Segmenting the Starbucks Customers with unpserpvised learning algorithm:

Introduction:

The data contains the 3 sets below:

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

- portfolio.json

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)

- profile.json

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

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

I wanted to segment Starbucks customers based on their demographics and spending behavior provided through the offers data available from Udacity.

The data contains 3 sets of 'portfolio', 'profile' and 'transcripts'. It represents 17,000 customers on the Starbucks rewards mobile app where Starbucks sends an offer every few days.

The goal is to use this data to build up a more comprehensive customer profile that will allow us to segment these customers.

The task was not easy because I chose this dataset based on my interest in the topic and the company rather what is available on the table, the data required lots of preprocessing and feature engineering which ate most of the time invested leaving little time to analyze and try to pick up trends before the project deadline

I created and played around the data and reached that the customers can be broken down to 8 clusters using k-means clustering.

Here are the offer

1. A discount offer (discount)

2. A buy-one-get-one-free offer (BOGO)

3. Merely an advertisement for a drink (info)

Process I used to created this project:

1. Data cleaning
2. Merging tables and then breaking them
3. Feature engineering
4. Inputting missing data
5. Feature scaling
6. PCA
7. Segmentation & Clustering

1. **Cleaning Data:**

Although the data looks innocent with very little null values, after some time you will notice that there are many things that needs work around. First, you have the below where that re records with missing values in gender and income or outlandish values in the age column:

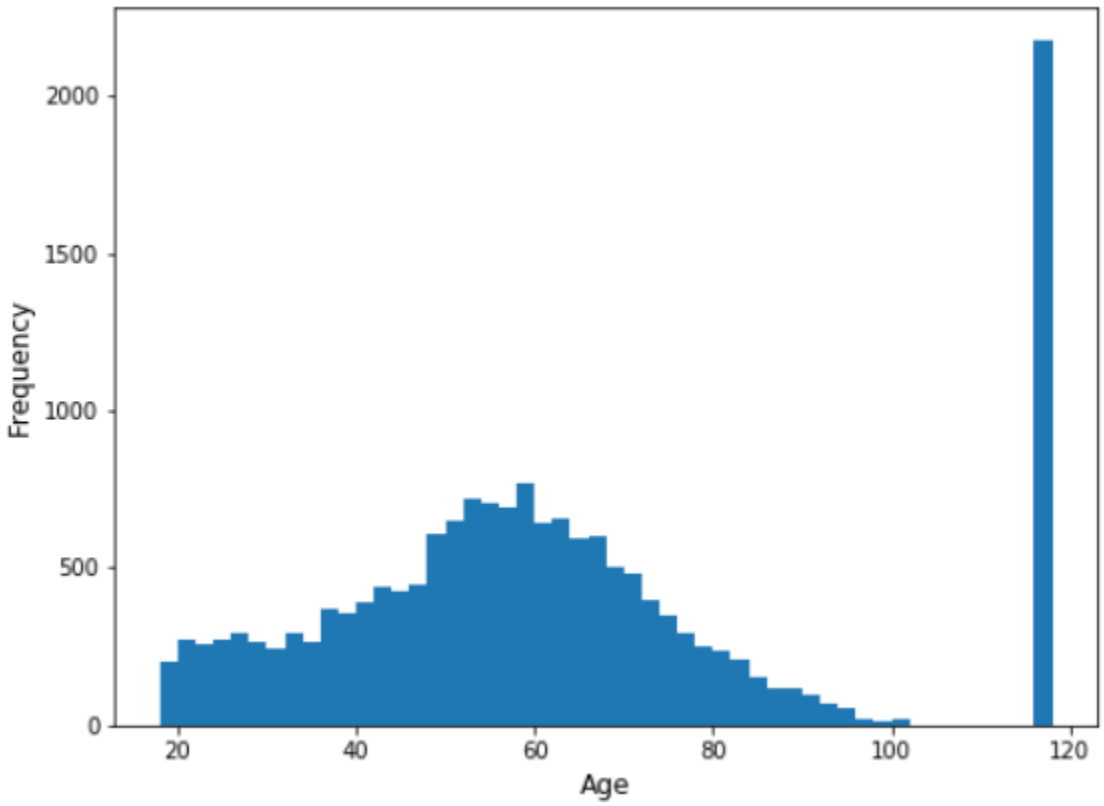
```
profile[profile.gender.isnull()].head()
```

	age	became_member_on	gender	id	income
0	118	20170212	None	68be06ca386d4c31939f3a4f0e3dd783	NaN
2	118	20180712	None	38fe809add3b4fcf9315a9694bb96ff5	NaN
4	118	20170804	None	a03223e636434f42ac4c3df47e8bac43	NaN
6	118	20170925	None	8ec6ce2a7e7949b1bf142def7d0e0586	NaN
7	118	20171002	None	68617ca6246f4fbc85e91a2a49552598	NaN

Same records have the missing age and gender

Here I think I was about to drop these records but then decided to keep them just to see if this data was junk, but upon merging the profile dataframe with the event table, you will see that these people decided not to fill these fields and it seems the age is from a predefined year upon sign up.

Here how the age is distributed:



Here are the missing values in profile with 2,175 records missing.

```
profile.isnull().sum()
```

```
age                    0
became_member_on       0
gender              2175
id                     0
income              2175
dtype: int64
```

We have more than 2000 records missing in income and age, and 2000 members above 100 years old, are these the same records?

Then there was the portfolio table where we had information about the offers available on the transcripts, but it had a list where the channels used to distribute the offer. Here I looped through this series and split the series into 4 different columns each with

channel using one hot encoder where if the channel was available for the offer 1 is included else a zero.

Portfolio looked before the cleaning:

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

After the splitting the channels column:

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

The big surprise was with the transcript dataset, at the beginning I though the only issue was to deal with the dictionary in the value column as visible below:

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

After looping through the dictionary I created 2 new columns I now have offer type and offer value:

	event	person	time	offer_type	offer_value
0	offer received	78afa995795e4d85b5d9ceeca43f5fef	0	offer id	9b98b8c7a33c4b65b9aebfe6a799e6d9
1	offer received	a03223e636434f42ac4c3df47e8bac43	0	offer id	0b1e1539f2cc45b7b9fa7c272da2e1d7
2	offer received	e2127556f4f64592b11af22de27a7932	0	offer id	2906b810c7d4411798c6938adc9daaa5
3	offer received	8ec6ce2a7e7949b1bf142def7d0e0586	0	offer id	fafdcd668e3743c1bb461111dcafc2a4
4	offer received	68617ca6246f4fbc85e91a2a49552598	0	offer id	4d5c57ea9a6940dd891ad53e9dbe8da0

Then the surprise the type was not only offers but also included “amount”

```
transcript.offer_type.value_counts()
```

```
offer_id    167581
amount      138953
Name: offer_type, dtype: int64
```

So these rows represent 2 things, the offers a customer receives and the value of the transactions.

So I split the transcript into 2 tables, one that holds the amount and one for the offer.

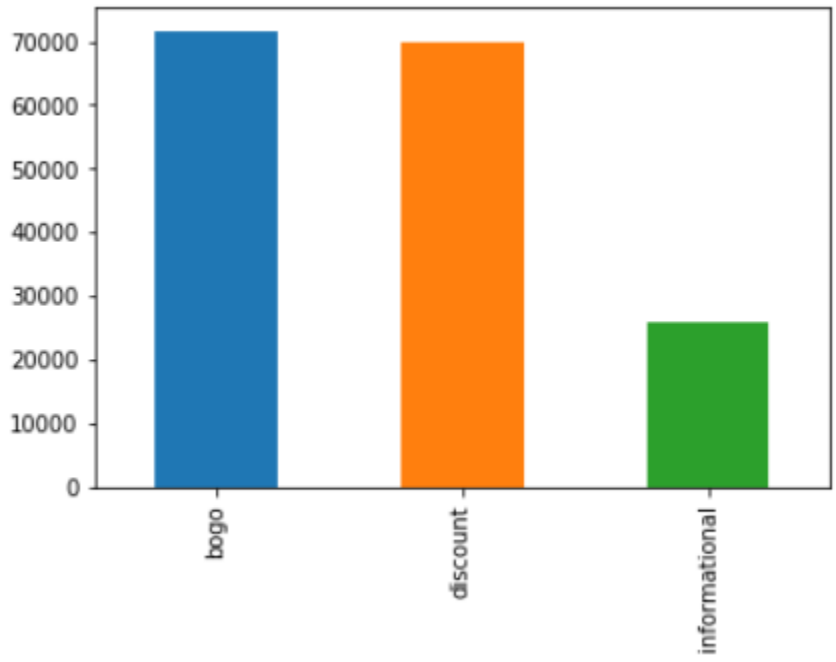
```
transcript_amount.head()
```

	user_id	time_spent	amount
12654	02c083884c7d45b39cc68e1314fec56c	0	0.83
12657	9fa9ae8f57894cc9a3b8a9bbe0fc1b2f	0	34.56
12659	54890f68699049c2a04d415abc25e717	0	13.23
12670	b2f1cd155b864803ad8334cdf13c4bd2	0	19.51
12671	fe97aa22dd3e48c8b143116a8403dd52	0	18.97

```
transcript_offers.head()
```

	event	user_id	time	offer_id
0	offer received	78afa995795e4d85b5d9ceeca43f5fef	0	9b98b8c7a33c4b65b9aebfe6a799e6d9
1	offer received	a03223e636434f42ac4c3df47e8bac43	0	0b1e1539f2cc45b7b9fa7c272da2e1d7
2	offer received	e2127556f4f64592b11af22de27a7932	0	2906b810c7d4411798c6938adc9daaa5
3	offer received	8ec6ce2a7e7949b1bf142def7d0e0586	0	fafdcd668e3743c1bb461111dcafc2a4
4	offer received	68617ca6246f4fbc85e91a2a49552598	0	4d5c57ea9a6940dd891ad53e9dbe8da0

Here are the three types of offers sent to the customers:



2. Merging and breaking the 4 tables:

Here I had several challenges, since I wanted to add as much as possible data from the offers to the client ID. First I merged the profile and the offers these customers received based on the offer_id then I broke it to 3 dataframes:

- Offers Received

]:

user_id	time_received	offer_id	age	gender	income	join_year	join_month	join_day	difficulty	duration	offer_type	reward	email	mobile	social	web
!afa995795e4d85b5d9ceeca43f5fef	0	9b98b8c7a33c4b65b9aebfe6a799e6d9	75	F	100000.0	2017	5	9	5	7	bogo	5	1	1	0	1
)223e636434f42ac4c3df47e8bac43	0	0b1e1539f2cc45b7b9fa7c272da2e1d7	118	None	NaN	2017	8	4	20	10	discount	5	1	0	0	1
l27556f4f64592b11af22de27a7932	0	2906b810c7d4411798c6938adc9daaa5	68	M	70000.0	2018	4	26	10	7	discount	2	1	1	0	1
6ce2a7e7949b1bf142def7d0e0586	0	fafdcd668e3743c1bb461111dcafc2a4	118	None	NaN	2017	9	25	10	10	discount	2	1	1	1	1
517ca6246f4fbc85e91a2a49552598	0	4d5c57ea9a6940dd891ad53e9dbe8da0	118	None	NaN	2017	10	2	10	5	bogo	10	1	1	1	1

- Offers Viewed

	user_id	time_viewed	offer_id	age	gender	income	join_year	join_month	join_day	difficulty	duration	offer_type	reward	email	mobile
12650	389bc3fa690240e798340f5a15918d5c	0	f19421c1d4aa40978ebb69ca19b0e20d	65	M	53000.0	2018	2	9	5	5	bogo	5	1	1
12651	d1ede868e29245ea91818a903fec04c6	0	5a8bc65990b245e5a138643cd4eb9837	53	O	52000.0	2017	9	16	0	3	informational	0	1	1
12652	102e9454054946fda62242d2e176fdce	0	4d5c57ea9a6940dd891ad53e9dbe8da0	69	F	57000.0	2016	8	14	10	5	bogo	10	1	1
12653	02c083884c7d45b39cc68e1314fec56c	0	ae264e3637204a6fb9bb56bc8210ddfd	20	F	30000.0	2016	7	11	10	7	bogo	10	1	1
12654	be8a5d1981a2458d90b255ddc7e0d174	0	5a8bc65990b245e5a138643cd4eb9837	39	M	51000.0	2014	5	27	0	3	informational	0	1	1

- Offers Completed

	user_id	time_completed	offer_id	age	gender	income	join_year	join_month	join_day	difficulty	duration	offer_type	reward	email	mobile
12656	9fa9ae8f57894cc9a3b8a9bbe0fc1b2f	0	2906b810c7d4411798c6938adc9daaa5	42	M	96000.0	2016	1	17	10	7	discount	2	1	1
12667	fe97aa22dd3e48c8b143116a8403dd52	0	fafdcd668e3743c1bb461111dcafc2a4	39	F	67000.0	2017	12	17	10	10	discount	2	1	1
12673	629fc02d56414d91bca360decdfa9288	0	9b98b8c7a33c4b65b9aebfe6a799e6d9	52	M	72000.0	2018	6	5	5	7	bogo	5	1	1
12683	676506bad68e4161b9bbaffeb039626b	0	ae264e3637204a6fb9bb56bc8210ddfd	37	M	92000.0	2017	5	15	10	7	bogo	10	1	1
12687	8f7dd3b2afe14c078eb4f6e6fe4ba97d	0	4d5c57ea9a6940dd891ad53e9dbe8da0	48	M	62000.0	2015	9	3	10	5	bogo	10	1	1

This helped me have a time of the event in separate column then I merged them together and this was an amazing move as I have now time each offer was received and viewed and if completed.

offer_id	age	gender	income	join_year	join_month	join_day	difficulty	duration	offer_type	reward	email	mobile	social	web	time_received	time_viewed	time_completed
9b98b8c7a33c4b65b9aebfe6a799e6d9	75	F	100000.0	2017	5	9	5	7	bogo	5	1	1	0	1	0	6.0	132.0
0b1e1539f2cc45b7b9fa7c272da2e1d7	118	None	NaN	2017	8	4	20	10	discount	5	1	0	0	1	0	6.0	NaN
0b1e1539f2cc45b7b9fa7c272da2e1d7	118	None	NaN	2017	8	4	20	10	discount	5	1	0	0	1	0	624.0	NaN
2906b810c7d4411798c6938adc9daaa5	68	M	70000.0	2018	4	26	10	7	discount	2	1	1	0	1	0	18.0	NaN
fafdcd668e3743c1bb461111dcafc2a4	118	None	NaN	2017	9	25	10	10	discount	2	1	1	1	1	0	12.0	NaN
fafdcd668e3743c1bb461111dcafc2a4	118	None	NaN	2017	9	25	10	10	discount	2	1	1	1	1	0	522.0	NaN
4d5c57ea9a6940dd891ad53e9dbe8da0	118	None	NaN	2017	10	2	10	5	bogo	10	1	1	1	1	0	84.0	NaN
f19421c1d4aa40978ebb69ca19b0e20d	65	M	53000.0	2018	2	9	5	5	bogo	5	1	1	1	1	0	0.0	60.0
f19421c1d4aa40978ebb69ca19b0e20d	65	M	53000.0	2018	2	9	5	5	bogo	5	1	1	1	1	0	0.0	600.0
f19421c1d4aa40978ebb69ca19b0e20d	65	M	53000.0	2018	2	9	5	5	bogo	5	1	1	1	1	0	504.0	60.0

Here I know there are duplicate records where the offer id and user id coupled are dropped.

3. Feature engineering:

I added the validity of the offers by subtracting the time received from the offer duration, and here I did a very opinionated conditioning when I decided what offers are worth pursuing:

Successful Offers:

For an offer to be successful, it has to be viewed and completed before it expires, this is the only way we are sure that an offer caused conversion

Effective offer:

- Viewed the offer before expiry and completed it after expiry

- Viewed the offer after expiry and completed it but it is important that the customer completed it after viewing it

All the rest are flagged as failed offers, and a dataframe is created to only include the successful and effective offers as per my standards

Here is the most confusing of all, since we have the offer ID and user ID but in the amount table, there are no offer ID in the newly created dataframe, so I merged them then conditioned the below to tray to match the amount spent to an offer:

In this dataframe, we have for each and every user with offer, all the amounts spent but this does not mean that each of these offers has this value, to get which of these offers might have a monetary value we need to condition on the below:

- Spending time shall be less than the time completed and after the offer has been viewed

- Spending time shall be less than the expiry and after the offer has been viewed

Now I have the below:

Since our goal is to segment the customers we need to do the below:

- Extract as much variables as possible from the offers

- Have the dataframe where each line is a customer

- Fill the missing values in age and income

- Conduct an unpservised learning customer segmentation

Now, I grouped the offers and users then I conditioned to create a new dataframe for each offer where I created 2 new features:

- Successful offer ratio
- Effective offer ratio

Now I have for each offer stats separate then I merged all for a final table where each line is a customer and each offer has a column:

	age	gender	user_id	income	join_year	join_month	join_day	num_0_succ	num_0_effect	total_0_spent	...	offer_6_succ_ratio
0	118	None	68be06ca386d4c31939f3a4f0e3dd783	NaN	2017	2	12	0.0	0.0	0.0	...	0.0
1	55	F	0610b486422d4921ae7d2bf64640c50b	112000.0	2017	7	15	0.0	0.0	0.0	...	0.0
2	118	None	38fe809add3b4fcf9315a9694bb96ff5	NaN	2018	7	12	0.0	0.0	0.0	...	0.0
3	75	F	78afa995795e4d85b5d9ceeca43f5fef	100000.0	2017	5	9	0.0	0.0	0.0	...	0.0
4	118	None	a03223e636434f42ac4c3df47e8bac43	NaN	2017	8	4	0.0	0.0	0.0	...	0.0

4- Imputing missing files:

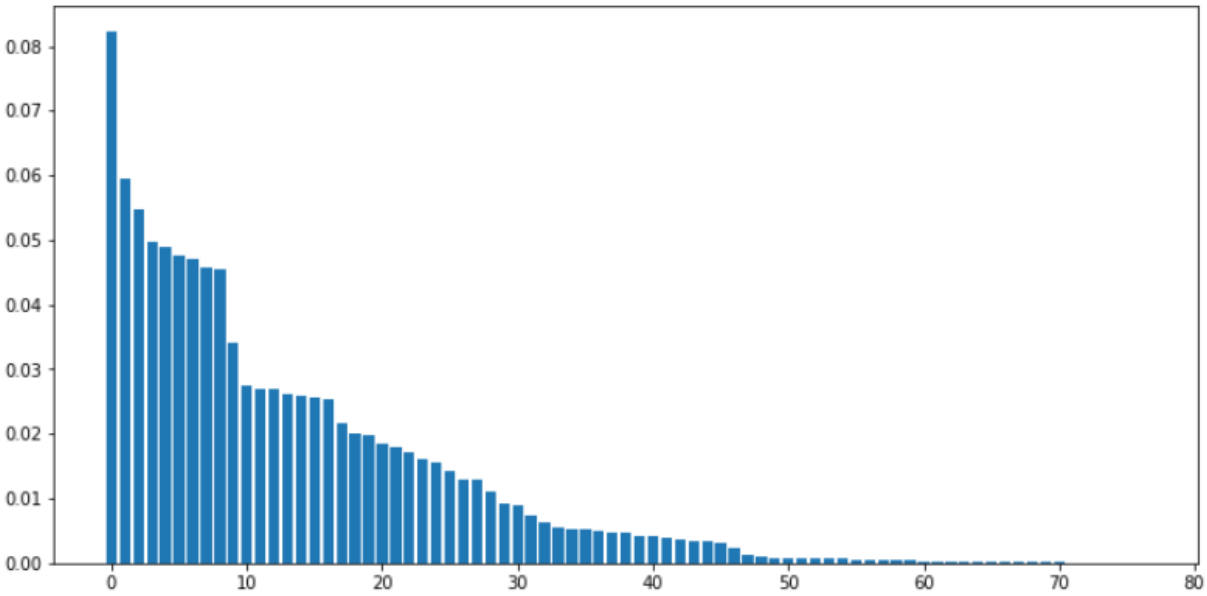
First I created a dataframe without the records with missing values, then I used the full dataframe to train and test the model, then I used a Random Forest Regressor to fill the missing values in the income and age, while I used Random Forest Classifier to predict the gender. Then I concatenated the new full dataframe to the original to have a table with no null values that might drop us an error.

5- Applying Standard Scaler:

I used the Standard scaler to scale all the values in the final dataframe as the PCA can suffer from extreme value like income when doing dimensionality reduction

6- PCS

First I performed a full scale PCA to see the number of components that can explain 90% of the features:



Then I have created a function where the major features contributing to the principal component. For example the biggest principal component PC-1, has the below variables explaining it:

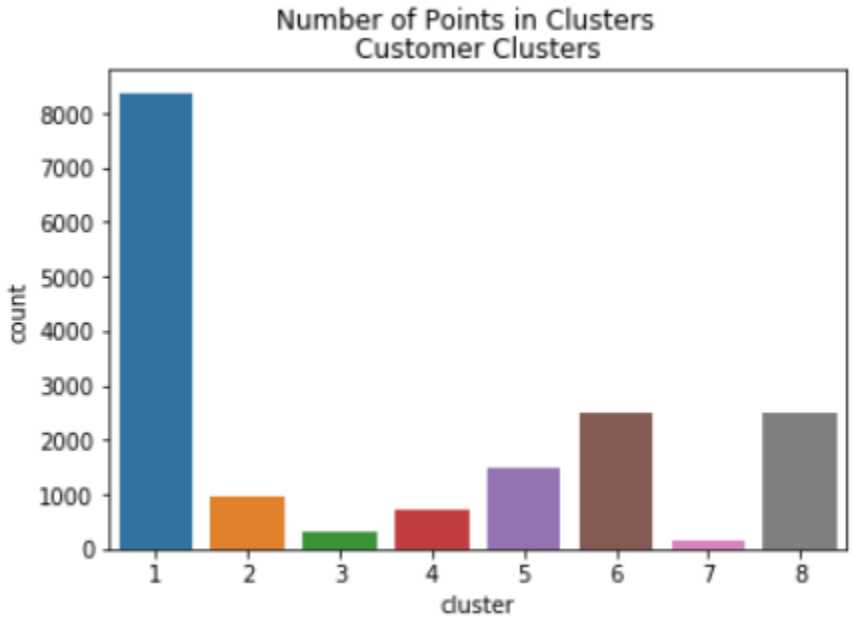
	PC-1
total_num_offers	0.331588
total_spent	0.277575
total_num_effect_offers	0.272600
num_offer_4_rec	0.175095
total_num_effective_offers	0.167677

Here we know that this principle component is highly influenced by the offers data including what the number of offers a customer received, total spent, number of effective…

7- Segmentation & Clustering:

I used the k-means clustering which is the simplest and most straight forward of the unsupervised learning algorithms and I ended up with 8 clusters at best after training and testing using cluster score.

Then I placed back the original customer profile provided in the dataset and I segmented the customers as below:



When doing inverse transformation, I have the below features that highly influenced the biggest over represented cluster 1:

	features	Values
4	join_year	0.159805
1	gender	0.048524
6	join_day	0.001822
2	user_id	0.000338
59	avg_response_6_diff	-0.015474

We know that the customer join date with the gender are highly important in grouping a customer.