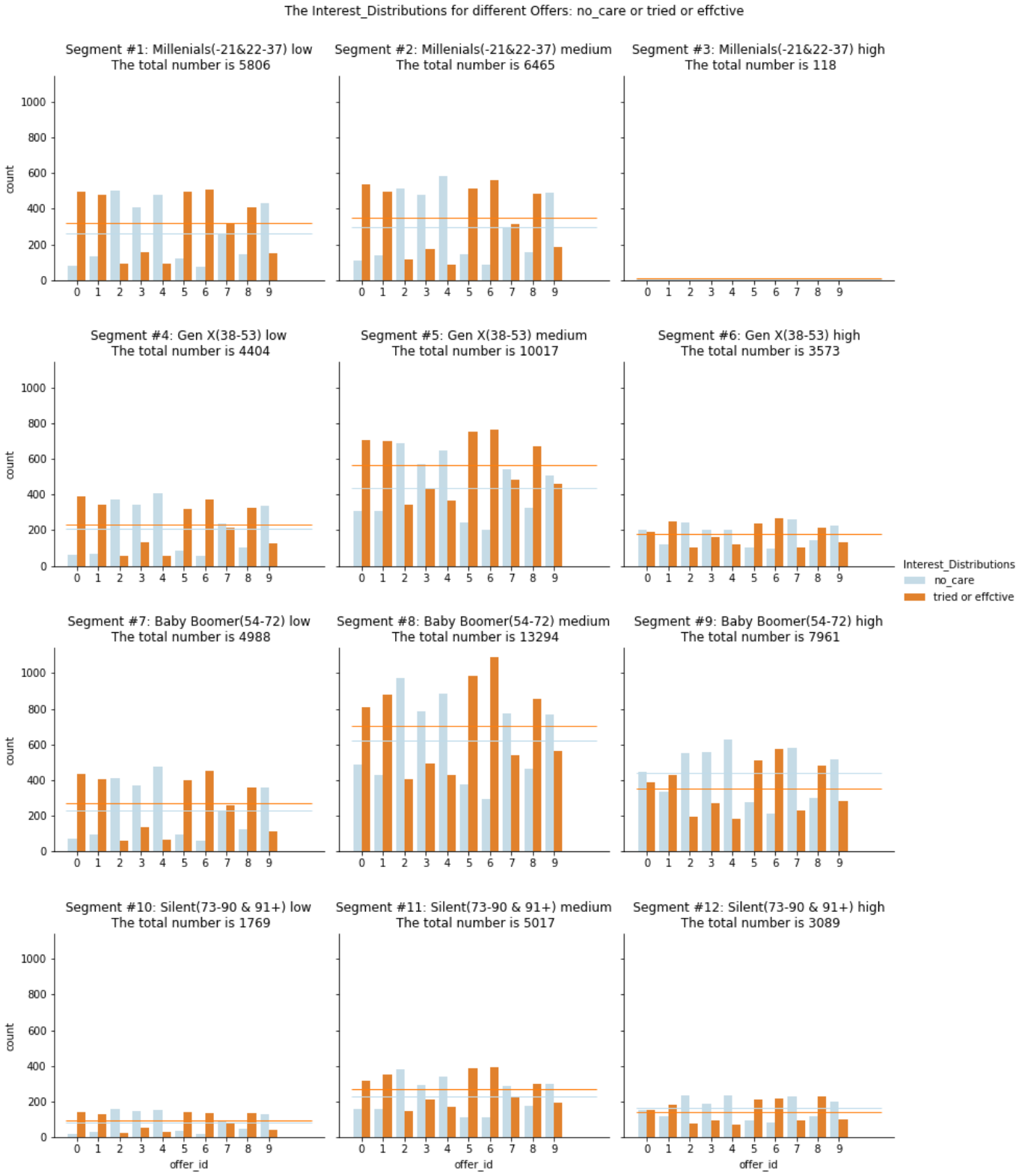
- **In general**

1. There are just 5 person, who never received the offer (**offer_id is '-1'**)
   a. Two in segment #7; One in each segment #8 #9 #11.
   b. They all are more than 50 years old and has more than one year membership. It seems they are regular customer and needn't receive the offer.
2. The offer distributions under income: *see segment #3 VS. segment #12*
   a. Young people have not so much money.
   b. Elder people tend to have more savings.
3. The offer distributions under age: *see segment #1 VS. segment #10*
   a. In the low income group, compared with young person, the elder person seems to receive less offers

- **In segments(subplots)**

1. In each segment, person reveive almost the same quantity of offers. *See the average line.*

## Q2 Interest Distribution towards different offers


The Interest_Distributions for different Offers: no_care or tried or effctive
Segment #1: Millenials(-21&22-37) low
The total number is 5806
Segment #2: Millenials(-21&22-37) medium
The total number is 6465
Segment #3: Millenials(-21&22-37) high
The total number is 118
Segment #4: Gen X(38-53) low
The total number is 4404
Segment #5: Gen X(38-53) medium
The total number is 10017
Segment #6: Gen X(38-53) high
The total number is 3573
Segment #7: Baby Boomer(54-72) low
The total number is 4988
Segment #8: Baby Boomer(54-72) medium
The total number is 13294
Segment #9: Baby Boomer(54-72) high
The total number is 7961
Segment #10: Silent(73-90 & 91+) low
The total number is 1769
Segment #11: Silent(73-90 & 91+) medium
The total number is 5017
Segment #12: Silent(73-90 & 91+) high
The total number is 3089
count
offer_id
Interest_Distributions
no_care
tried or effctive

```
# offer 0, 1, 5, 6, 8
portfolio_raw[portfolio_raw.offer_id.isin(['0','1','5','6','8'])]
```

| | reward | difficulty | duration | offer_type | offer_id | email | mobile | social | web |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 10 | 10 | 7 | bogo | 0 | 1 | 1 | 1 | 0 |
| **1** | 10 | 10 | 5 | bogo | 1 | 1 | 1 | 1 | 1 |
| **5** | 3 | 7 | 7 | discount | 5 | 1 | 1 | 1 | 1 |
| **6** | 2 | 10 | 10 | discount | 6 | 1 | 1 | 1 | 1 |
| **8** | 5 | 5 | 5 | bogo | 8 | 1 | 1 | 1 | 1 |

```
# offer 2, 3, 4
portfolio_raw[portfolio_raw.offer_id.isin(['2','3','4'])]
```

| | reward | difficulty | duration | offer_type | offer_id | email | mobile | social | web |
|---|---|---|---|---|---|---|---|---|---|
| **2** | 0 | 0 | 4 | informational | 2 | 1 | 1 | 0 | 1 |
| **3** | 5 | 5 | 7 | bogo | 3 | 1 | 1 | 0 | 1 |
| **4** | 5 | 20 | 10 | discount | 4 | 1 | 0 | 0 | 1 |

```
# offer 7, 9
portfolio_raw[portfolio_raw.offer_id.isin(['7','9'])]
```

| | reward | difficulty | duration | offer_type | offer_id | email | mobile | social | web |
|---|---|---|---|---|---|---|---|---|---|
| **7** | 0 | 0 | 3 | informational | 7 | 1 | 1 | 1 | 0 |
| **9** | 2 | 10 | 7 | discount | 9 | 1 | 1 | 0 | 1 |

- **In general**

1. The customers care more about offer 0, 1, 5, 6, 8
2. Offer 2, 3, 4 are not in interest
3. For offer 7, 9, some cares, some doesn't care

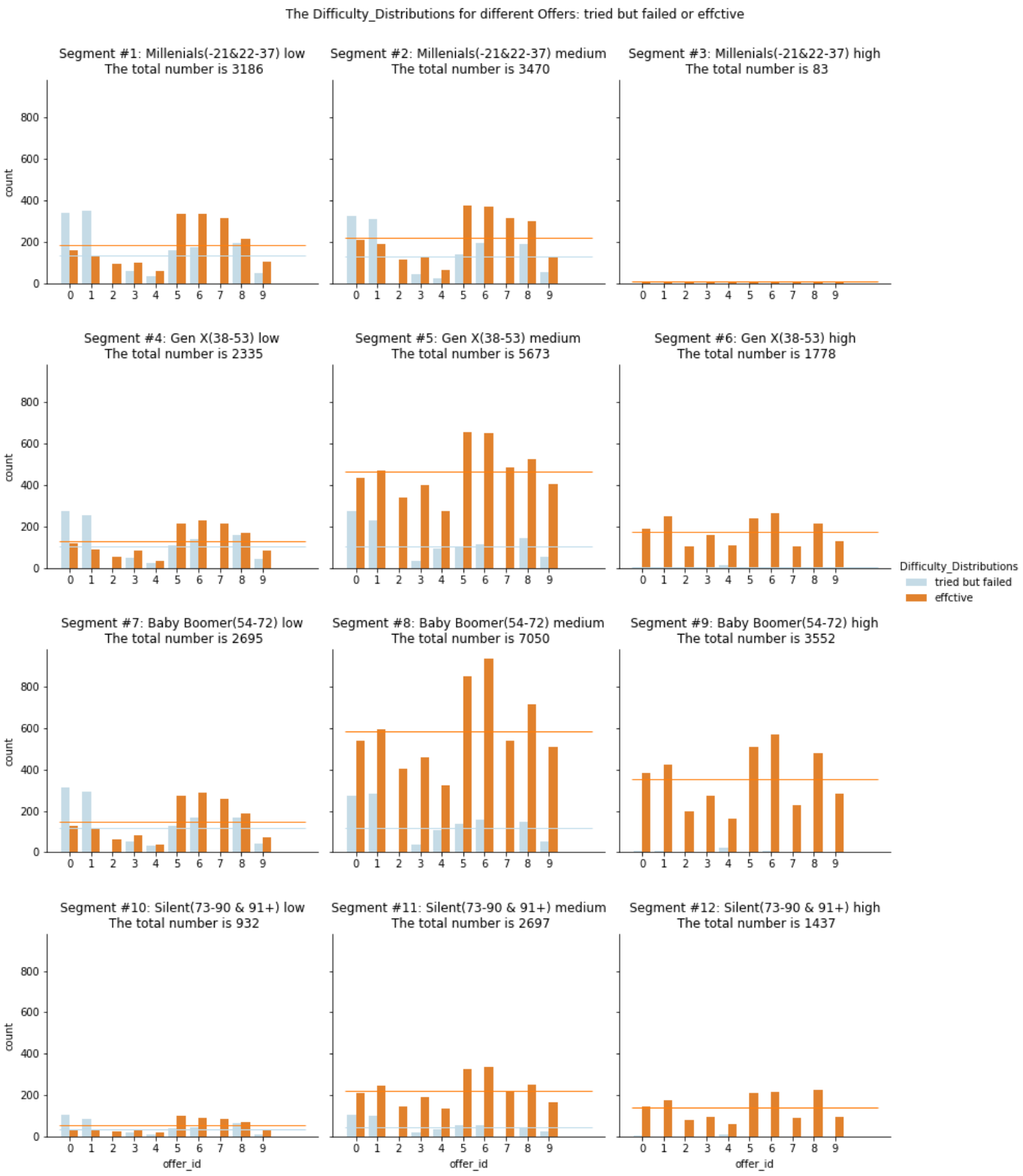
From the three tables above, we can conclude that:

1. 'social' is an import factor to attract people to complete the offer
2. 'bogo' with medium difficulty are more popular
3. 'discount' with less difficulty are more popular

- **In segments(subplots)**

1. Offer 0, 1
    a. The high income groups show less interst compared with other income group. *(see Segments #3 #6 #9 #12)*
2. Offer 5, 6, 8
    a. Customer shows great interst(in all Segments)

## Q3 Difficulty Distribution of different offers


The Difficulty_Distributions for different Offers: tried but failed or effctive
Segment #1: Millenials(-21&22-37) low
The total number is 3186
Segment #2: Millenials(-21&22-37) medium
The total number is 3470
Segment #3: Millenials(-21&22-37) high
The total number is 83
Segment #4: Gen X(38-53) low
The total number is 2335
Segment #5: Gen X(38-53) medium
The total number is 5673
Segment #6: Gen X(38-53) high
The total number is 1778
Segment #7: Baby Boomer(54-72) low
The total number is 2695
Segment #8: Baby Boomer(54-72) medium
The total number is 7050
Segment #9: Baby Boomer(54-72) high
The total number is 3552
Segment #10: Silent(73-90 & 91+) low
The total number is 932
Segment #11: Silent(73-90 & 91+) medium
The total number is 2697
Segment #12: Silent(73-90 & 91+) high
The total number is 1437
count
offer_id
Difficulty_Distributions
tried but failed
effctive

```
# offer 5, 6, 8
portfolio_raw[portfolio_raw.offer_id.isin(['5','6','8'])]
```

| | reward | difficulty | duration | offer_type | offer_id | email | mobile | social | web |
|---|---|---|---|---|---|---|---|---|---|
| 5 | 3 | 7 | 7 | discount | 5 | 1 | 1 | 1 | 1 |
| 6 | 2 | 10 | 10 | discount | 6 | 1 | 1 | 1 | 1 |
| 8 | 5 | 5 | 5 | bogo | 8 | 1 | 1 | 1 | 1 |

```
# offer 0, 1
portfolio_raw[portfolio_raw.offer_id.isin(['0','1'])]
```

| | reward | difficulty | duration | offer_type | offer_id | email | mobile | social | web |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 10 | 10 | 7 | bogo | 0 | 1 | 1 | 1 | 0 |
| 1 | 10 | 10 | 5 | bogo | 1 | 1 | 1 | 1 | 1 |

- **Summary: Level of completion for each offer**

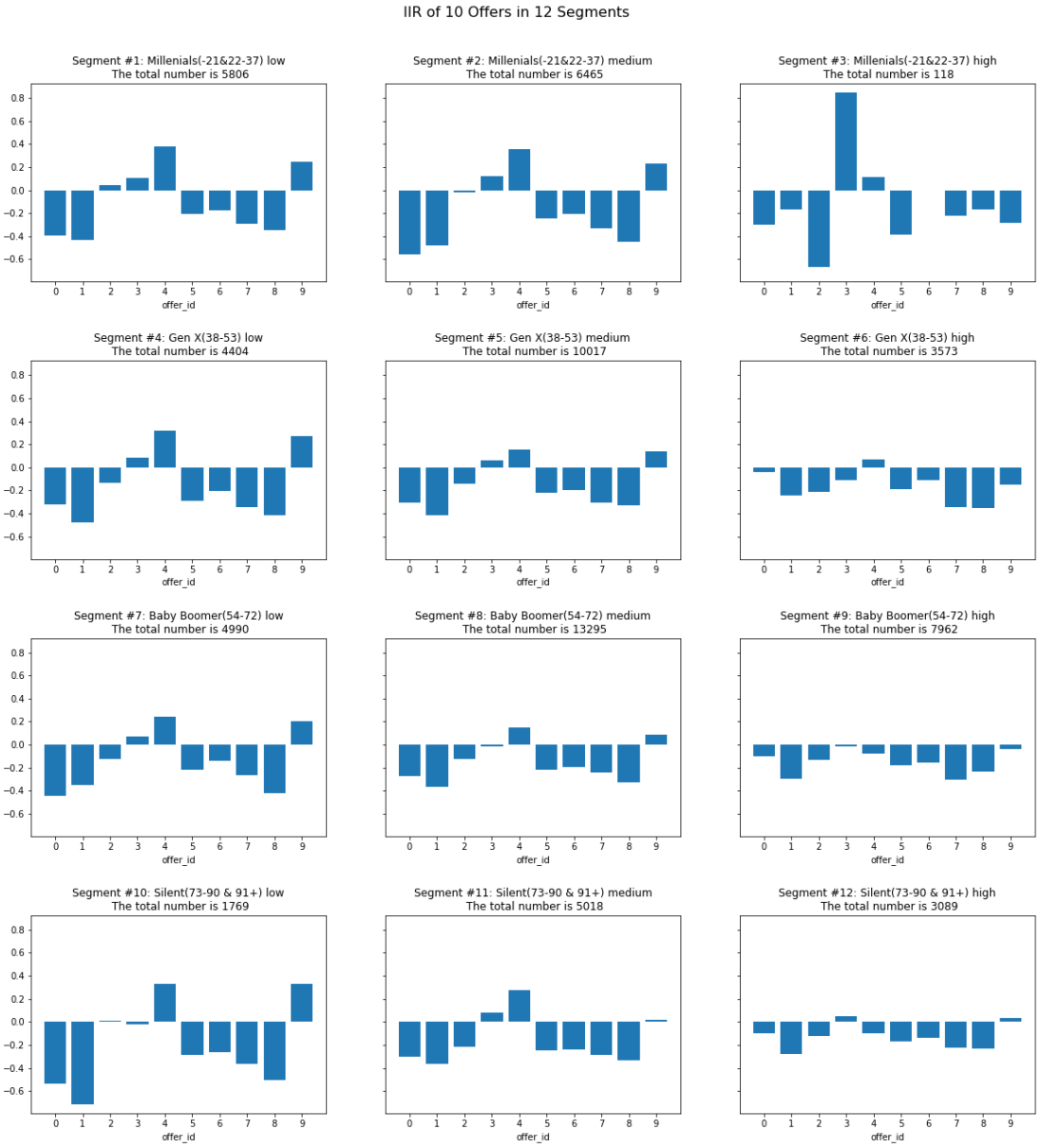
1. Offer 5, 6, 8 are better completed
   a. most are 'discount', and there is 'social' factor
   b. 'difficulty' of 'bogo' is not so much
2. Offer 0, 1 are harder to complete
   a. 'difficulty' of 'bogo' is a little bit heavy
3. Offer 7: an informational offer
   a. richer customers don't care (*see segment #6 #9 #12*, compared to the average line)
   b. more attracted to less rich people (see average line)
4. Offer 2, 3, 4 are more attracted in medium elder and rich people (*see segment #5 #8*)
5. The person with high income tends to complete all offers (*see segment #6 #9 #12*)
   a. even for the offer 0, 1

## Q4 Index IIR: is the offer significant popular?

Definition of IIR: Incremental Response Rate

$$IIR = \frac{n_1}{sum_1} - \frac{n_0}{sum_0}$$

| Symbol | Meaning |
|---|---|
| $n_1$ | number of Purchasers in Treated Group |
| $sum_1$ | Total number of Purchasers in Treated Group |
| $n_0$ | number of Purchasers in Control Group |
| $sum_0$ | Total number of Purchasers in Control Group |




```
# offer 0, 1, 2, 3, 4
portfolio_raw[portfolio_raw.offer_id.isin(['0','1','2','3','4'])]
```

| | reward | difficulty | duration | offer_type | offer_id | email | mobile | social | web |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 10 | 10 | 7 | bogo | 0 | 1 | 1 | 1 | 0 |
| **1** | 10 | 10 | 5 | bogo | 1 | 1 | 1 | 1 | 1 |
| **2** | 0 | 0 | 4 | informational | 2 | 1 | 1 | 0 | 1 |
| 3 | 5 | 5 | 7 | bogo | 3 | 1 | 1 | 0 | 1 |
| **4** | 5 | 20 | 10 | discount | 4 | 1 | 0 | 0 | 1 |

- In general: which offer seems popular?

1. Offer 2, 3, 4 have positive IIR
    a. for offer 4, the difficulty is 20, maybe few people want to complete it, so it shows high IIR
    b. for offer 3, difficulty is not big, but reward is ok, so it's popular
2. Offer 0,1 have a huge negtive IIR
    a. specially for the low income person(*see segments #1 #4 #7 #10*)

- **In segments(subplots)**

1. The rich people seem not so excited about offer received(*see Segments #6 #9 #12*)
    a. Maybe they are too rich to be encouraged from the reward of offer

## Summary for Data Exploration

1. The channel to send an offer
    - Through **'social'** is a better way.
2. The type of offer
    - People like 'discount', because the reward is real money compared to 'bogo'(get another same thing) and 'informational'(just information)
3. The content of offer
    - If the 'difficulty' is too much, e.g. 20, people have less desire to complete the offer
    - 5–10 maybe a good range of 'difficulty'
4. For all offers, considering of Interest(care) and Difficulty(Level of completion)
    - Offer 5, 6, 8 are more attracted and more easy to complete: could be sent to all customers
    - Offer 0, 1 are hard to complete, but could be sent to high income customers
    - Offer 2, 3, 4 are more attracted to the medium elder, income people(`Segment #5 #8`)
    - Offer 7 for less income people(maybe the information is attracted by them)

---

# Feature Extraction

Based on the `model_dataset_raw` above, after some objects columns(offer_id, offer_type, gender etc.) transformed to 0–1 variables, all the potential features have been extracted.

| Features | Type | Explanation | Total Count | NaN Count |
|---|---|---|---|---|
| **person** | int | | 66501 | – |
| **time_received** | float | | 66501 | – |
| **time_viewed** | float | | 66501 | – |
| | | | | |

| | | | | |
|---|---|---|---|---|
| **transaction_cnt** | int | count of transactions under this offer of the customer | 66501 | – |
| **time_completed** | float | | 66501 | – |
| **amount_with_offer** | float | How much money has been paid under this offer<br>'0.0' represent no transaction | 66501 | – |
| **amount_total** | float | Total amount of paid money for each customer | 66501 | – |
| **offer_received_cnt** | float | Count of all received offers | 66501 | – |
| **reward** | float | Reward after completing the offer | 66501 | – |
| **difficulty** | float | The minimum consumption to complete the offer | 66501 | – |
| **duration** | float | The valid duration of the offer | 66501 | – |
| **email** | float | One Channel to send offer | 66501 | – |
| **mobile** | float | One Channel to send offer | 66501 | – |
| **social** | float | One Channel to send offer | 66501 | – |
| **web** | float | One Channel to send offer | 66501 | – |
| | | | | |

| | | | | |
|---|---|---|---|---|
| **age** | int | Age of the customer | 66501 | – |
| **income** | float | Yearly income of the customer | 66501 | – |
| **member_days** | int | The days from enroll date to 2019.01.01 | 66501 | – |
| **label_seg** | int | 1–12: 12 segments based on age and income | 66501 | – |
| **gender_F**<br>**gender_M**<br>**gender_O** | int | 0–1 variables of genedr | 66501 | – |
| **group_effective_offer**<br>**group_no_care**<br>**group_tried** | int | 0–1 variables of group(the group of none_offer has been removed) | 66501 | – |
| **offer_0**<br>**offer_1**<br>**.....**<br>**offer_9** | int | 0–1 variables of 10 kinds offers | 66501 | – |

---

# Model

I wonder whether Machine Learning will find some intersting points of the data. Especially in the following situations:

1. Offer is going to be sent to a customer, will this offer effective?
2. Offer is already sent to a customer, is this offer effective?
3. Given basic infos of a customer, how to recommend an offer with the most effctivity?

To answer these questions, I build a model pipeline:

- Select features and target(for different issue concerned use different features and target)
- Select classifiers and compare the perform of all classifiers
- Select the suitable parameters of the best performed classifier by using grid search method

- Analyse the result

**Notice:** One Neural Network is also built for regeression analysing.

## Issue1: Offer is going to be sent to a customer, will this offer effective?

| | Object | Description |
|---|---|---|
| **Data Set** | Subset data of 3 offer response groups | • no_care<br>• tried<br>• effective_offer |
| **Target** | label_group | 0: customer doesn't care the offer<br>1: Within the duration of offer, customer tried or completed the transactions |
| **Features** | age | basic info about customer |
| | income | basic info about customer |
| | member_days | basic info about customer |
| | gender_ | basic info about customer<br>(3 kinds of 0–1 variables) |
| | offer_ | offer id<br>(10 kinds of 0–1 variables) |
| | amount_total | amount paid of all transactions |
| | offer_received_cnt | number of all received offers |
| | time_received | receive time for this offer |

Model training result:

| | model | train_time | test_time | train_score | test_score |
|---|---|---|---|---|---|
| 0 | KNeighborsClassifier | 2.942335 | 13.412032 | 0.817613 | 0.662431 |
| 1 | SVC | 1685.705038 | 22.675041 | 0.699474 | 0.696414 |
| 2 | NuSVC | 22959.358365 | 43.541081 | 0.780620 | 0.703406 |
| 3 | DecisionTreeClassifier | 2.100798 | 0.068962 | 1.000000 | 0.649124 |
| 4 | RandomForestClassifier | 27.510256 | 0.927469 | 0.999981 | 0.721750 |
| 5 | AdaBoostClassifier | 6.932032 | 0.262848 | 0.715226 | 0.716788 |
| 6 | GradientBoostingClassifier | 25.969138 | 0.057967 | 0.730808 | 0.727163 |

### Issue1 Summary:

1. The first model KNeighborsClassifier could be a reference model
2. SVC and NuSVC take more time, I attempt to continue without them.
3. DecisionTreeClassifier and RandomForestClassifier both have a high score in training, is there something speciall?
    - Notice that: the test time is much more less than train time. The possible reason is that the data set has a simple structure so that all predict with same result(*see the deployment of Issue2*: no matter how I change the input data, it all shows the same result.)
4. For all, the accuracy of predicting is around 70%, it seems models are not so appropriate in this situation.

## Issue2: Offer is already sent to a customer, is this offer effective?

| | Object | Description |
|---|---|---|
| **Data Set** | Subset data of 3 offer response groups | • no_care<br>• tried<br>• effective_offer |
| **Target** | label_group | 0: customer doesn't care the offer<br>1: Within the duration of offer, customer tried or completed the transactions |
| **Features** | age | basic info about customer |
| | income | basic info about customer |

| | | |
|---|---|---|
| | member_days | basic info about customer |
| | gender_ | basic info about customer<br>(3 kinds of 0–1 variables) |
| | offer_ | offer id<br>(10 kinds of 0–1 variables) |
| | amount_total | amount paid of all transactions |
| | offer_received_cnt | number of all received offers |
| | time_received | receive time for this offer |
| | amount_with_offer | amount paid for this offer |
| | time_viewed | view time for this offer.<br>If not, the value is 0.0 |

Model training result:

| | model | train_time | test_time | train_score | test_score |
|---|---|---|---|---|---|
| 0 | KNeighborsClassifier | 1.301255 | 32.317506 | 0.884398 | 0.784903 |
| 1 | DecisionTreeClassifier | 0.609653 | 0.015999 | 1.000000 | 0.878280 |
| 2 | RandomForestClassifier | 10.026261 | 0.350799 | 1.000000 | 0.912413 |
| 3 | AdaBoostClassifier | 3.316102 | 0.184913 | 0.897030 | 0.896173 |
| 4 | GradientBoostingClassifier | 11.604355 | 0.033999 | 0.911692 | 0.908804 |

### Issue2 Summary:

1. As a reference model, KNeighborsClassifier performs not bad.
2. DecisionTreeClassifier and RandomForestClassifier both have a full score in training, is there something speciall?
    - Notice that: the test time is much more less than train time. The possible reason is that the data set has a simple structure so that all predict with same result(*see the deployment of Issue2*: no matter how I change the input data, it all shows the same result.)

P.S.: I've used the GradientBoostingClassifier as the target model to deploy my project.

## Issue3: Given basic infos of a customer, how to recommend an offer with the most effctivity?

| | Object | Description |
|---|---|---|
| **Data Set** | Subset data of 2 offer response groups(at least transaction exists) | • tried<br>• effective_offer |
| **Target** | offer_(10 classes) | 0: uneffective in this offer_id<br>1: effective in this offer_id |
| **Features** | age | basic info about customer |
| | income | basic info about customer |
| | member_days | basic info about customer |
| | gender_ | basic info about customer<br>(3 kinds of 0–1 variables) |
| | amount_total | amount paid of all transactions |
| | offer_received_cnt | number of all received offers |
| | time_received | receive time for this offer |
| | amount_with_offer | amount paid for this offer |
| | time_viewed | view time for this offer.<br>If not, the value is 0.0 |

Model training result:

| | model | train_time | test_time | train_score | test_score |
|---|---|---|---|---|---|
| 0 | KNeighborsClassifier | 0.535691 | 5.152049 | 0.236869 | 0.048868 |
| 1 | DecisionTreeClassifier | 2.167758 | 0.016993 | 1.000000 | 0.148897 |
| 2 | RandomForestClassifier | 31.097193 | 1.230297 | 0.999534 | 0.008169 |
| 3 | MultiOutputClassifier | 54.604750 | 0.126927 | 0.019061 | 0.010318 |

### Issue3 Summary:

1. For all, the predicting performs are totally bad.

- But still, DecisionTreeClassifier and RandomForestClassifier have high score in training, what's going on?
  - Notice that: the test time if much more less than train time. The possible reason is that the data set has a simple structure so that all predict with same result(*see the deployment of Issue2*: no matter how I change the input data, it all shows the same result.)

2. Pay attention to that the GradientBoostingClassifier() is not suitable for multi-class problem.
   So I used **MultiOutputClassifier(GradientBoostingClassifier())**

## Additional Issue: Neural Network for regeression

| | Object | Description |
|---|---|---|
| **Data Set** | all model_dataset | |
| **Target** | amount_total (float) | The total amount of money paid by a costomer |
| **Features** | age | basic info about customer |
| | income | basic info about customer |
| | member_days | basic info about customer |
| | gender_ | basic info about customer (3 kinds of 0-1 variables) |
| | reward | bounded with offer |
| | difficulty | bounded with offer |
| | duration | bounded with offer |
| | email | bounded with offer |
| | mobile | bounded with offer |
| | social | bounded with offer |
| | web | bounded with offer |
| | transaction_cnt | count of all transactions for a customer |
| | offer_received_cnt | count of received offers for a customer |
| | group_effctive_offer | Label of |

| | | |
|---|---|---|
| | | group(effctive_offer) |
| | group_no_care | Label of group(no_care) |
| | group_tried | Label of group(tried) |

Model training result:

```
epoch:1/20.. Training Loss: 20550.215.. Test Loss: 16089.338.. Time Cost: 85.232s..
epoch:2/20.. Training Loss: 18629.431.. Test Loss: 15645.789.. Time Cost: 108.798s..
epoch:3/20.. Training Loss: 18059.719.. Test Loss: 15397.492.. Time Cost: 88.476s..
epoch:4/20.. Training Loss: 18591.784.. Test Loss: 17092.785.. Time Cost: 84.721s..
epoch:5/20.. Training Loss: 17867.469.. Test Loss: 16893.158.. Time Cost: 106.455s..
epoch:6/20.. Training Loss: 17776.402.. Test Loss: 16882.059.. Time Cost: 88.753s..
epoch:7/20.. Training Loss: 17767.603.. Test Loss: 16883.391.. Time Cost: 87.397s..
epoch:8/20.. Training Loss: 17766.653.. Test Loss: 16884.016.. Time Cost: 97.170s..
epoch:9/20.. Training Loss: 17766.521.. Test Loss: 16884.375.. Time Cost: 93.455s..
epoch:10/20.. Training Loss: 17766.495.. Test Loss: 16884.461.. Time Cost: 150.855
s..
epoch:11/20.. Training Loss: 17766.488.. Test Loss: 16884.436.. Time Cost: 90.398s..
epoch:12/20.. Training Loss: 17766.486.. Test Loss: 16884.402.. Time Cost: 75.996s..
epoch:13/20.. Training Loss: 17766.486.. Test Loss: 16884.424.. Time Cost: 70.614s..
epoch:14/20.. Training Loss: 17766.486.. Test Loss: 16884.420.. Time Cost: 71.088s..
epoch:15/20.. Training Loss: 17766.486.. Test Loss: 16884.416.. Time Cost: 70.461s..
epoch:16/20.. Training Loss: 17766.486.. Test Loss: 16884.414.. Time Cost: 84.942s..
epoch:17/20.. Training Loss: 17766.486.. Test Loss: 16884.414.. Time Cost: 93.021s..
epoch:18/20.. Training Loss: 17766.486.. Test Loss: 16884.414.. Time Cost: 82.828s..
epoch:19/20.. Training Loss: 17766.486.. Test Loss: 16884.414.. Time Cost: 100.545
s..
epoch:20/20.. Training Loss: 17766.486.. Test Loss: 16884.414.. Time Cost: 94.549s..
```

### Additional Issue summary:

This experiment for regression analysing seems useless because of the stagnant and huge training loss.

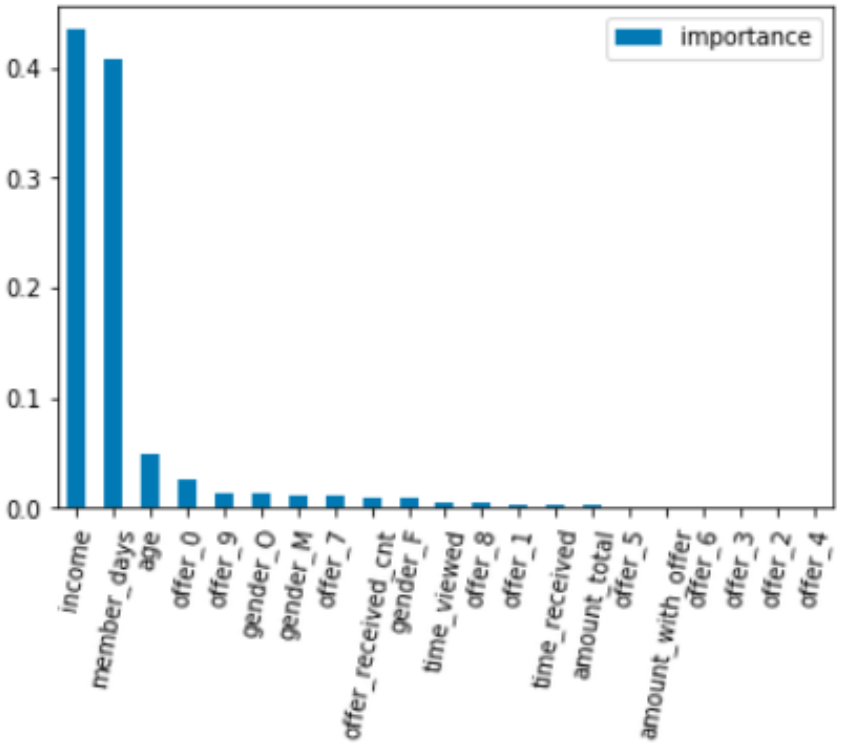
---

# Results: deployment of Issue2

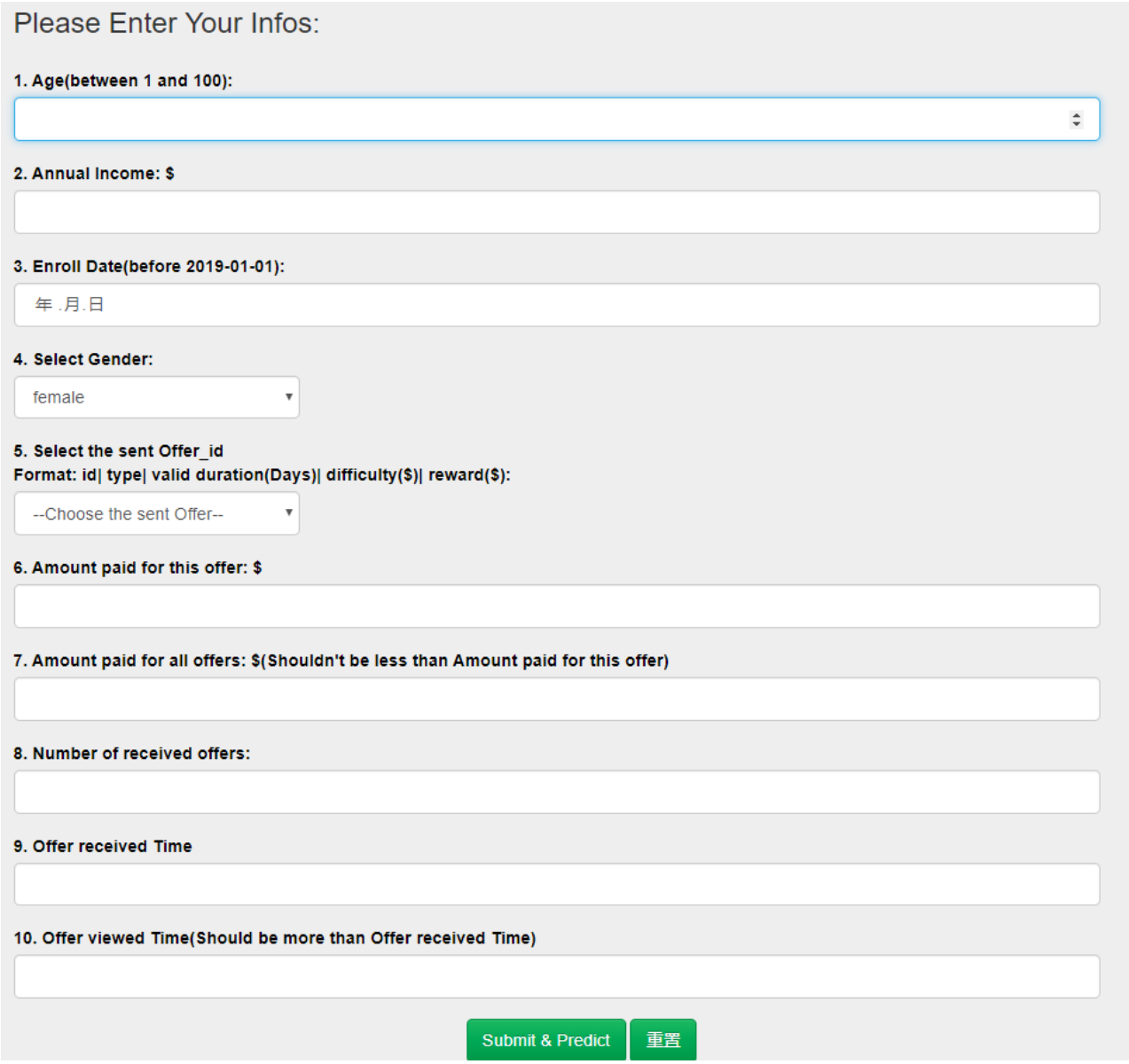
Offer is already sent to a customer, is this offer effective?

Classifier: **GradientBoostingClassifier**

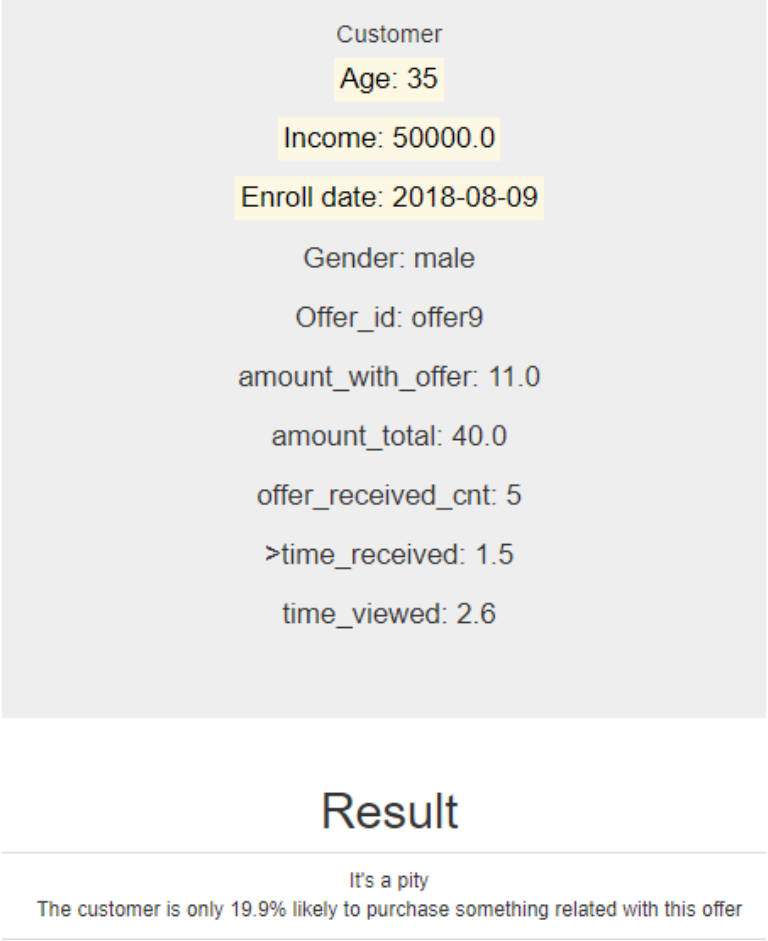
Features importances order:




Deployment – input the data:




Deployment – result:




# Conclusion

1. Two analysis methods: heuristic exploration & model building
    a. The model method fits not good at the reality: for example the deployment of Issue2, when I change the input data, the result seems always the same.
    b. But the heuristic exploration makes more sense.
    c. In this case we could try unsupervised Machine Learning method like Cluster. See References[6]
2. About Data Set
    a. Maybe when the amount of data is big enough, we could get some resonable founds by using the supervised Machine Learning method.
    b. Besides, there is no features of customer id, the customer exists in the form of the segment group, maybe when the transactions of an unique customer more frequently occurs, there would be some patterns of the consuming behavior.
3. About the Segment

a. Here I use the information of age and income. We can also use other method to segment the customers, e.g. age and gender.

---

For more details of this project, you could refer to my Github Repository
I would like to thank Udacity & Starbucks for all the supports, especially for teaching assistant.

## References

[1]Create dummies from a column with multiple values in pandas
[2]Starbucks Capstone Challenge: Using Starbucks app user data to predict effective offers
[3]Starbucks Promotion Optimization
[4]generations-and-age
[5]single taxable income
[6]Investigating Starbucks Customers Segmentation using Unsupervised Machine Learning