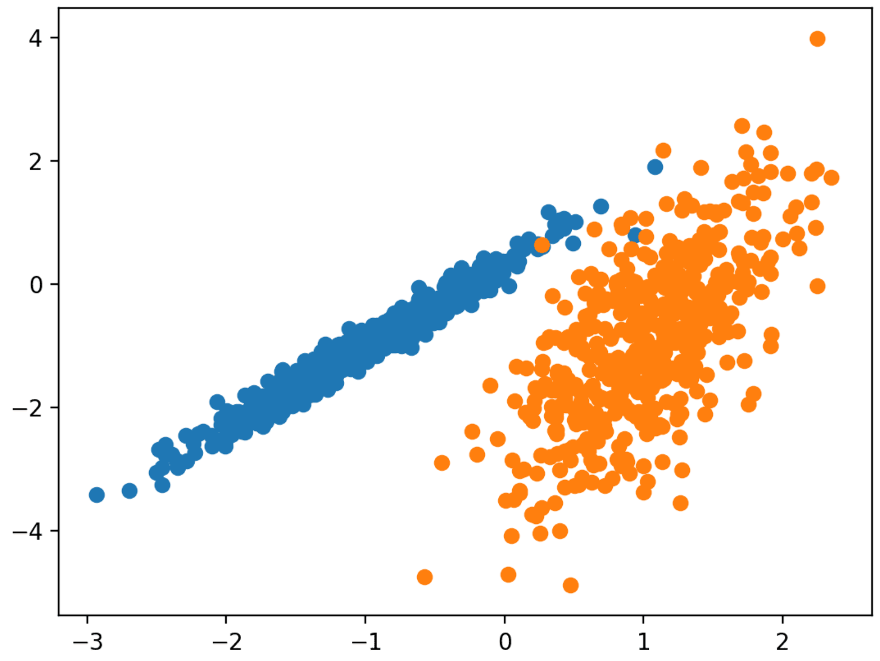
Customer Segmentation with Starbucks Dataset



Project’s domain background

Due to the recent advancements in technology, the world is causing us to adapt extremely fast to the changes. It is thanks to the technology that I am able to write this proposal and email it to a mentor to be revised. With this being said, it is no secret that large enterprises and multi millionaire companies use this amazing invention called “the internet” to reach a larger audience and grow their business.

Starbucks is one of these many businesses that take advantage of the internet to obtain new customers daily. The following are some of the ways that Starbucks uses technology to reach new customers.

1. Free Wi-Fi: What a better way to attract buyers and tempt them to buy your product than giving them free access to Wi-Fi almost any time of the day?
2. Mobile or online purchase: Studies show that 11% of Starbucks sales come from mobile purchases.
3. Discounts and Offers: Like any other store, Starbucks sends their customers offers which can be very tempting but rewarding. This therefore helps to create loyal customers and increase sales.

Therefore, Starbucks is one of the first companies to take advantage of technology as a marketing strategy. “[…] In many technology-related industries, competition is intense and can often be the reason why a startup is not able to be profitable. However, competition cuts across the board and can contribute to potential business failure, regardless of the sector.”[1] With that, half of small businesses survive beyond five years. Additionally, one third of these businesses make it to 10 years.

With that being said, marketing is crucial for understanding how to meet customer's needs. This of course keeping in mind that customers vary in characteristics like flavor preference, behaviors, traditions, economic situation, etc. This is where artificial intelligence and machine learning comes in. With the recent developments in the area, machine learning has been able to use marketing information to detect behavior patterns in customers and segmenting groups of people based on demographic information.

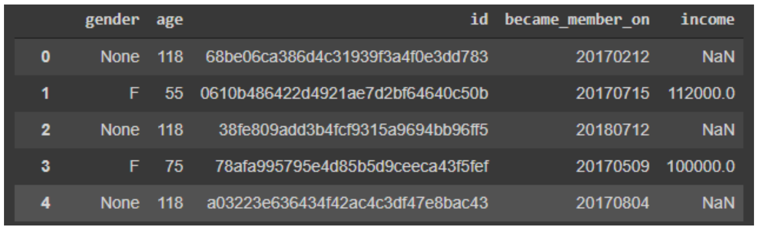
For this reason, this project will study the behaviors and reactions of different Starbucks customers. We will determine which customers are actually influenced by offers and which are not. This also tells us which customers are considered “loyal”. Finally, the end goal is to predict how a demographic group will react to an offer.

Datasets and inputs:

The datasets used for this project are the following:

profile.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)

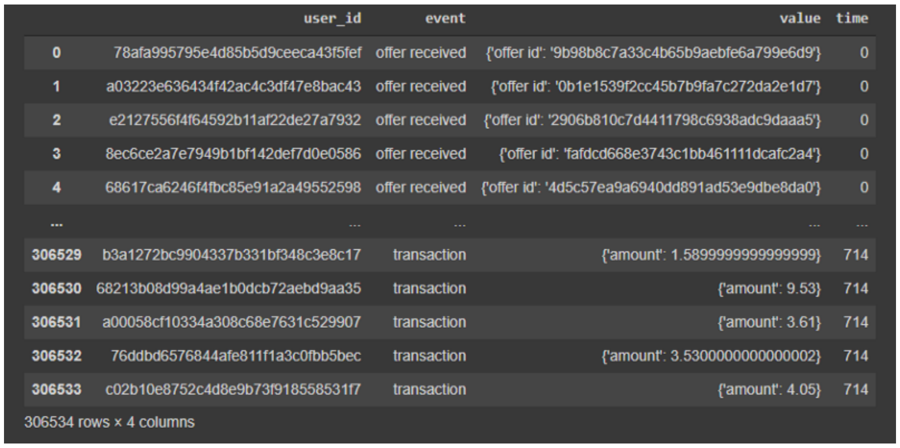


	gender	age	id	became_member_on	income
0	None	118	68be06ca386d4c31939f3a4f0e3dd783	20170212	NaN
1	F	55	0610b486422d4921ae7d2bf64640c50b	20170715	112000.0
2	None	118	38fe809add3b4fcf9315a9694bb96ff5	20180712	NaN
3	F	75	78afa995795e4d85b5d9ceeca43f5fef	20170509	100000.0
4	None	118	a03223e636434f42ac4c3df47e8bac43	20170804	NaN

profile.json

Portfolio.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



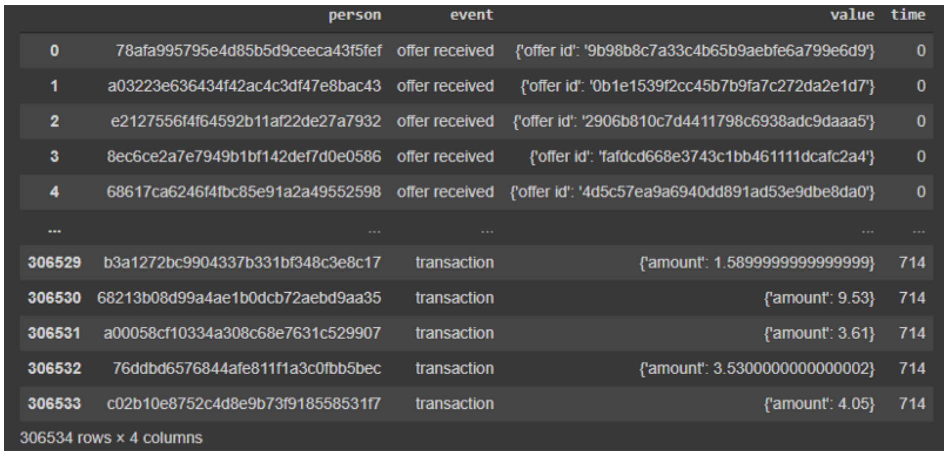
	user_id	event	value	time
0	78afa995795e4d85b5d9ceeca43f5fef	offer received	{'offer id': '9b98b8c7a33c4b65b9aebfe6a799e6d9'}	0
1	a03223e636434f42ac4c3df47e8bac43	offer received	{'offer id': '0b1e1539f2cc45b7b9fa7c272da2e1d7'}	0
2	e2127556f4f64592b11af22de27a7932	offer received	{'offer id': '2906b810c7d4411798c6938adc9daaa5'}	0
3	8ec6ce2a7e7949b1bf142def7d0e0586	offer received	{'offer id': 'fafdcd668e3743c1bb461111dcafc2a4'}	0
4	68617ca6246f4fbc85e91a2a49552598	offer received	{'offer id': '4d5c57ea9a6940dd891ad53e9dbe8da0'}	0
...	...	...	...	...
306529	b3a1272bc9904337b331bf348c3e8c17	transaction	{'amount': 1.5899999999999999}	714
306530	68213b08d99a4ae1b0dcb72aebd9aa35	transaction	{'amount': 9.53}	714
306531	a00058cf10334a308c68e7631c529907	transaction	{'amount': 3.61}	714
306532	76ddbd6576844afe811f1a3c0fbb5bec	transaction	{'amount': 3.5300000000000002}	714
306533	c02b10e8752c4d8e9b73f918558531f7	transaction	{'amount': 4.05}	714

306534 rows × 4 columns

portfolio.json

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



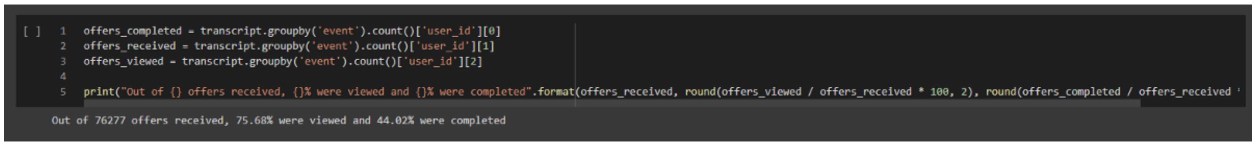
	person	event	value	time
0	78afa995795e4d85b5d9ceeca43f5fef	offer received	{'offer id': '9b98b8c7a33c4b65b9aebfe6a799e6d9'}	0
1	a03223e636434f42ac4c3df47e8bac43	offer received	{'offer id': '0b1e1539f2cc45b7b9fa7c272da2e1d7'}	0
2	e2127556f4f64592b11af22de27a7932	offer received	{'offer id': '2906b810c7d4411798c6938adc9daaa5'}	0
3	8ec6ce2a7e7949b1bf142def7d0e0586	offer received	{'offer id': 'fafdcd668e3743c1bb461111dcafc2a4'}	0
4	68617ca6246f4fbc85e91a2a49552598	offer received	{'offer id': '4d5c57ea9a6940dd891ad53e9dbe8da0'}	0
...	...	...	...	...
306529	b3a1272bc9904337b331bf348c3e8c17	transaction	{'amount': 1.5899999999999999}	714
306530	68213b08d99a4ae1b0dcb72aebd9aa35	transaction	{'amount': 9.53}	714
306531	a00058cf10334a308c68e7631c529907	transaction	{'amount': 3.61}	714
306532	76ddbd6576844afe811f1a3c0fbb5bec	transaction	{'amount': 3.5300000000000002}	714
306533	c02b10e8752c4d8e9b73f918558531f7	transaction	{'amount': 4.05}	714

306534 rows × 4 columns

transcript.json

Into the fun part

So first things first, we start exploring each of the given datasets and quickly start making calculations to explain some of the numbers in the data. The first is to fid out the percentage of offers viewed to received and completed to received.



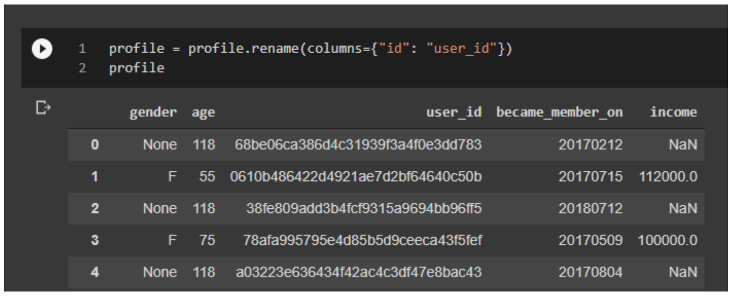
```
[ ]
offers_completed = transcript.groupby('event').count()['user_id'][0]
offers_received = transcript.groupby('event').count()['user_id'][1]
offers_viewed = transcript.groupby('event').count()['user_id'][2]

print("Out of {} offers received, {}% were viewed and {}% were completed".format(offers_received, round(offers_viewed / offers_received * 100, 2), round(offers_completed / offers_received
```

```
Out of 76277 offers received, 75.68% were viewed and 44.02% were completed
```

We can see that 75.68% of the offers received were viewed and 44.02% were completed.

Prepare the data for merging

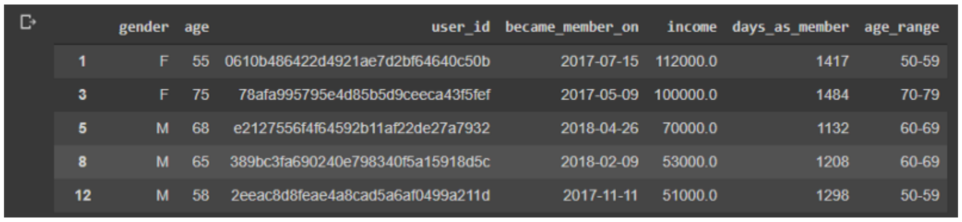
We prepare the data for merging by changing some column names and retrieving some data from object like values in the data frame. This looks something like this:



```python
profile = profile.rename(columns={"id": "user_id"})
profile
```

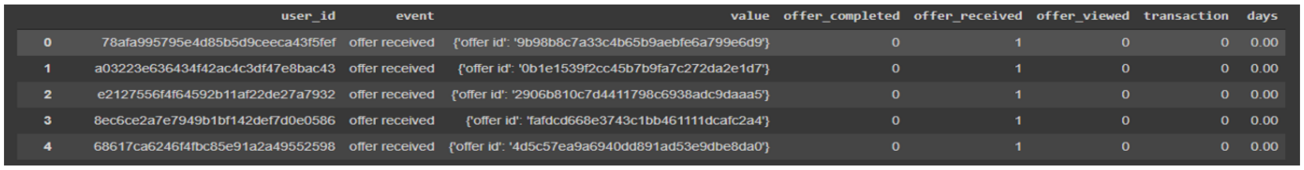
	gender	age	user_id	became_member_on	income
0	None	118	68be06ca386d4c31939f3a4f0e3dd783	20170212	NaN
1	F	55	0610b486422d4921ae7d2bf64640c50b	20170715	112000.0
2	None	118	38fe809add3b4fcf9315a9694bb96ff5	20180712	NaN
3	F	75	78afa995795e4d85b5d9ceeca43f5fef	20170509	100000.0
4	None	118	a03223e636434f42ac4c3df47e8bac43	20170804	NaN

For the portfolio dataset, we change the format of the date of becoming a member and add new columns to better visualize what we have ("days_as_memeber", "age_range").



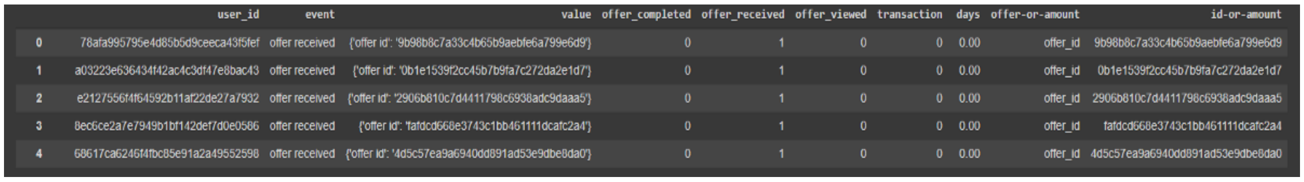
	gender	age	user_id	became_member_on	income	days_as_member	age_range
1	F	55	0610b486422d4921ae7d2bf64640c50b	2017-07-15	112000.0	1417	50-59
3	F	75	78afa995795e4d85b5d9ceeca43f5fef	2017-05-09	100000.0	1484	70-79
5	M	68	e2127556f4f64592b11af22de27a7932	2018-04-26	70000.0	1132	60-69
8	M	65	389bc3fa690240e798340f5a15918d5c	2018-02-09	53000.0	1208	60-69
12	M	58	2eeac8d8feae4a8cad5a6af0499a211d	2017-11-11	51000.0	1298	50-59

We then extract the data from the transcript by separating the orders made into 4 different groups: offer_completed: customer completed the offer, offer_received: customer received the offer, offer_viewed: customer viewed the offer, transaction: customer created a transaction which indicates a purchase. And that looks like this:



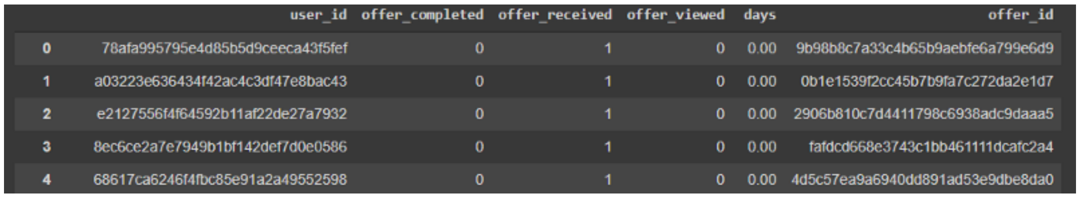
	user_id	event	value	offer_completed	offer_received	offer_viewed	transaction	days
0	78afa995795e4d85b5d9ceeca43f5fef	offer received	{'offer id': '9b98b8c7a33c4b65b9aebfe6a799e6d9'}	0	1	0	0	0.00
1	a03223e636434f42ac4c3df47e8bac43	offer received	{'offer id': '0b1e1539f2cc45b7b9fa7c272da2e1d7'}	0	1	0	0	0.00
2	e2127556f4f64592b11af22de27a7932	offer received	{'offer id': '2906b810c7d4411798c6938adc9daaa5'}	0	1	0	0	0.00
3	8ec6ce2a7e7949b1bf142def7d0e0586	offer received	{'offer id': 'fafdcd668e3743c1bb461111dcafc2a4'}	0	1	0	0	0.00
4	68617ca6246f4fbc85e91a2a49552598	offer received	{'offer id': '4d5c57ea9a6940dd891ad53e9dbe8da0'}	0	1	0	0	0.00

We also separate the value columns to indicate which values are transactions and which values are columns. This therefore tells us that there may be multiple forms of repeated offers and users in this dataset.



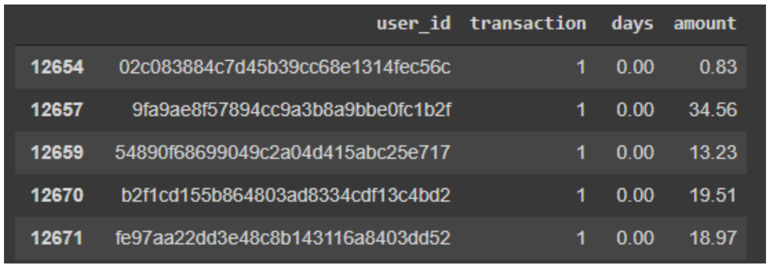
	user_id	event	value	offer_completed	offer_received	offer_viewed	transaction	days	offer-or-amount	id-or-amount
0	78afa995795e4d85b5d9ceeca43f5fef	offer received	{'offer id': '9b98b8c7a33c4b65b9aebfe6a799e6d9'}	0	1	0	0	0.00	offer_id	9b98b8c7a33c4b65b9aebfe6a799e6d9
1	a03223e636434f42ac4c3df47e8bac43	offer received	{'offer id': '0b1e1539f2cc45b7b9fa7c272da2e1d7'}	0	1	0	0	0.00	offer_id	0b1e1539f2cc45b7b9fa7c272da2e1d7
2	e2127556f4f64592b11af22de27a7932	offer received	{'offer id': '2906b810c7d4411798c6938adc9daaa5'}	0	1	0	0	0.00	offer_id	2906b810c7d4411798c6938adc9daaa5
3	8ec6ce2a7e7949b1bf142def7d0e0586	offer received	{'offer id': 'fafdcd668e3743c1bb461111dcafc2a4'}	0	1	0	0	0.00	offer_id	fafdcd668e3743c1bb461111dcafc2a4
4	68617ca6246f4fbc85e91a2a49552598	offer received	{'offer id': '4d5c57ea9a6940dd891ad53e9dbe8da0'}	0	1	0	0	0.00	offer_id	4d5c57ea9a6940dd891ad53e9dbe8da0

And then we separate this dataset into two different ones. One with offers and the other with transactions. I will show later why this was done. This is the dataset for offers:



	user_id	offer_completed	offer_received	offer_viewed	days	offer_id
0	78afa995795e4d85b5d9ceeca43f5fef	0	1	0	0.00	9b98b8c7a33c4b65b9aebfe6a799e6d9
1	a03223e636434f42ac4c3df47e8bac43	0	1	0	0.00	0b1e1539f2cc45b7b9fa7c272da2e1d7
2	e2127556f4f64592b11af22de27a7932	0	1	0	0.00	2906b810c7d4411798c6938adc9daaa5
3	8ec6ce2a7e7949b1bf142def7d0e0586	0	1	0	0.00	fafdcd668e3743c1bb461111dcafc2a4
4	68617ca6246f4fbc85e91a2a49552598	0	1	0	0.00	4d5c57ea9a6940dd891ad53e9dbe8da0

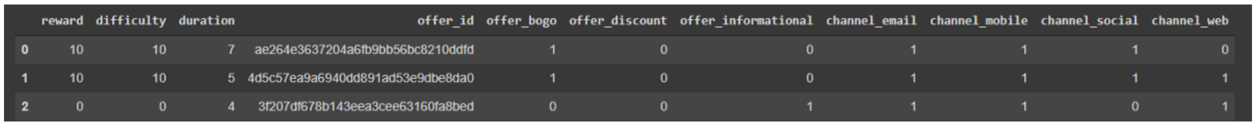
And this is the one for transactions:



	user_id	transaction	days	amount
12654	02c083884c7d45b39cc68e1314fec56c	1	0.00	0.83
12657	9fa9ae8f57894cc9a3b8a9bbe0fc1b2f	1	0.00	34.56
12659	54890f68699049c2a04d415abc25e717	1	0.00	13.23
12670	b2f1cd155b864803ad8334cdf13c4bd2	1	0.00	19.51
12671	fe97aa22dd3e48c8b143116a8403dd52	1	0.00	18.97

Extracting values from portfolio

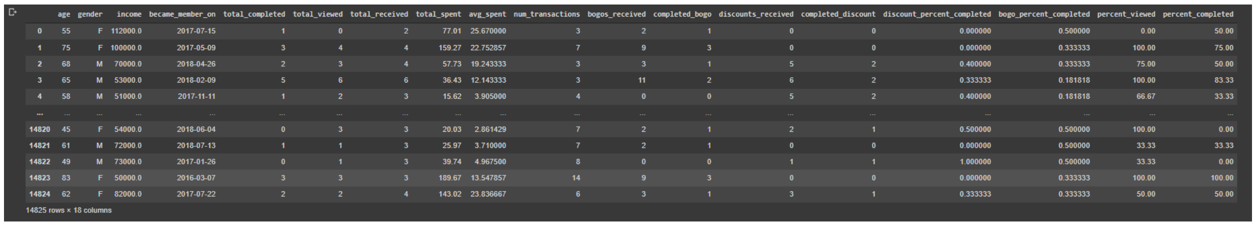
We also need to extract values from the portfolio dataset which has columns like channels holding other values. And this would look like this:



	reward	difficulty	duration	offer_id	offer_bogo	offer_discount	offer_informational	channel_email	channel_mobile	channel_social	channel_web
0	10	10	7	ae264e3637204a6fb9bb56bc8210ddfd	1	0	0	1	1	1	0
1	10	10	5	4d5c57ea9a6940dd891ad53e9dbe8da0	1	0	0	1	1	1	1
2	0	0	4	3f207df678b143eea3cee63160fa8bed	0	0	1	1	1	0	1

Merging

Now we go ahead and merge all of the data into one by iterating through profile and adding useful information about each customer into the dataset entry. If you want to know the whole process behind this data merge, you can checkout the code on my GitHub which is at the end of this article. Which turns out to be like this:

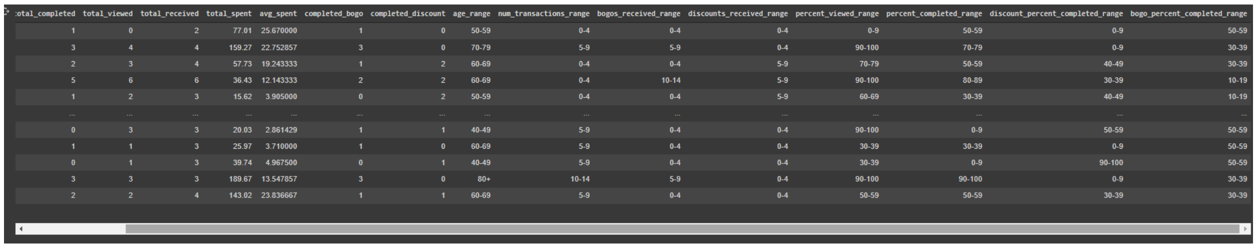


	age	gender	income	became_member_on	total_completed	total_viewed	total_received	total_spent	avg_spent	num_transactions	bogos_received	completed_bogo	discounts_received	completed_discount	discount_percent_completed	bogo_percent_completed	percent_viewed	percent_completed
0	55	F	112000.0	2017-07-15	1	0	2	77.01	25.670000	3	2	1	0	0	0.000000	0.500000	0.00	50.00
1	75	F	100000.0	2017-05-09	3	4	4	159.27	22.752857	7	9	3	0	0	0.000000	0.333333	100.00	75.00
2	68	M	70000.0	2018-04-26	2	3	4	57.73	19.243333	3	3	1	5	2	0.400000	0.333333	75.00	50.00
3	65	M	53000.0	2018-02-09	5	6	6	36.43	12.143333	3	11	2	6	2	0.333333	0.181818	100.00	83.33
4	58	M	51000.0	2017-11-11	1	2	3	15.62	3.905000	4	0	0	5	2	0.400000	0.181818	66.67	33.33
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
14820	45	F	54000.0	2018-06-04	0	3	3	20.03	2.861429	7	2	1	2	1	0.500000	0.500000	100.00	0.00
14821	61	M	72000.0	2018-07-13	1	1	3	25.97	3.710000	7	2	1	0	0	0.000000	0.500000	33.33	33.33
14822	49	M	73000.0	2017-01-26	0	1	3	39.74	4.967500	8	0	0	1	1	1.000000	0.500000	33.33	0.00
14823	83	F	50000.0	2016-03-07	3	3	3	189.67	13.547857	14	9	3	0	0	0.000000	0.333333	100.00	100.00
14824	62	F	82000.0	2017-07-22	2	2	4	143.02	23.836667	6	3	1	3	1	0.333333	0.333333	50.00	50.00

14825 rows × 18 columns

Data exploration

Now we will explore some of the characteristics of the previously generated data and determine which columns are useful and how we can categorize the columns to begin the process of customer segmentation. Categorizing the data helps making data clustering algorithms’ job easier.

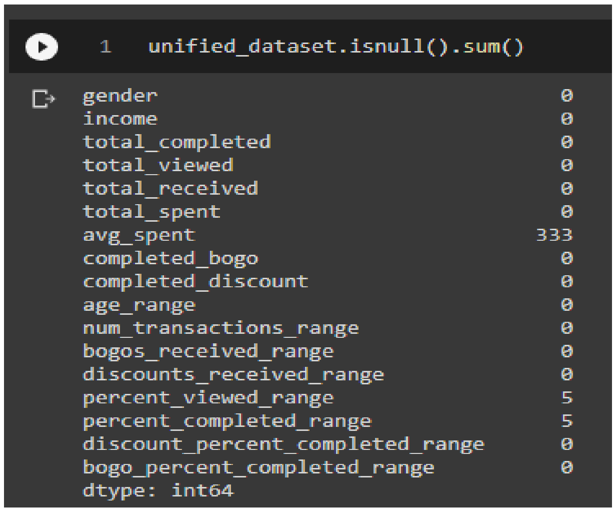


:otal_completed	total_viewed	total_received	total_spent	avg_spent	completed_bogo	completed_discount	age_range	num_transactions_range	bogos_received_range	discounts_received_range	percent_viewed_range	percent_completed_range	discount_percent_completed_range	bogo_percent_completed_range
1	0	2	77.01	25.670000	1	0	50-59	0-4	0-4	0-4	0-9	50-59	0-9	50-59
3	4	4	159.27	22.752857	3	0	70-79	5-9	5-9	0-4	90-100	70-79	0-9	30-39
2	3	4	57.73	19.243333	1	2	60-69	0-4	0-4	5-9	70-79	50-59	40-49	30-39
5	6	6	36.43	12.143333	2	2	60-69	0-4	10-14	5-9	90-100	80-89	30-39	10-19
1	2	3	15.62	3.905000	0	2	50-59	0-4	0-4	5-9	60-69	30-39	40-49	10-19
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
0	3	3	20.03	2.861429	1	1	40-49	5-9	0-4	0-4	90-100	0-9	50-59	50-59
1	1	3	25.97	3.710000	1	0	60-69	5-9	0-4	0-4	30-39	30-39	0-9	50-59
0	1	3	39.74	4.967500	0	1	40-49	5-9	0-4	0-4	30-39	0-9	90-100	50-59
3	3	3	189.67	13.547857	3	0	80+	10-14	5-9	0-4	90-100	90-100	0-9	30-39
2	2	4	143.02	23.836667	1	1	60-69	5-9	0-4	0-4	50-59	50-59	30-39	30-39

Getting rid of null values

Null values are really no use. All they do is distort the data and cause problems further down the line. For this, we go ahead and eliminate any row or entry that has any missing or null values.

333 null values were found in the “avg_spent” category. So we dropped those entries.



```
unified_dataset.isnull().sum()
```

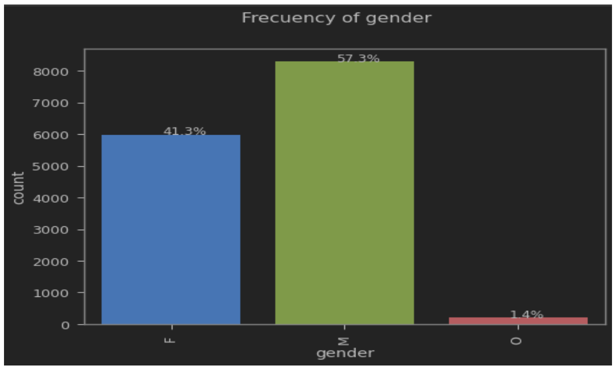
```
gender                              0
income                              0
total_completed                     0
total_viewed                        0
total_received                      0
total_spent                         0
avg_spent                         333
completed_bogo                      0
completed_discount                  0
age_range                           0
num_transactions_range              0
bogos_received_range                0
discounts_received_range            0
percent_viewed_range                5
percent_completed_range             5
discount_percent_completed_range    0
bogo_percent_completed_range        0
dtype: int64
```

Visualizing some categories

We will go ahead and visualize some of the key features of this dataset and draw conclusions based on obtained results.

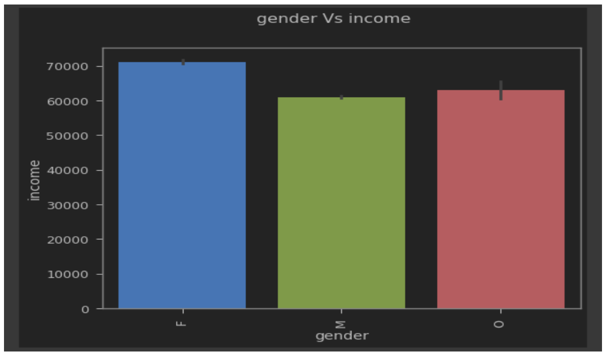
Visualize Gender:

This shows that there are more men than women in the customer base. 57.2% being men, 41.4% being women and 1.4% in the other category



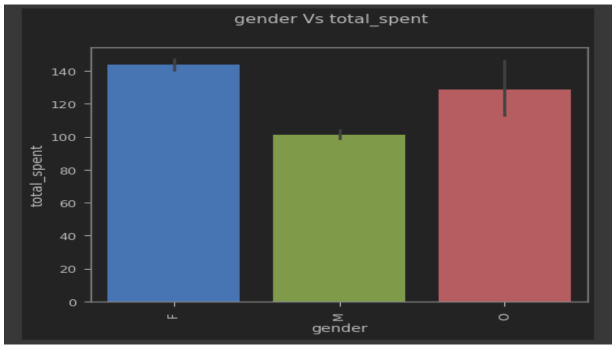
Visualize incomes:

Here we can notice that women in this dataset have higher incomes than men do.

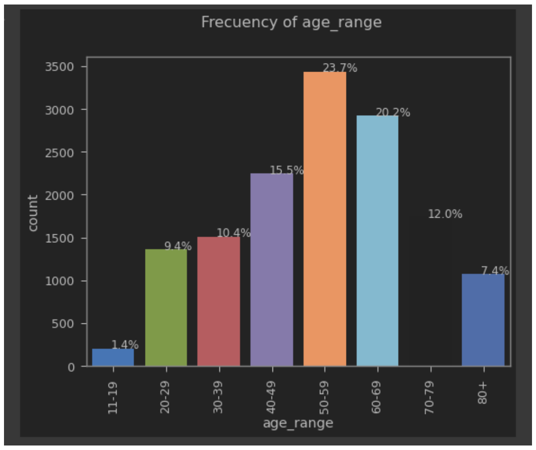


Visualize Total spent by gender:

Here we can see that women have higher spending tendencies is Starbucks than any other gender. Men have the least spending tendencies in the dataset.

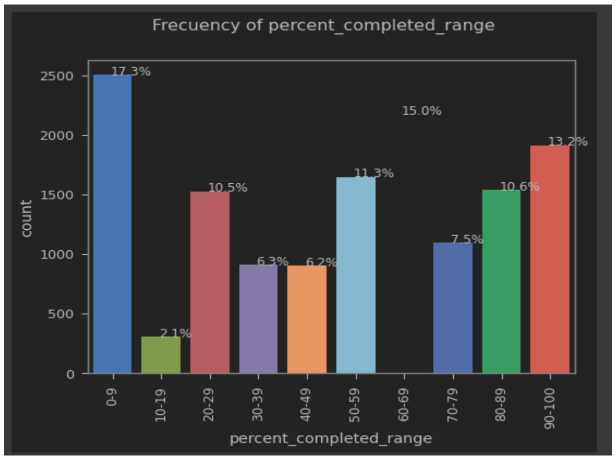


Visualize age groups by decades:

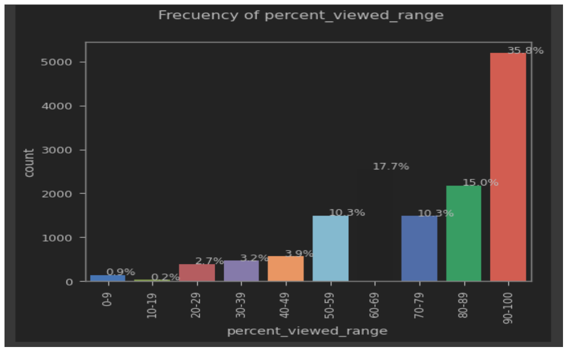


Visualize total offers completed:

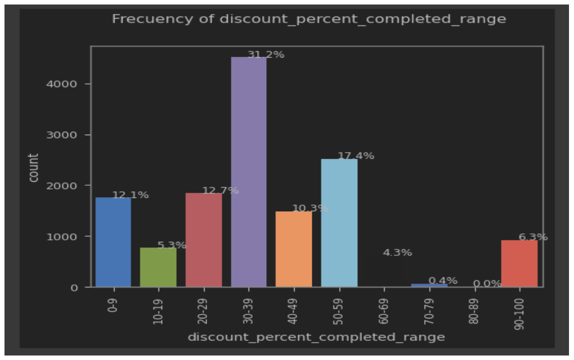
This graphic shows the percentage of people that completed a number of offers out of the ones that were sent to them.



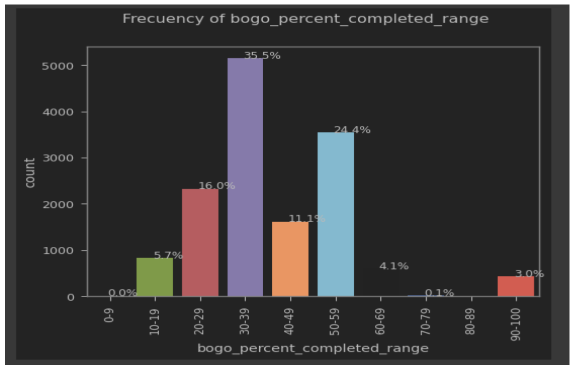
Visualize total offers viewed:



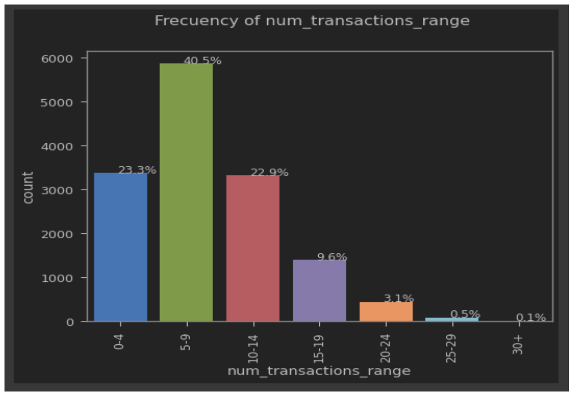
Visualize groups of people that completed a percentage range of discount offers:



Visualize groups of people that completed a percentage range of bogo offers:

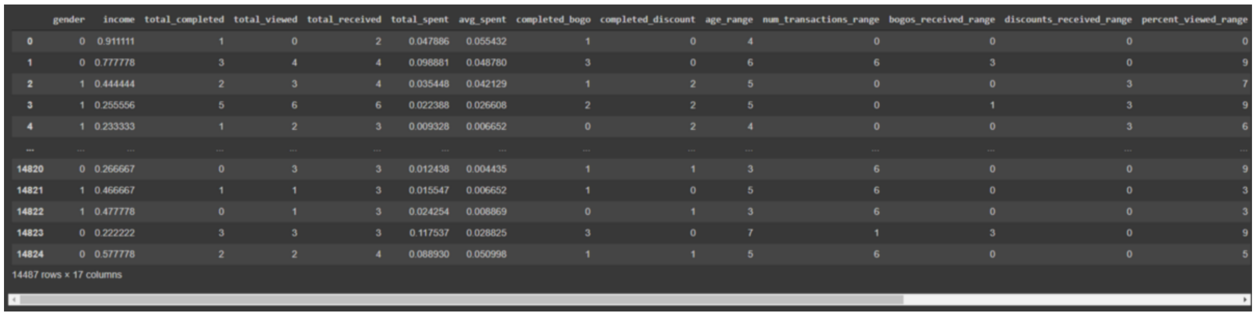


Visualize ranges of transactions done by customers:



Data Scaling

The next step would be to preprocess and scale the data to assign keys to categorical values and scale other values to be lower than 1. And this is the result:

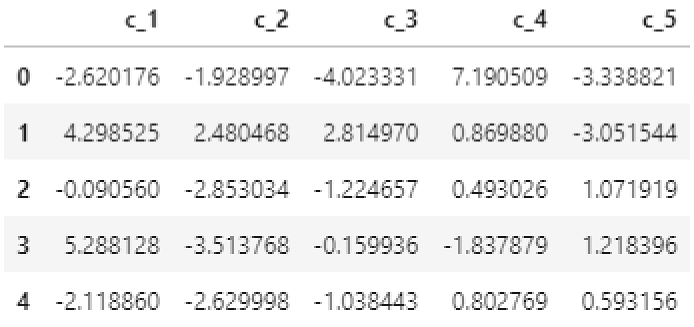


	gender	income	total_completed	total_viewed	total_received	total_spent	avg_spent	completed_bogo	completed_discount	age_range	num_transactions_range	bogos_received_range	discounts_received_range	percent_viewed_range
0	0	0.911111	1	0	2	0.047886	0.055432	1	0	4	0	0	0	0
1	0	0.777778	3	4	4	0.098881	0.048780	3	0	6	6	3	0	9
2	1	0.444444	2	3	4	0.035448	0.042129	1	2	5	0	0	3	7
3	1	0.255556	5	6	6	0.022388	0.026608	2	2	5	0	1	3	9
4	1	0.233333	1	2	3	0.009328	0.006652	0	2	4	0	0	3	6
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
14820	0	0.266667	0	3	3	0.012438	0.004435	1	1	3	6	0	0	9
14821	1	0.466667	1	1	3	0.015547	0.006652	1	0	5	6	0	0	3
14822	1	0.477778	0	1	3	0.024254	0.008869	0	1	3	6	0	0	3
14823	0	0.222222	3	3	3	0.117537	0.028825	3	0	7	1	3	0	9
14824	0	0.577778	2	2	4	0.088930	0.050998	1	1	5	6	0	0	5

14487 rows × 17 columns

PCA on the data

What is PCA you may ask? PCA stands for Principal Component Analysis and it is the reduction in dimensionality of data in order to process the data with as few dimensions as possible. When we apply PCA on our scaled data we reduce the dimension of its features and only choose some features that are most important for the clustering algorithm. The result of the obtained pca data is the following:



	c_1	c_2	c_3	c_4	c_5
0	-2.620176	-1.928997	-4.023331	7.190509	-3.338821
1	4.298525	2.480468	2.814970	0.869880	-3.051544
2	-0.090560	-2.853034	-1.224657	0.493026	1.071919
3	5.288128	-3.513768	-0.159936	-1.837879	1.218396
4	-2.118860	-2.629998	-1.038443	0.802769	0.593156

K-means:

K-means is a popular data science clustering algorithm that segments data points with similar features into spherical clusters.

[…] "It ends up basically assuming that clusters are going to be gaussian balls rather than anything else"[2].

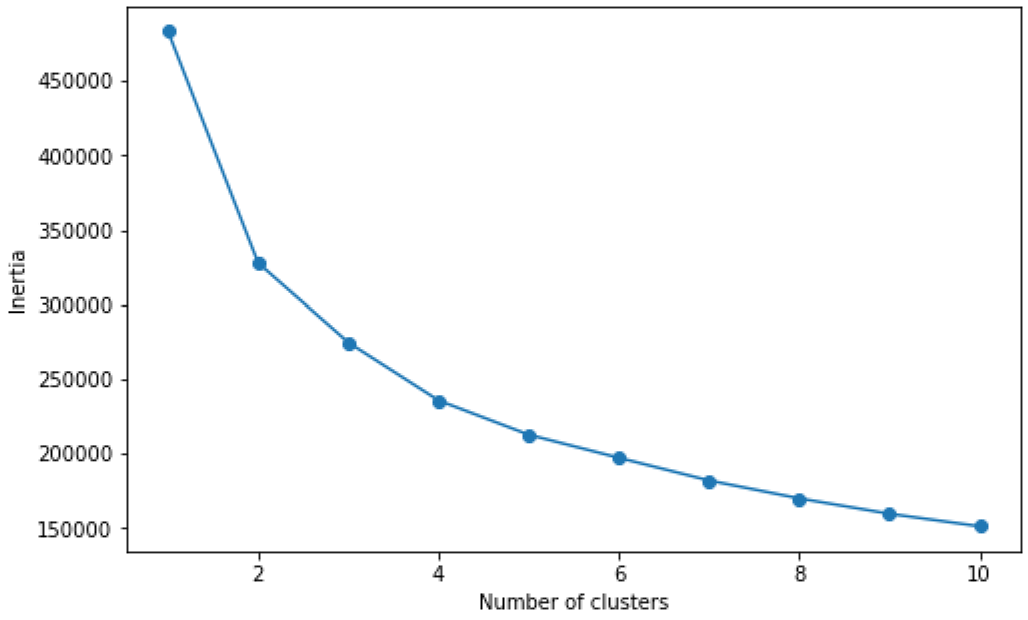
However, I do have to warn about this algorithm. It is not as precise as we may have learned before. This is effectively because of the fact that it assumes that clusters are just spheres and takes into account too much data noise that may cause non precise predictions.

Metrics:

Elbow method Inertia measures how well a data set is clustered by K-Means. It is calculated by measuring the distance between each data point and its centroid, squaring this distance and summing these squares into a cluster.

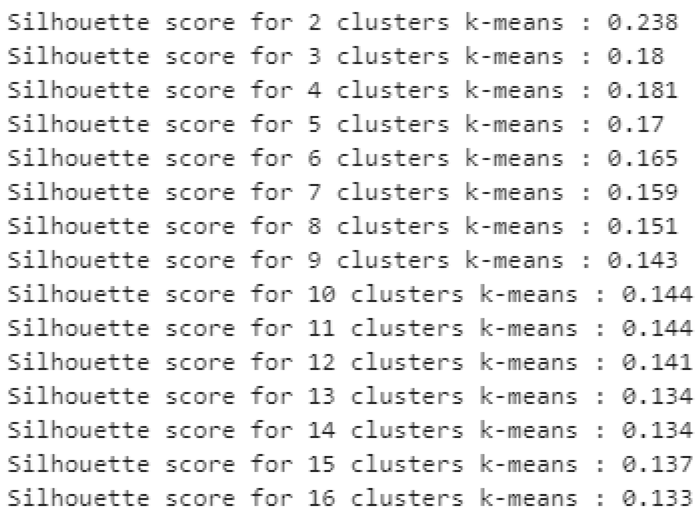
A good model is one with low inertia AND a low number of clusters (K). However, this is a trade-off because as K increases, the inertia decreases.

To find the optimal K for a data set, use the Elbow method; find the point where the decrease in inertia begins to decrease.



Silhouette value:

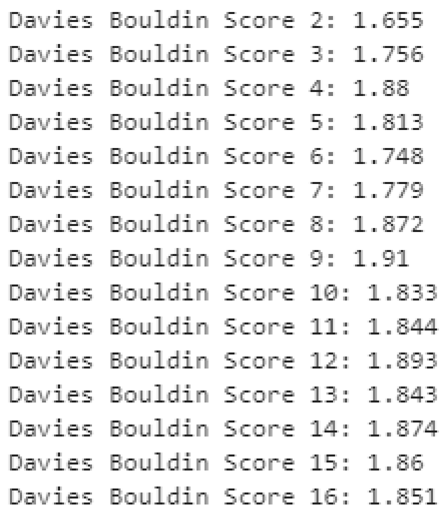
The silhouette value measures how similar a point is to its own cluster (cohesion) compared to other clusters (separation).



```
Silhouette score for 2 clusters k-means : 0.238
Silhouette score for 3 clusters k-means : 0.18
Silhouette score for 4 clusters k-means : 0.181
Silhouette score for 5 clusters k-means : 0.17
Silhouette score for 6 clusters k-means : 0.165
Silhouette score for 7 clusters k-means : 0.159
Silhouette score for 8 clusters k-means : 0.151
Silhouette score for 9 clusters k-means : 0.143
Silhouette score for 10 clusters k-means : 0.144
Silhouette score for 11 clusters k-means : 0.144
Silhouette score for 12 clusters k-means : 0.141
Silhouette score for 13 clusters k-means : 0.134
Silhouette score for 14 clusters k-means : 0.134
Silhouette score for 15 clusters k-means : 0.137
Silhouette score for 16 clusters k-means : 0.133
```

Davies Bouldin:

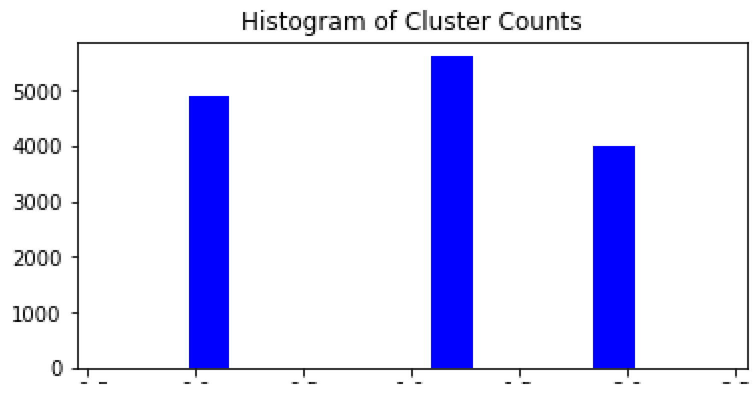
Average similarity measure of each cluster to its most similar cluster, where similarity is the ratio of distances within the cluster to distances between clusters. The minimum score is zero, and lower values indicate better clustering.



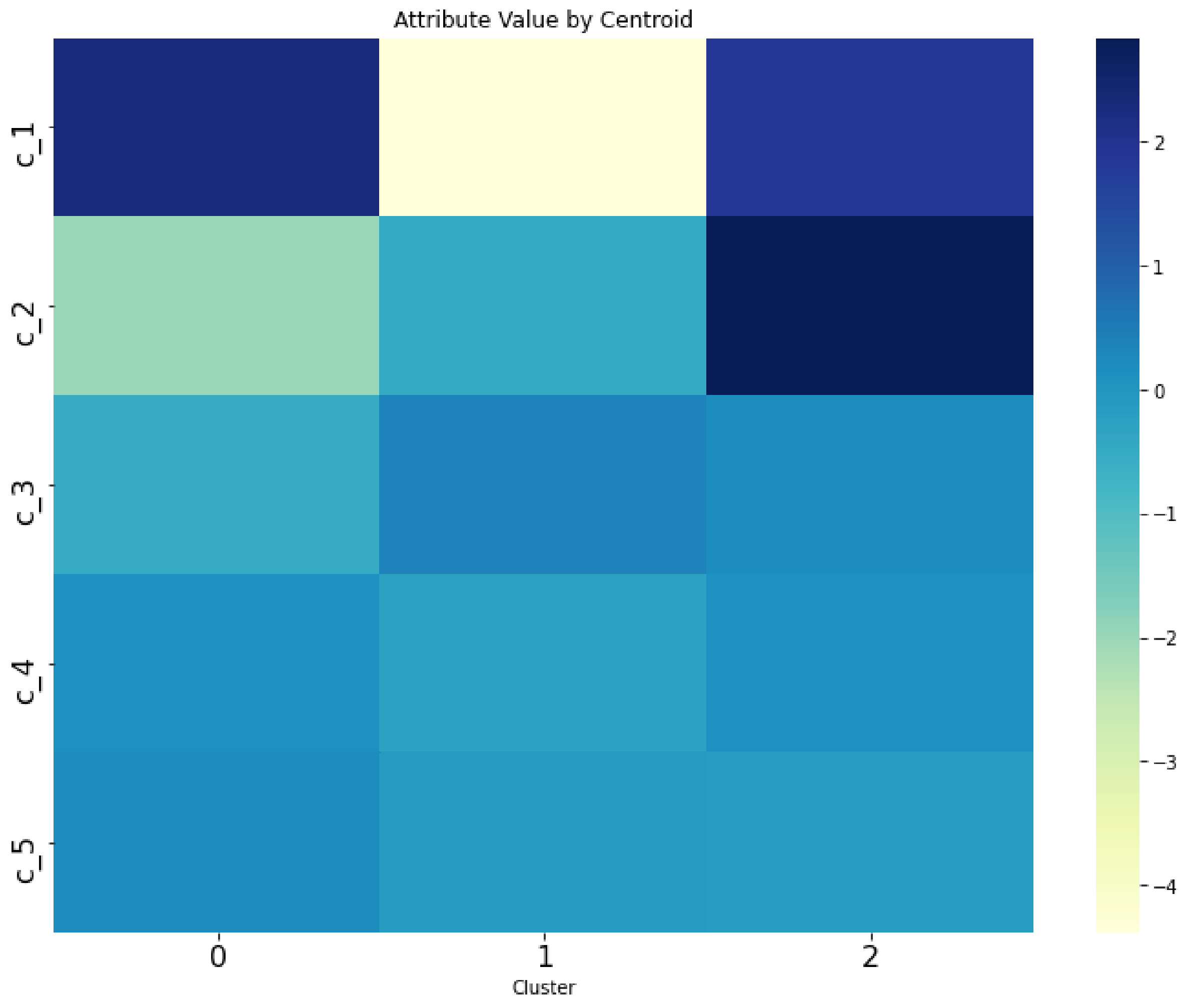
```
Davies Bouldin Score 2: 1.655
Davies Bouldin Score 3: 1.756
Davies Bouldin Score 4: 1.88
Davies Bouldin Score 5: 1.813
Davies Bouldin Score 6: 1.748
Davies Bouldin Score 7: 1.779
Davies Bouldin Score 8: 1.872
Davies Bouldin Score 9: 1.91
Davies Bouldin Score 10: 1.833
Davies Bouldin Score 11: 1.844
Davies Bouldin Score 12: 1.893
Davies Bouldin Score 13: 1.843
Davies Bouldin Score 14: 1.874
Davies Bouldin Score 15: 1.86
Davies Bouldin Score 16: 1.851
```

Visualize the distribution of data over clusters:

Based on the result obtained from the previous methods, it seems that using three clusters is the best choice. So, we get the cluster labels for each of our data points (counties) and visualize the distribution of points over each cluster.



Visualizing Centroids in Component Space:



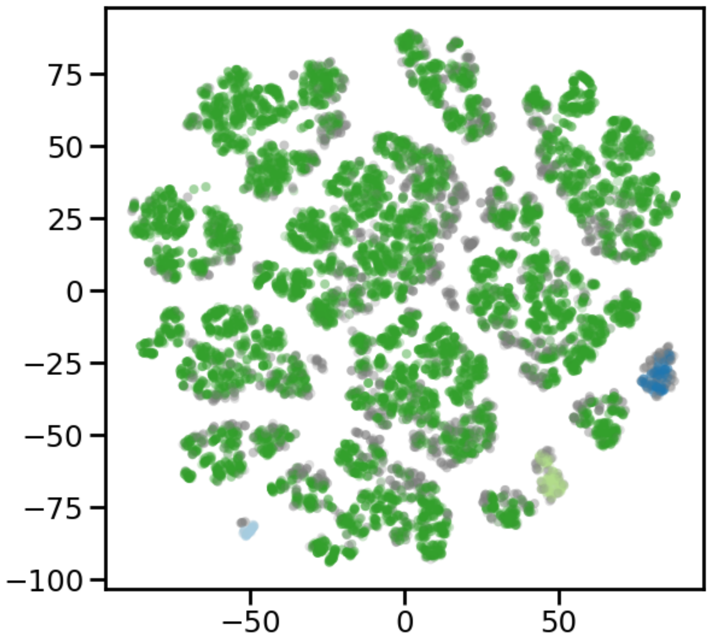
HDBSCAN

[…] “HDBSCAN is a clustering algorithm developed by Campello, Moulavi, and Sander. It extends DBSCAN by converting it into a hierarchical clustering algorithm, and then using a technique to extract a flat clustering based in the stability of clusters.” [3]. Basically, HDBSCAN uses the density based features if DBSCAN to calculated data clusters without knowing the probability density function.

This is great because well, we do not know how the PDF of our data neither do we know how many clusters can well suit our data. So we go ahead and try different parameters and previously saved datasets to try to achieve the best results and finally label each data point with its corresponding data cluster.

Cluster with Scaled dataset:

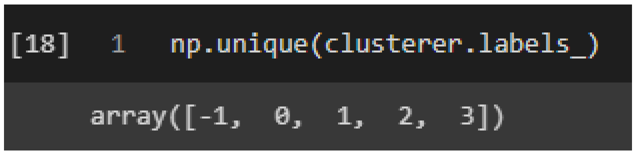
So first we try our luck with the scaled dataset that was obtained earlier and the results were the following:



It is worth mentioning that the black dots inside the graph represent the noise detected by the HDBSCAN algorithm, and they will be labeled as -1. In the meantime, it can be observed that the green cluster represents a large percentage of the data, as shown in the following image.

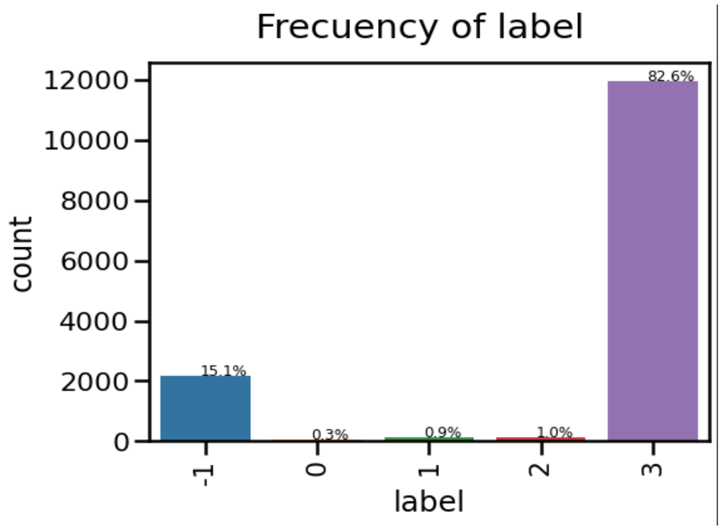
NOTE: After several trials it was defined that the best parameters were min_cluster_size=35 and min_samples=10 as it groups the data better and generates less noise.

We can also see that the algorithm identified 4 main clusters excluding the noise data points:



```
[18] np.unique(clusterer.labels_)
     array([-1,  0,  1,  2,  3])
```

And the frequency of data points for each label are as follows:

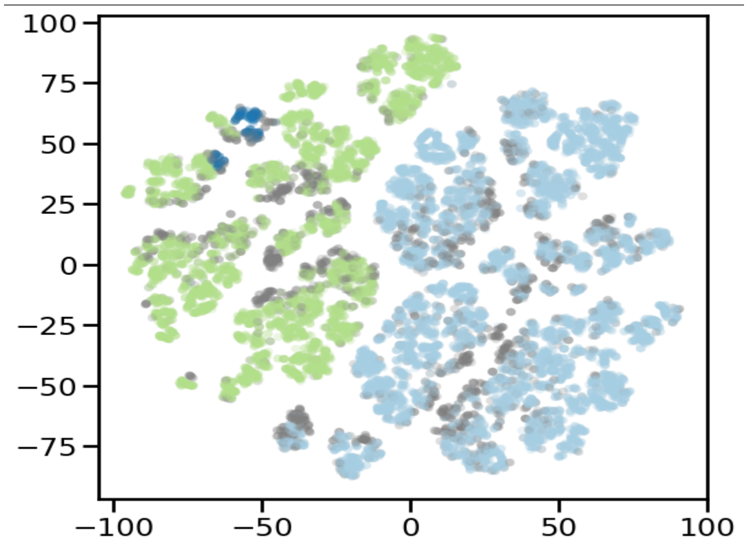


As we can see, cluster 3 covers 82.6% of the data noise covers 15.1%. We can then assume that the algorithm did not perform as well because PCA was not applied to the input data.

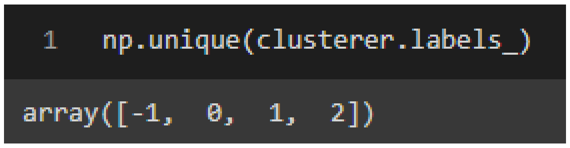
Next we tried the same procedure but with PCA applied to the input data.

Cluster with scaled dataset with PCA:

The results obtained were much more appreciable than the previous one:



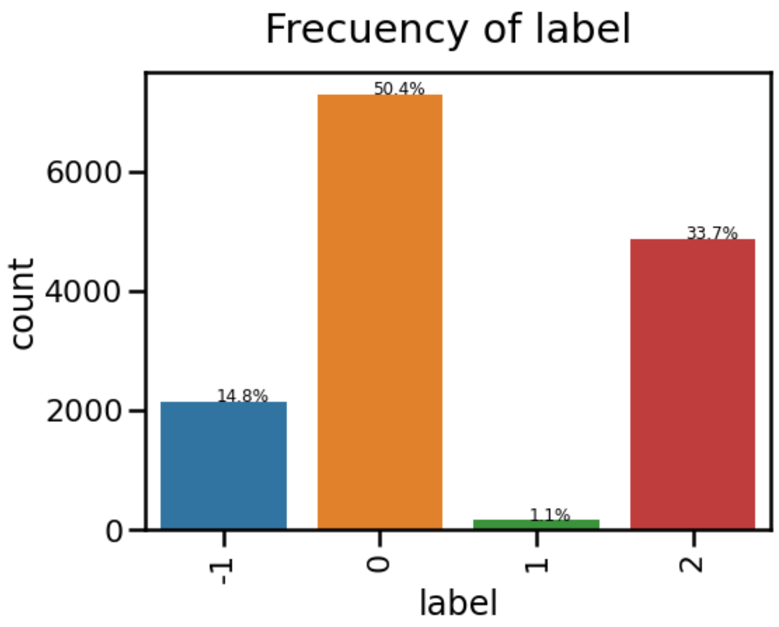
There were a total of 3 data clusters generated excluding the noise data.



```
np.unique(clusterer.labels_)
```

```
array([-1,  0,  1,  2])
```

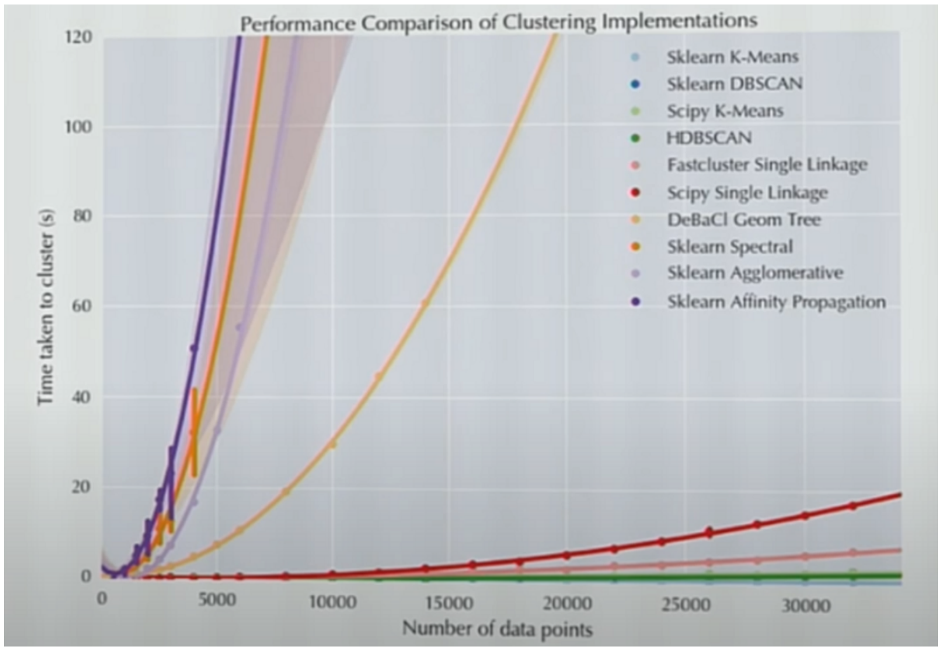
And the frequency for each cluster is as follows:

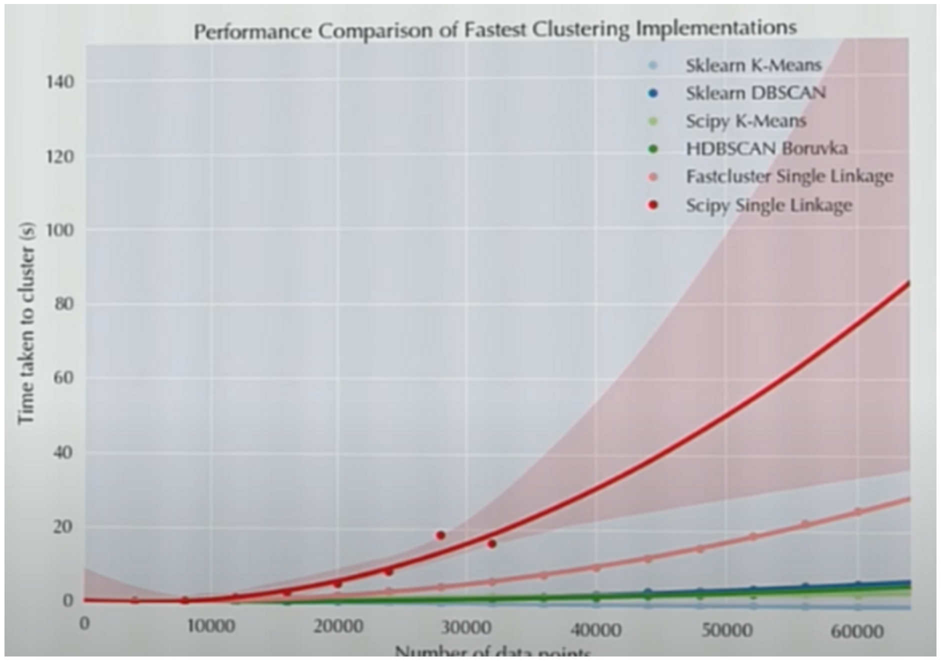


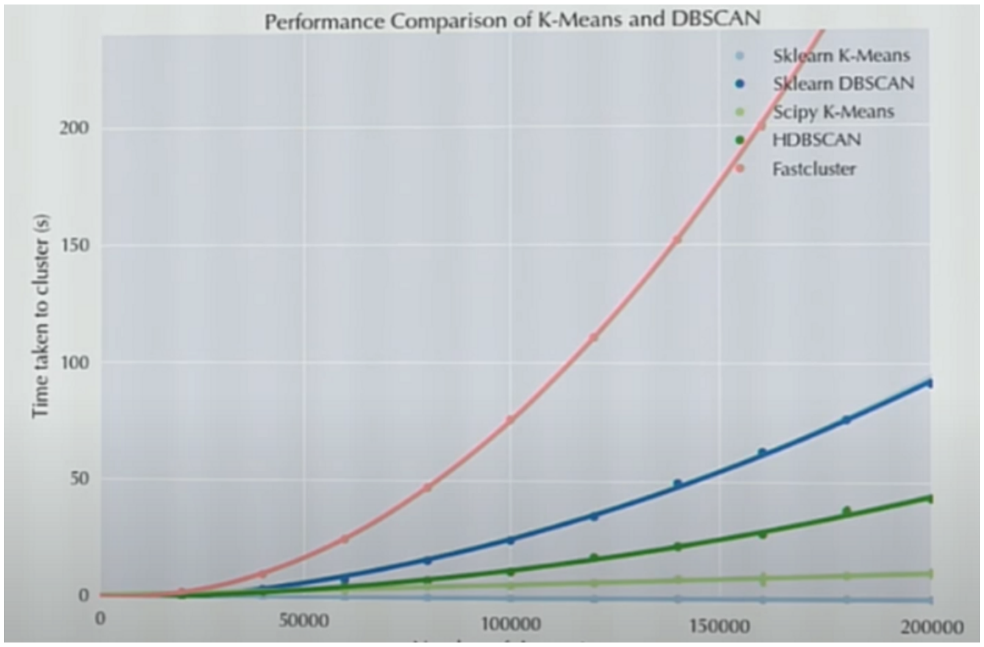
Why HDBSCAN?

The answer is that it simply is better at clustering data and it basically does all of the had work for you. It is way better than K-means and its derivatives and can calculate up to a million data points.

Here is a bit of comparisons done by Leland McInnes [2] to show how HDBSCAN performs better than other algorithms. He trialed all algorithms with the same dataset with varying types of data on each trial. Here is what the results showed:


Performance Comparison of Clustering Implementations
Time taken to cluster (s)
Number of data points
0
20
40
60
80
100
120
0
5000
10000
15000
20000
25000
30000
Sklearn K-Means
Sklearn DBSCAN
Scipy K-Means
HDBSCAN
Fastcluster Single Linkage
Scipy Single Linkage
DeBaCl Geom Tree
Sklearn Spectral
Sklearn Agglomerative
Sklearn Affinity Propagation


Performance Comparison of Fastest Clustering Implementations
Time taken to cluster (s)
Number of data points
0
20
40
60
80
100
120
140
0
10000
20000
30000
40000
50000
60000
Sklearn K-Means
Sklearn DBSCAN
Scipy K-Means
HDBSCAN Boruvka
Fastcluster Single Linkage
Scipy Single Linkage

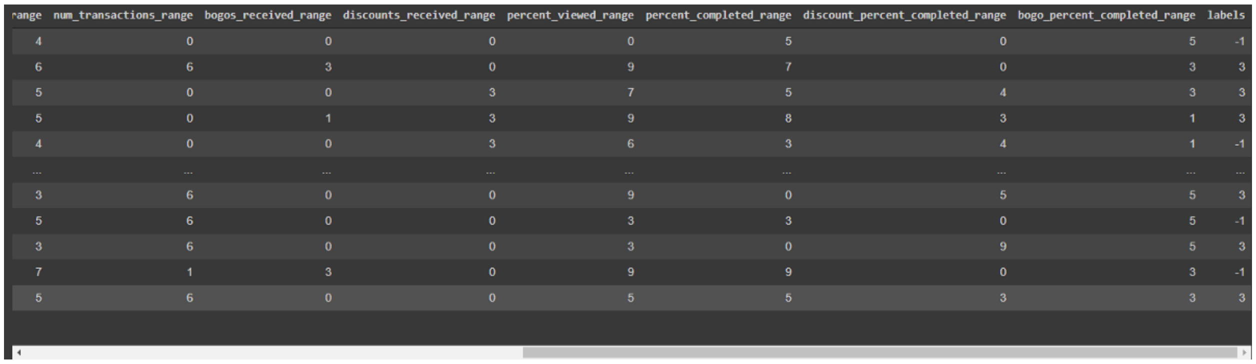


What we can observe from the previous images is that HDBSCAN outperforms most algorithms in all three of the occasions even though more data was being added each time.

Final Segmentation

So, now that we have found the cluster label for each point, why not go ahead and place them in a their corresponding rows in the first scaled dataset that was created earlier?

We add the new column to the data and we get something that looks like this:



range	num_transactions_range	bogos_received_range	discounts_received_range	percent_viewed_range	percent_completed_range	discount_percent_completed_range	bogo_percent_completed_range	labels
4	0	0	0	0	5	0	5	-1
6	6	3	0	9	7	0	3	3
5	0	0	3	7	5	4	3	3
5	0	1	3	9	8	3	1	3
4	0	0	3	6	3	4	1	-1
...	...	...	...	...	...	...	...	...
3	6	0	0	9	0	5	5	3
5	6	0	0	3	3	0	5	-1
3	6	0	0	3	0	9	5	3
7	1	3	0	9	9	0	3	-1
5	6	0	0	5	5	3	3	3

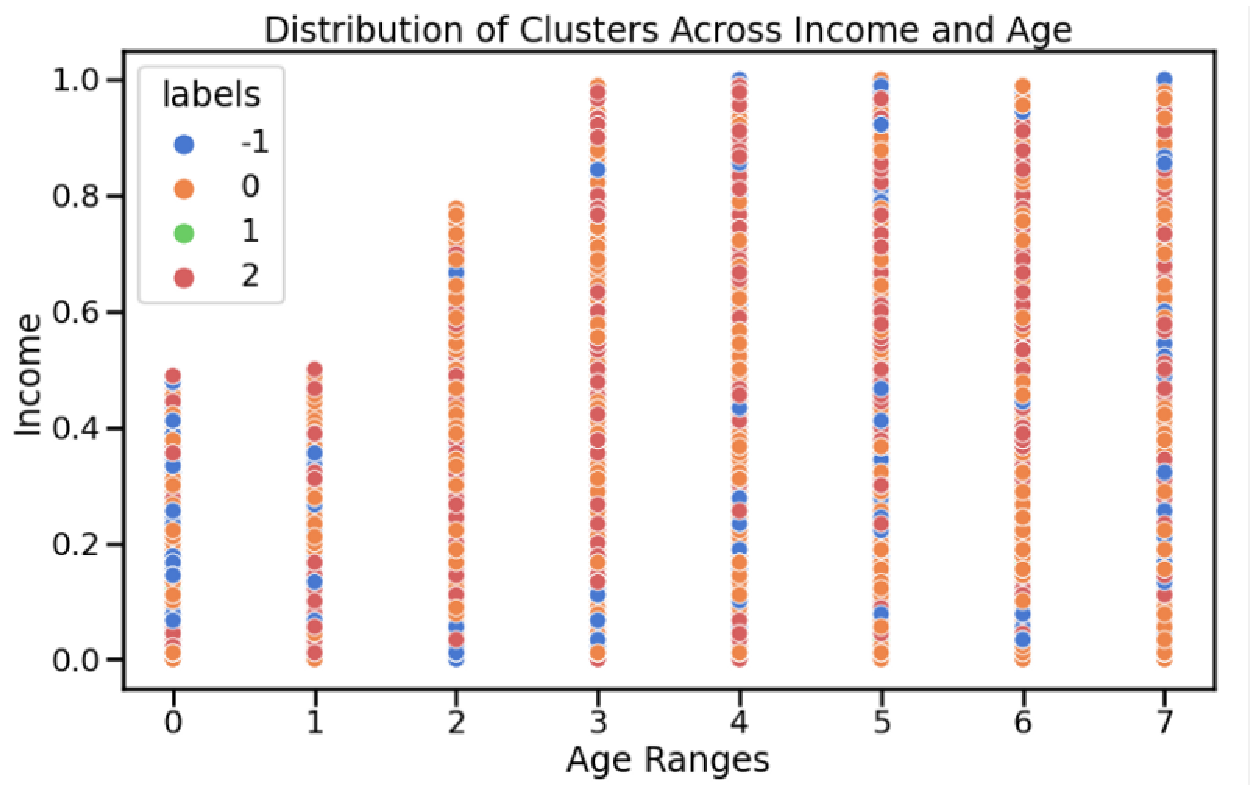
And as previously mentioned before, those labeled as -1 are noise data points. If you want to, you can go ahead and remove those points. But I will keep them to give the results more weight and credibility.

Results

So after so graphical representations and calculations, we were able to draw a few conclusions about the clusters and predict which cluster group is more likely to accept an offer.

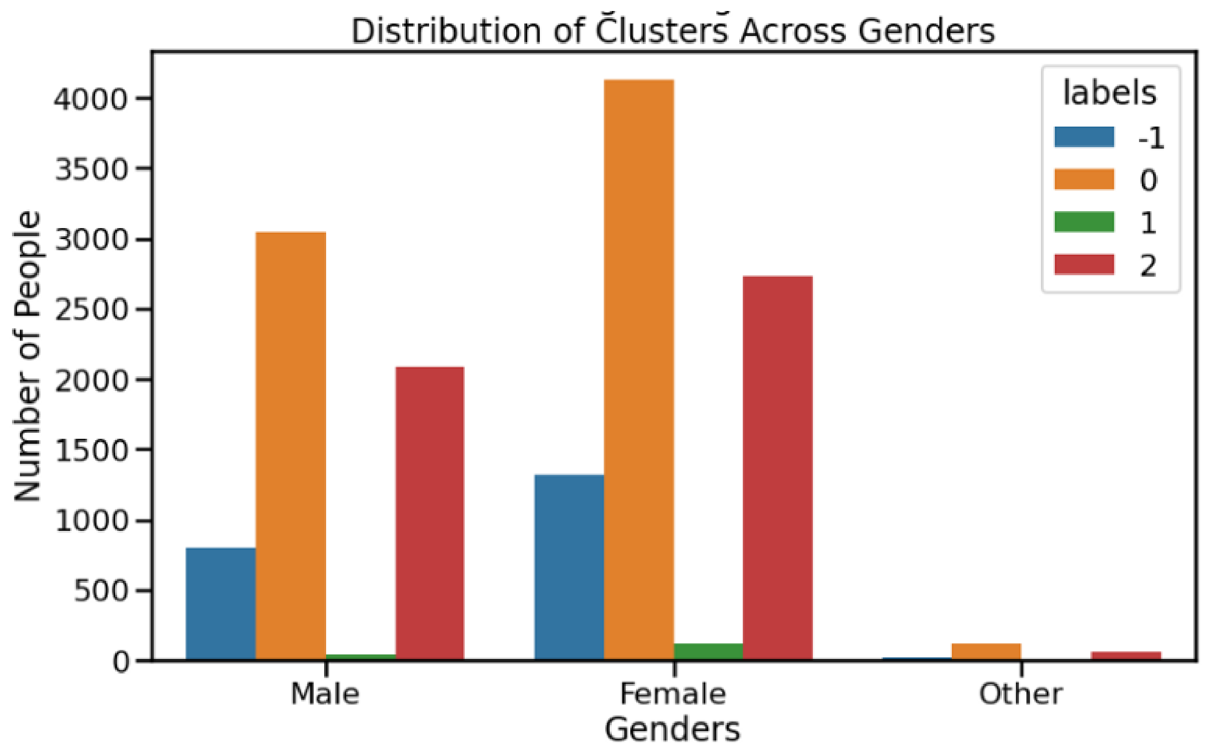
Age and income:

First will take a look at the distribution of the cluster across the relationship between age and income.



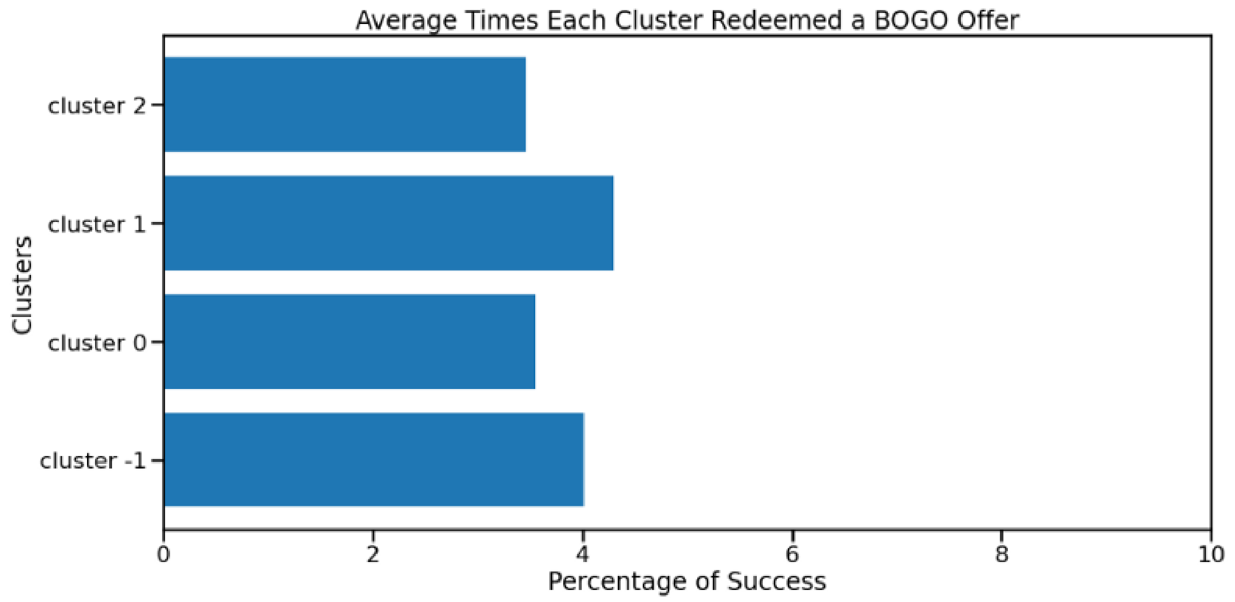
Here we can see the graph indicated to us that the older people are the ones with the most income. Cluster one points to those customers who are 70+ years old and have the highest incomes. While cluster two may indicate those customers who are in the middle age and also have high incomes. However this does not tell us much which type of offer would best suit each cluster.

Gender:



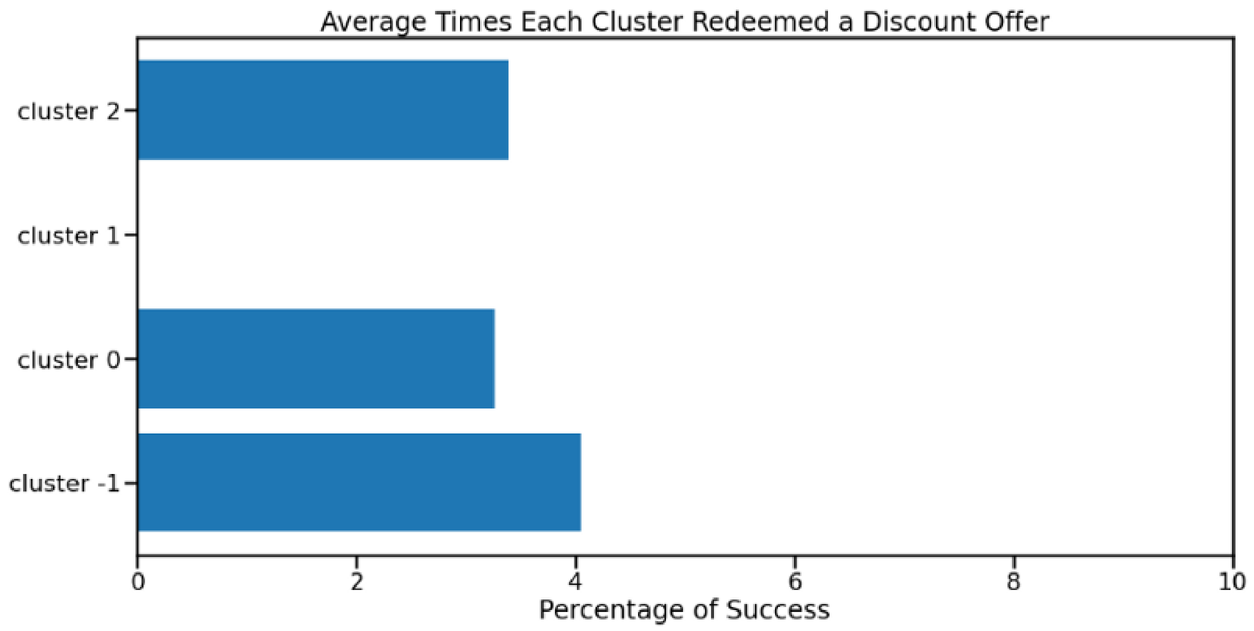
Gender did not have much of an effect in the assignment of clusters to customers. However it is worth mentioning that each cluster has more females than males which may be useful information further down the line.

Offer types



BOGOS

According to the graphics, we can see that the cluster with the highest average of times that a BOGO was redeemed is cluster 1. However, as we observed earlier, cluster 1 only contains 1.1% of the total data. So in this case we will explore the possibility of the highest value coming from cluster 0 or cluster 2. Cluster -1 is not considered because it is considered as noise. Visually we see that between these two clusters, cluster 0 has the highest average value. So without any fear of assumption we can go ahead and predict that cluster ***0 will respond the best to BOGO***.



Discount Offers

In the case of the discount offers however, on average, cluster two completed the most offers with a small difference to cluster 0. Cluster 1 has 0 completed offers on average and again we do not consider cluster -1 in this case because it is noise. So again without fear of assumption, we go ahead and say that ***cluster 2 responds best to discount offers***.

References

1. Otar, C., 2018. What Percentage Of Small Businesses Fail—And How Can You Avoid Being One Of Them?
2. High Quality, High Performance Clustering with HDBSCAN | SciPy 2016 | Leland McInnes
3. How HDBSCAN Works

Project Link

The GitHub repository containing all of the project material is here. Keep in mind that I ran the notebooks in Google Colab and in order to read external data I had to save the data in my external drive.

Also, if you intend on running the training notebook, you should also keep in mind that these cells were run on Amazon Sagemaker, so you may need an account in order to run it.

Contact me

If you wish to contact me or view some of my work, you can do so with the following medias:

Github

Twitter

LinkedIn

Personal Site