DSC 630 – Predictive Analytics

Course Project

Kimberly Cable

(Individual)

Fall, 2022

Data Selection and Project Proposal

Overview

Olist is a Brazilian e-commerce company that connects small businesses to a larger marketplace. It gives these small businesses a way to manage their products, shipping, and online payments. They have approximately 200,000 users in about 180 countries.

Business Problem

With any online retailer, retaining customers is key but knowing who may leave and who stays could be guesswork. Understanding how and why customers stay and shop and why they leave is pivotal to a company's business.

Knowing who your customers are and how and why they shop or do not return plays a big part in customer service which ultimately enhances a customer's satisfaction level. A high satisfaction level could lead to overall better reviews and with these good reviews, their sellers will see more new sales.

This study will hope to find similar traits among its customers. This will help the marketing team know who best to send offers to or to whom might they need to send a discount coupon because it looks like they haven't purchased in a while.

Data

The data has been sourced from Kaggle which has 100,000 online orders from 2016 to 2018.

The data consists of records with products, customers, and review information for each transaction provided by Olist.

The data consists of the following datasets (see Figure 1):

- Customers: This dataset has information about the customer and their location.
- Geolocation: This dataset has information about the Brazilian zip codes and their latitude/longitude coordinates.
- Order Items: This dataset has information about the items purchased within each order.
- Order Payments: This dataset has information about the order payment options.
- Order Reviews: This dataset has information about the reviews made by the customers.
- Orders: This dataset has information about all customer orders.
- Products: This dataset has information about all the products sold by Olist.
- Sellers: This dataset has information about the sellers that fulfilled the orders made at Olist.
- Category Name Translation: This dataset has English/Portuguese translations for all products sold at Olist.

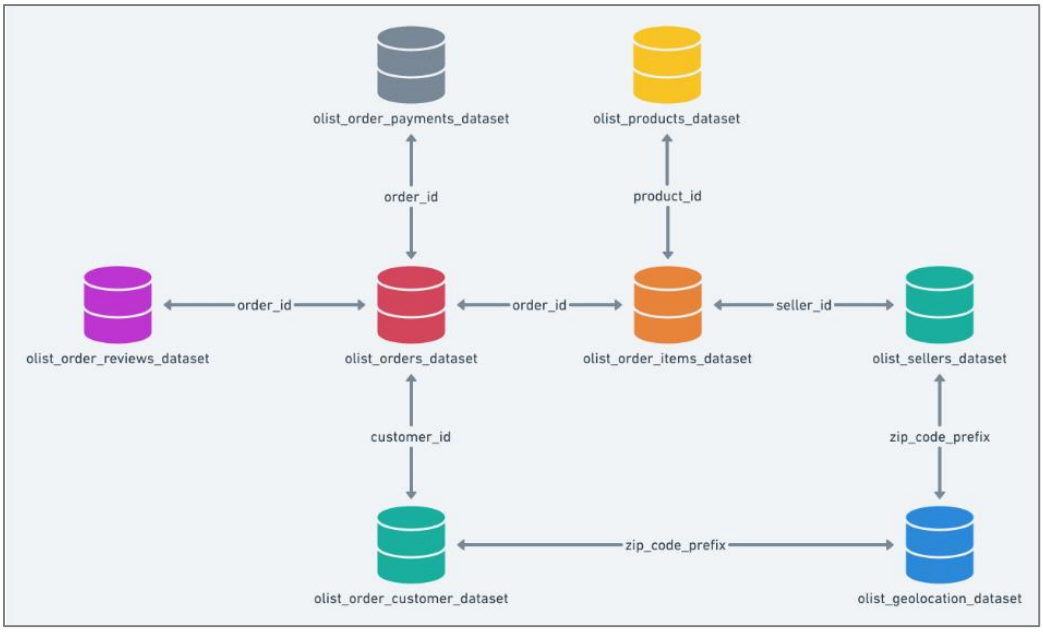


Figure 1: Data Schema

Model Selection

I plan to use the k-means clustering method. K-means is used to help identify clusters or groups within your dataset. This model will help group customers based on previous purchases and reviews that they made. I will also investigate RFM (recency, frequency, monetary) analysis. It is a marketing technique used to rank and group customers based on the number of times the customer has purchased, the last time the customer has made a purchase, and the total dollar amount the customer has spent.

Model Evaluation

To make sure my model is performing correctly, I will use the elbow method and/or the silhouette coefficient to make sure I am using the correct number of clusters. I hope to use a decision tree using the clusters found through the k-means clustering analysis. This will help analyze the results of the clusters.

Learning Objective

I hope to learn more about the customers who shop with Olist. This will help sellers target different types of individuals by grouping them into different categories to know whom they need to target with promotions and whom they need to go after to bring them back.

Risks

As with any dataset, there may be inaccuracies with the data, data may be missing or invalid. Also, the model I have chosen may not be the correct choice and another may be more suited for this data. Further analysis may need to be done.

Contingency Plan

If k-means clustering is not showing good accuracies for clustering, I may investigate another method for segmentation. Re-examining the data may also help. Removing outliers, narrowing the feature selection, etc. may help with model building and evaluation.

Preliminary Analysis

Will the data be able to answer the questions?

The data is comprised of 9 datasets from a dump of the company's database tables. The ids were left in the datasets to be used for indexing purposes. There are not needed for the model.

Customers:

- customer_id: key to the orders dataset. Each order has a unique customer_id.
- customer_unique_id: unique identifier of a customer.
- customer_zip_code_prefix: first five digits of customer zip code
- customer_city: customer city name
- customer_state: customer state

Since the customer_id is unique to each customer, I dropped the customer_unique_id. I also dropped customer_zip_code_prefix and customer_city as they will not be needed and just left customer_state to analyze where most customers reside in each state in Brazil.

Final dataset: 99,441 rows and 2 columns

Orders:

- order_id: unique identifier of the order

- customer_id: key to the customer dataset. Each order has a unique customer_id.
- order_status: reference to the order status (delivered, shipped, etc).
- order_purchase_timestamp: shows the purchase timestamp.
- order_approved_at: shows the payment approval timestamp.
- order_delivered_carrier_date: shows the order posting timestamp. When it was handled to the logistic partner.
- order_delivered_customer_date: Shows the actual order delivery date to the customer.
- order_estimated_delivery_date: shows the estimated delivery date that was informed to the customer at the purchase moment.

I converted all date columns to a datetime type.

Looking at the graph of orders over time, I noticed little to no orders in 2016 and little to no orders after August 2018 so I dropped those orders (see Figure 2).

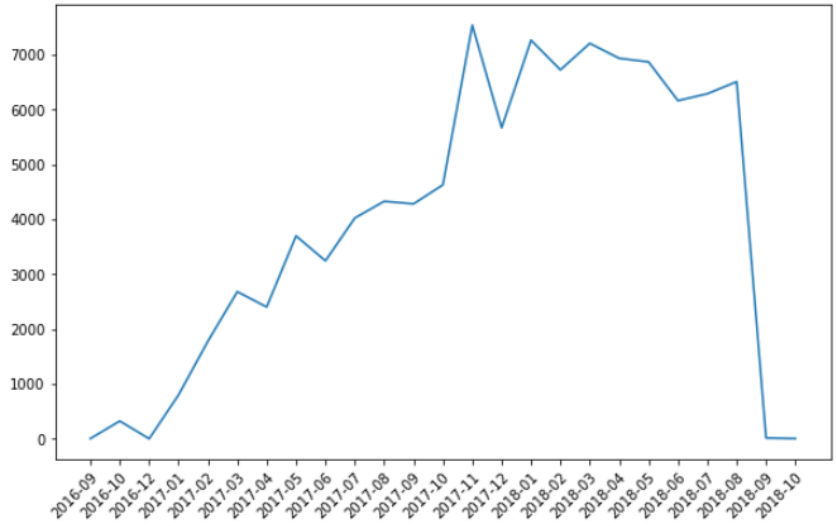


Figure 2: Number of Orders over Time

There were 82 empty values in the order_approved_at column. I looked at the order_status of each of the 82 values. 63 were 'canceled', 14 were 'delivered', and 5 were 'created'. The 'delivered' status should not have any empty order_approved_at values so I replaced the null values with the order_purchase_timestamp.

There were 1,602 empty values in the order_delivered_carrier_date column. The order status for 'delivered' was only 2 and the rest had status' that were okay for this field. I replaced the 2 empty values with the order_approved_at date.

There were 2,727 empty values in the order_delivered_customer_date column. The order status of 'delivered' was only 8 and the rest had status' that were okay for that field. Looking at the histogram, I saw that it was skewed left so to replace this field, I got the difference between the order_delivered_customer_date and the order_delivered_carrier_date. Then I took the median number of days. I added the number of days to the order_delivered_carrier_date to get the new order_delivered_customer_date for those empty values. In this case, the median difference was 7 days which was added to the order_delivered_carrier_date.

Final Dataset: 92, 580 rows and 8 columns

Order Items:

- order_id: unique identifier of the order
- order_item_id: sequential number identifying a number of items included in the same order.
- product_id: product unique identifier
- seller_id: seller unique identifier
- shipping_limit_date: Shows the seller shipping limit date for handling the order over to the logistic partner.
- price: item price
- freight_value: item freight value item (if an order has more than one item the freight value is split between items)

I dropped the seller_id and the shipping_limit_date as those two columns were not needed.

Final Dataset: 112,650 rows and 5 columns

Order Reviews:

- review_id: unique review identifier

- order_id: unique identifier of the order
- review_score: Note ranging from 1 to 5 given by the customer on a satisfaction survey.
- review_comment_title: Comment title from the review left by the customer, in Portuguese.
- review_comment_message: Comment message from the review left by the customer, in Portuguese.
- review_creation_date: Shows the date in which the satisfaction survey was sent to the customer.
- review_answer_timestamp: Shows satisfaction survey answer timestamp.

I dropped all columns except the review_id and the review_score. I may add this back with more research.

Final Dataset: 99, 224 rows and 2 columns

Products:

- product_id: unique product identifier
- product_category_name: root category of product, in Portuguese.
- product_name_lenght: number of characters extracted from the product name.
- product_description_lenght: number of characters extracted from the product description.
- product_photos: number of product published photos
- product_weight_g: product weight measured in grams.
- product_length_cm: product length measured in centimeters.
- product_height_cm: product height measured in centimeters.
- product_width_cm: product width measured in centimeters.

I merged the category name translation dataset with the products dataset. There were 2 category names that did not have translations. I googled the Portuguese names and replaced those names with their English translations.

I dropped all columns except the product_id and the product_category_name as the other columns were not needed.

Final Dataset: 32,951 rows and 2 columns

Sellers:

- seller_id: seller unique identifier
- seller_zip_code_prefix: first 5 digits of seller zip code
- seller_city: seller city name
- seller_state: seller state

Final Dataset: 3095 rows and 4 columns

Order Payments:

- order_id: unique identifier of the order.
- payment_sequential: a customer may pay an order with more than one payment method. If he does so, a sequence will be created to
- payment_type: method of payment chosen by the customer.
- payment_installments: number of installments chosen by the customer.
- payment_value: transaction value.

Final Dataset: 103,886 rows and 5 columns

Geolocation:

- geolocation_zip_code_prefix: first 5 digits of zip code
- geolocation_lat: latitude
- geolocation_lng: longitude
- geolocation_city: city name
- geolocation_state: state

I did not use this dataset as all customers were in Brazil.

Product Category Name Translation:

- Product_category_name: category name in Portuguese
- Product_category_name_english: category name in English

I did not use this dataset on its own and merged it into the products dataset.

Overview

Looking at the data initially, it looks like there will be enough columns and rows for me to choose from to perform a customer segmentation analysis.

Visualizations that will be useful

As I am looking to do modeling for customer segmentation, knowing more about how many items a customer purchased and how much they spent will be useful information. Also, how the different features may have played into how the customer scored their order the way they did.

I plan to graph the following. Most of what we learn from these visualizations will be to preliminarily see what kinds of customers olist has.

- Number of orders per year, month, day of the week, and time of day:
  - What months, days or time of day do our customers mostly shop?
- Top and bottom categories that customers purchased from. Both in the number of items purchased and the amount spent:
  - What are the most popular categories our customers like to purchase?
- The number of orders per State:
  - Do customers reside throughout Brazil or only in a few States?
- Review scores count per score:
  - This will give us a look into how we are doing with the various customers.
- Review scores per order status:
  - Does the status of an order affect the review score?
- Review score per difference between the purchase date and delivered date:
  - Does the amount of time between the purchase date and the actual delivered date affect the review score?
- Review score per difference between the estimated delivery date and actual delivery date:
  - Does the amount of time between the estimated delivery date and the actual delivery date affect the review score?
- Top and bottom customers in spending:
  - How much do our customers spend?
- Top and bottom customers in the number of orders purchased from olist:
  - How many items do our customer purchase?

Do you need to adjust the data and/or driving questions?

After I do some exploratory analysis, I would like to look at RFM (recency, frequency, monetary) analysis to help me look at the buying behavior of the customers and then start building the customer segmentation model. Also, initially, I was thinking of doing either K-means or RFM, but I think I will do RFM first then K-means with the RFM data to help achieve my goals of finding out more about how to keep customers and receive high scores.

Do I need to adjust my model/evaluation choices?

Doing a little research in customer segmentation analysis, I think I will need to do a Recency, Frequency, and Monetary (RFM) analysis first then use this information for a K-means clustering model. This will help qualitatively rank and group the customers so the K-means clustering model can easily distinguish the different clusters.

Are my original expectations still reasonable?

The original expectations were to understand how and why customers stay and shop and why they leave. I believe with the data I have and the tools I have presented I can still achieve this.

Finalizing My Results

Prepping the Data

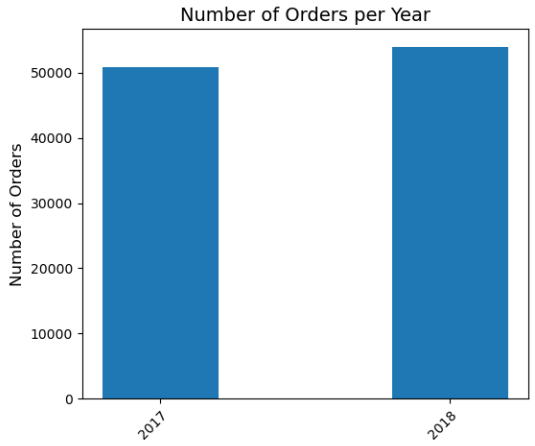
After initially prepping each dataset during the preliminary analysis, I merged all datasets using the following keys:

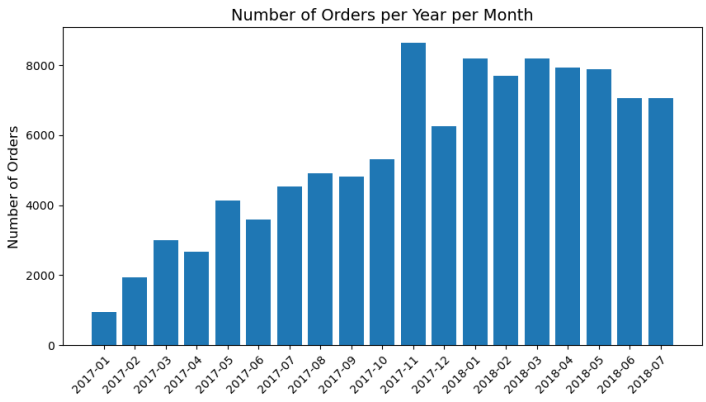
Order Reviews ← order_id → Orders ← order_id → Order Items ← product_id

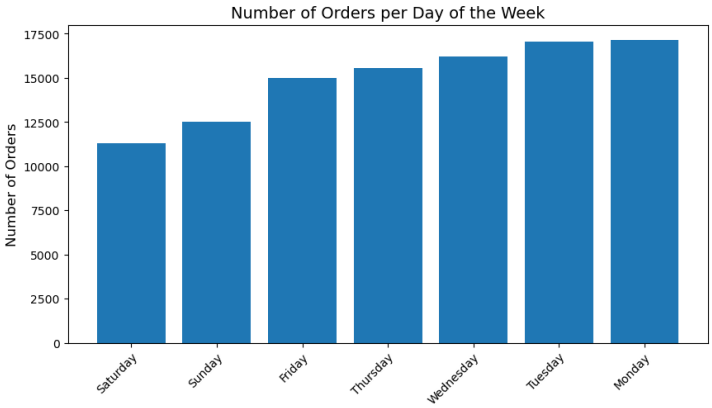
Once merged, the dataset had 104,782 rows and 15 columns.

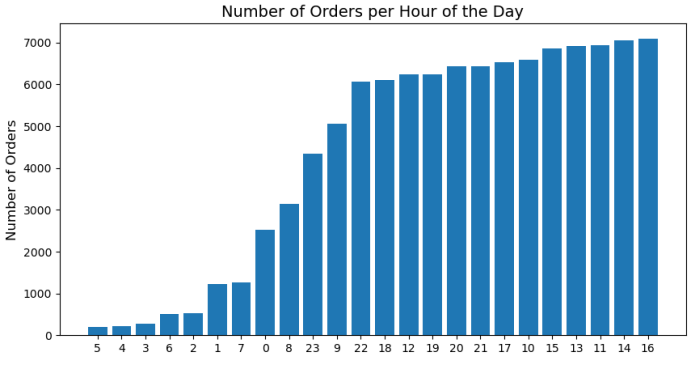
Visualizations:

Order Purchases Timeframes:



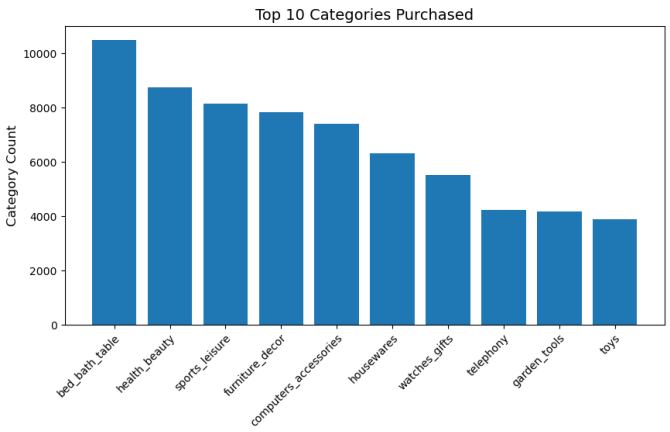


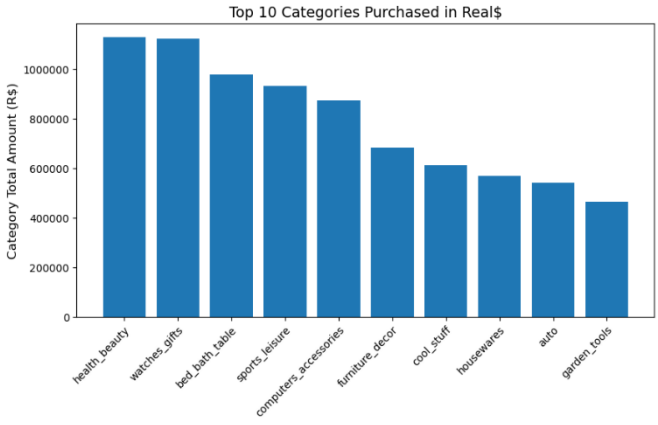




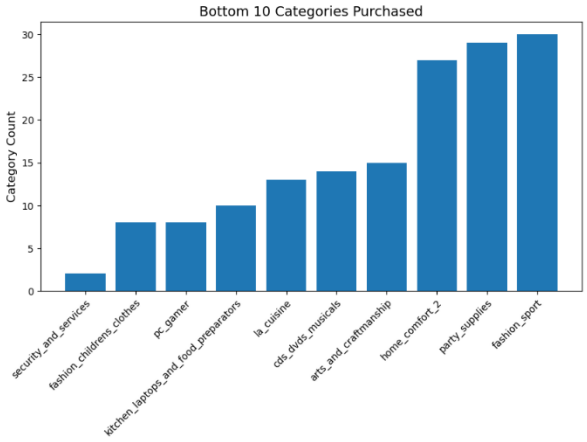
The number of orders grew between 2017 to 2018. November and January had the highest number of orders probably due to the holidays. March, April, and May were also good purchasing months. Weekdays were also more popular purchasing days as the beginning of the week was best probably due to items needed from the weekend. and early evenings.

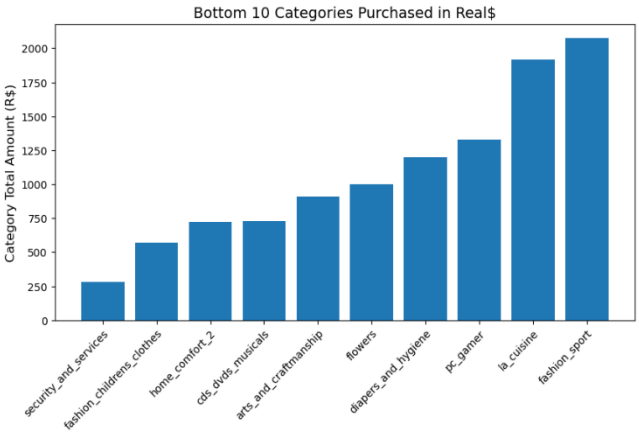
Categories





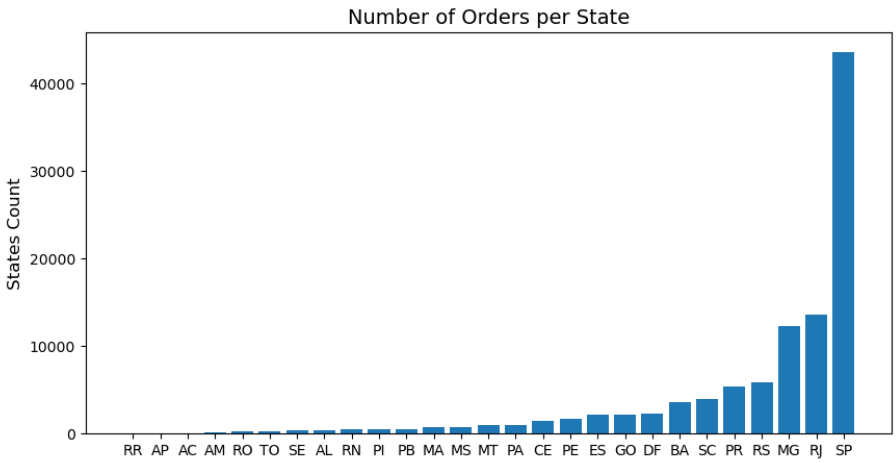
Most items that were purchased were from the Bed, Bath & Table category but the top category in the amount purchased was Health & Beauty. Most customers purchased items in the bedroom/bathroom area or that were used in the bedroom/bathroom area.





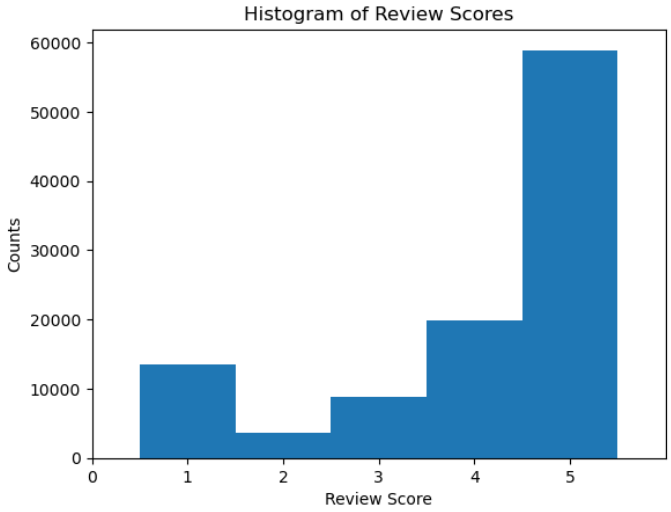
Looking at both the number of items purchased and the amount spent, the Security and Services and Children's clothes categories are at the bottom. It could be that there are not many sellers of these items or more marketing needs to be allotted for these bottom categories.

States

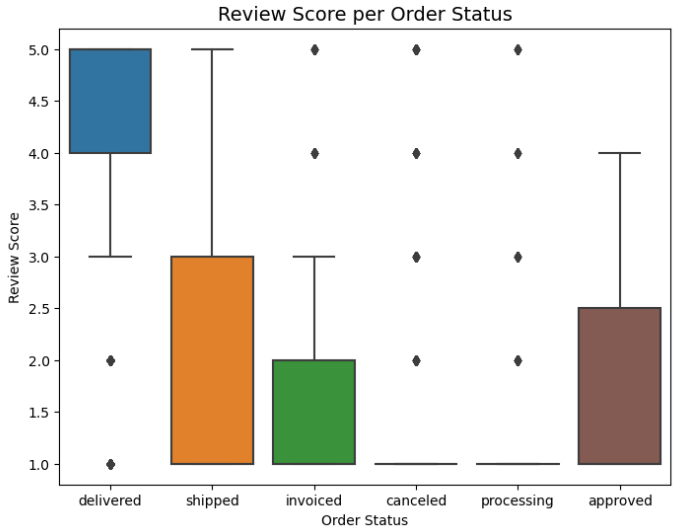


The top State was São Paulo by far compared to the other States in Brazil. Marketing may investigate advertising more in States other than Sao Paulo.

Review Score

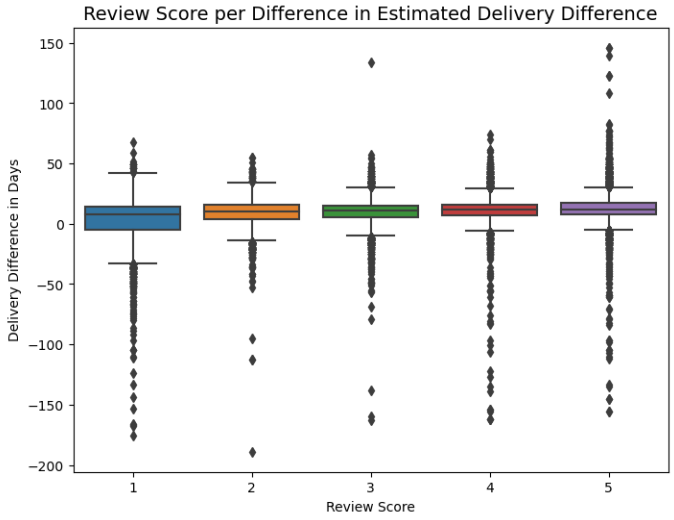


The review score of 5 is by far the most used by olist customers which is a good sign.

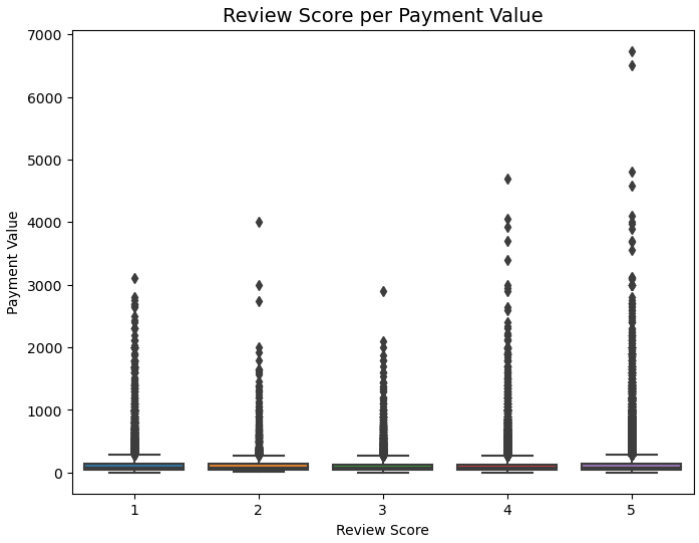


Looking at the review score based on the order status does show purchases that were delivered by far getting a review score of 5 and the other statuses, not surprisingly, were lower.

Wanting to see if the review score changed between the difference between when the customer ordered the item to when it was delivered, it looks like the review score was lower when the item took longer to get delivered which is not surprising either.



I was then curious if the difference between the estimated delivery date and the actual delivery date had an influence on the review score. It looks like the it didn't make much of a difference.



Next, I wanted to see if review scores had any effect on the amount of the purchase, but it doesn't look to have any affect.

Customers

	customer_id	review_score	order_status	price	freight_value	product_category_name	cust_delivery_diff	est_delivery_diff
4201	d6646ea91d8cd9fc7e6882a7068779d4	5	delivered	81.99	14.51	computers_accessories	7	20
4350	679f84ceb2ee4ca5bca0c3ea34647746	5	delivered	59.90	17.67	garden_tools	20	4
13293	b4afeb58ac51bc903c5362286c6a5cfe	5	delivered	19.30	11.73	drinks	11	5
49414	10de381f8a8d23fff822753305f71cae	5	delivered	65.49	16.22	furniture_decor	19	5
54499	b7770073b02ed1d626a027ce86a4ff82	5	delivered	66.90	31.65	sports_leisure	10	44
67152	0d93f21f3e8543a9d0d8ece01561f5b2	5	delivered	20.70	16.11	housewares	8	8
67939	1ff773612ab8934db89fd5afa8afe506	5	delivered	284.99	16.87	drinks	14	18
77657	20c93357daf05d1c3a092be59aea2c2b	5	delivered	20.50	16.91	drinks	10	14
90355	0e772d9e02b17408e716f35cd1dcc222	5	delivered	36.99	11.85	bed_bath_table	10	13
96279	adb32467ecc74b53576d9d13a5a55891	5	delivered	51.00	1.20	garden_tools	14	20

This chart shows an example of customers that gave olist a review score of 5 and how much they spent along with the difference in days from estimated delivery to actual delivery and the difference in days between estimated delivery and actual delivery.

	customer_id	review_score	order_status	price	freight_value	product_category_name	cust_delivery_diff	est_delivery_diff
6639	91f92cfee46b79581b05aa974dd57ce5	1	delivered	108.00	15.52	watches_gifts	11	13
19725	d5f2b3f597c7ccafbb5cac0bcc3d6024	1	delivered	59.00	13.43	garden_tools	14	10
24699	4a60b2ce1ee8c7b828e4bbcca5b86b41	1	delivered	137.90	38.81	computers_accessories	14	1
25887	be1c4e52bb71e0c54b11a26b8e8d59f2	1	delivered	49.99	7.10	bed_bath_table	5	11
37089	78fc46047c4a639e81ff65f0396e02fe	1	delivered	109.97	34.04	furniture_living_room	5	13
46421	be1b70680b9f9694d8c70f41fa3dc92b	1	delivered	100.00	10.12	computers_accessories	10	2
68674	cb87122c4871e202777cf243fbea2d12	1	delivered	149.91	0.14	computers_accessories	11	23
77621	a7693fba2ff9583c78751f2b66ecab9d	1	delivered	29.99	7.78	telephony	8	5
102470	fc3d1daec319d62d49bfb5e1f83123e9	1	delivered	1.20	7.89	health_beauty	14	-4
102724	9eb3d566e87289dcb0acf28e1407c839	1	delivered	5.31	15.23	housewares	10	9

This chart shows an example of customers that gave olist a review score of 1.

Looking at the two charts, you can't distinguish between those that gave a high review score to those that gave a low review score.

Final Dataset for Model

I dropped all columns that were not needed for the Recency, Frequency, Monetary Analysis, and K-Means Clustering analysis.

The final dataset includes the following columns:

customer_id, order_id, order_purchase_timestamp, and the price.

	customer_id	order_purchase_timestamp	order_id	price
0	9ef432eb6251297304e76186b10a928d	2017-10-02 10:56:33	e481f51cbdc54678b7cc49136f2d6af7	29.99
1	a20e8105f23924cd00833fd87daa0831	2017-08-15 18:29:31	128e10d95713541c87cd1a2e48201934	29.99
2	26c7ac168e1433912a51b924fbd34d34	2017-08-02 18:24:47	0e7e841ddf8f8f2de2bad69267ecfbcf	29.99
3	53904ddbea91e1e92b2b3f1d09a7af86	2017-10-23 23:26:46	bfc39df4f36c3693ff3b63fcbea9e90a	29.99
4	b0830fb4747a6c6d20dea0b8c802d7ef	2018-07-24 20:41:37	53cdb2fc8bc7dce0b6741e2150273451	118.70
...	...	...	...	...
104777	609b9fb8cad4fe0c7b376f77c8ab76ad	2017-08-10 21:21:07	e8fd20068b9f7e6ec07068bb7537f781	356.00
104778	609b9fb8cad4fe0c7b376f77c8ab76ad	2017-08-10 21:21:07	e8fd20068b9f7e6ec07068bb7537f781	356.00
104779	a2f7428f0cafbc8e59f20e1444b67315	2017-12-20 09:52:41	cfa78b997e329a5295b4ee6972c02979	55.90
104780	39bd1228ee8140590ac3aca26f2dfe00	2017-03-09 09:54:05	9c5dedf39a927c1b2549525ed64a053c	72.00
104781	edb027a75a1449115f6b43211ae02a24	2018-03-08 20:57:30	66dea50a8b16d9b4dee7af250b4be1a5	68.50

104782 rows × 4 columns

RFM Analysis + K-Means Clustering

Recency, Frequency, and Monetary (RFM) Analysis look at data on historical customer behavior to predict how might a new customer act in the future. I will use RFM analysis to look at three key items:

1. How recently a customer has purchased with olist
2. How many orders did a customer purchase with olist
3. How much money a customer has spent with olist

Once I have the three key factors of recency, frequency, and monetary value, I will use this data to group the customers using K-Means Clustering. This will allow the company to target different groups of customers depending on how much they spend, how frequently they shop, and how many items they typically purchase.

RFM Analysis

Recency

To get recency, I found the most recent purchase date and calculated the number of days other customers had purchased from, and compared it to this date. I did this because the data I was using was from 2017 and 2018.

```
# Group dataset by customer id and get the max purchase date
recency_df = olist_df_model.groupby(by = 'customer_id', as_index = False)['order_purchase_timestamp'].max()
recency_df.rename(columns = {"order_purchase_timestamp": "last_purchase_date"}, inplace = True)
recency_df["last_purchase_date"] = pd.to_datetime(recency_df["last_purchase_date"])
recency_df.head()
```

```
# Get the most recent purchase date and use it to calculate number of days from this date on other rows
recent_date = olist_df_model['order_purchase_timestamp'].max()
recency_df['Recency'] = recency_df['last_purchase_date'].apply(lambda x: (recent_date - x).days)
```

```
recency_df.head()
```

	customer_id	last_purchase_date	Recency
0	00012a2ce6f8dcda20d059ce98491703	2017-11-14 16:08:26	259
1	000161a058600d5901f007fab4c27140	2017-07-16 09:40:32	380
2	0001fd6190edaaf884bcaf3d49edf079	2017-02-28 11:06:43	518
3	0002414f95344307404f0ace7a26f1d5	2017-08-16 13:09:20	349
4	000379cdec625522490c315e70c7a9fb	2018-04-02 13:42:17	120

Frequency

To get frequency, I totaled up the number of orders a customer had purchased with olist.

```
frequency_df = olist_df_model.groupby(by = 'customer_id', as_index = False)['order_id'].count()
frequency_df.columns = ['customer_id', 'Frequency']
frequency_df.head()
```

	customer_id	Frequency
0	00012a2ce6f8dcda20d059ce98491703	1
1	000161a058600d5901f007fab4c27140	1
2	0001fd6190edaaf884bcaf3d49edf079	1
3	0002414f95344307404f0ace7a26f1d5	1
4	000379cdec625522490c315e70c7a9fb	1

Monetary

To get monetary, I totaled up the amount of each order ordered from olist.

```
monetary_df = olist_df_model.groupby(by = 'customer_id', as_index = False)['price'].sum()
monetary_df.columns = ['customer_id', 'Monetary']
monetary_df.head()
```

	customer_id	Monetary
0	00012a2ce6f8dcda20d059ce98491703	89.80
1	000161a058600d5901f007fab4c27140	54.90
2	0001fd6190edaaf884bcaf3d49edf079	179.99
3	0002414f95344307404f0ace7a26f1d5	149.90
4	000379cdec625522490c315e70c7a9fb	93.00

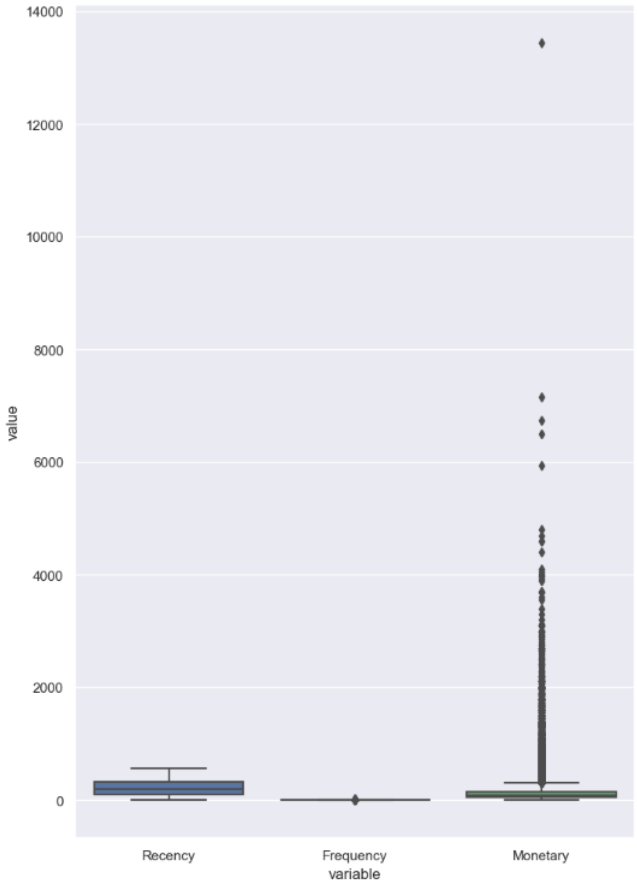
I then merged the recency, frequency, and monetary datasets into one and converted the customer_id to the index.

```
rf_df = recency_df.merge(frequency_df, on = 'customer_id')
rfm_df = rf_df.merge(monetary_df, on = 'customer_id').drop(columns = 'last_purchase_date')
```

```
rfm_df = rfm_df.set_index('customer_id')
rfm_df.head()
```

customer_id	Recency	Frequency	Monetary
00012a2ce6f8dcda20d059ce98491703	259	1	89.80
000161a058600d5901f007fab4c27140	380	1	54.90
0001fd6190edaaf884bcaf3d49edf079	518	1	179.99
0002414f95344307404f0ace7a26f1d5	349	1	149.90
000379cdec625522490c315e70c7a9fb	120	1	93.00

Examining the RFM dataset



The frequency values only had 1 value and the monetary values had a lot of outliers. I decided to drop the frequency values and used a standard scaler to normalize the recency and monetary values to be used in the k-means clustering model.

```
rm_df = rfm_df.drop('Frequency', axis = 1)
rm_df.head()
```

customer_id	Recency	Monetary
00012a2ce6f8dcda20d059ce98491703	259	89.80
000161a058600d5901f007fab4c27140	380	54.90
0001fd6190edaaf884bcaf3d49edf079	518	179.99
0002414f95344307404f0ace7a26f1d5	349	149.90
000379cdec625522490c315e70c7a9fb	120	93.00

```
standardizer = StandardScaler()
rm_scaled = standardizer.fit_transform(rm_df)
```

```
rm_scaled
```

```
array([[ 0.22988532, -0.2310531 ],
       [ 1.06757242, -0.39666251],
       [ 2.0229511 ,  0.19692147],
       ...,
       [-1.0785681 , -0.42987929],
       [-1.23087484,  0.2913995 ],
       [ 0.73526679, -0.55373045]])
```

Silhouette Score

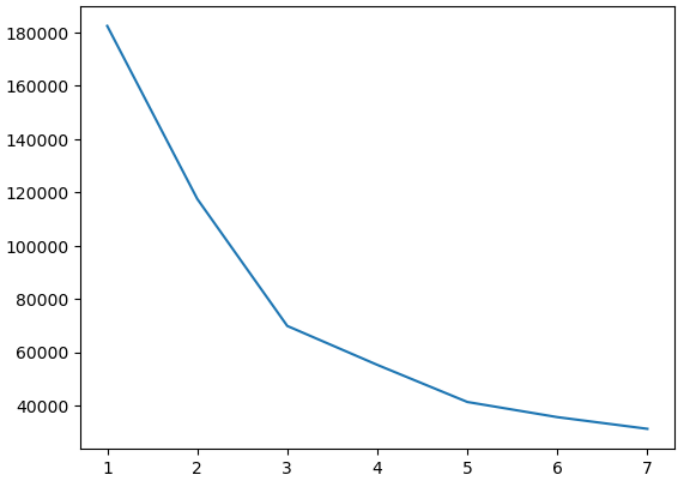
To help determine the optimal number of clusters for the k-means clustering model, I used the Silhouette Score. The silhouette score determines how similar an object is to its current cluster and the other clusters. Scores range from -1 to 1 where the higher value indicates that the object is similar to its own cluster.

```python
# Clusters for 2 - 6
range_n_clusters = [2, 3, 4, 5, 6]
silhouette_scores = []

for n_clusters in range_n_clusters:

    # Initialize the clusterer with n_clusters value and a random generator
    # seed of 10 for reproducibility.
    clusterer = KMeans(n_clusters = n_clusters, random_state=10)
    cluster_labels = clusterer.fit_predict(rm_scaled)

    # The silhouette_score gives the average value for all the samples.
    # This gives a perspective into the density and separation of the formed
    # clusters
    silhouette_avg = silhouette_score(rm_scaled, cluster_labels)
    silhouette_scores.append(silhouette_avg)

    print(f"For n_clusters = {n_clusters}. The average silhouette score is {silhouette_avg}")
```

```
For n_clusters = 2. The average silhouette score is 0.46731442416207347
For n_clusters = 3. The average silhouette score is 0.5010325943820231
For n_clusters = 4. The average silhouette score is 0.5111099455792638
For n_clusters = 5. The average silhouette score is 0.43112081689947196
For n_clusters = 6. The average silhouette score is 0.43634171713433667
```

Elbow method

The elbow method is another method to see how many clusters are optimal for clustering.



Looking at both the silhouette score and the elbow method, I used 3 as the optimal number of clusters.

K-Means Clustering

K-means clustering is an unsupervised machine learning method used to identify groups of data objects in a dataset. The k in k-means is the number of clusters we found from the silhouette analysis in the previous step. Here we will use 3 clusters in our model.

```
kmeans = KMeans(n_clusters = 3, max_iter = 50)
kmeans.fit(rm_scaled)
```

```
KMeans(max_iter=50, n_clusters=3)
```

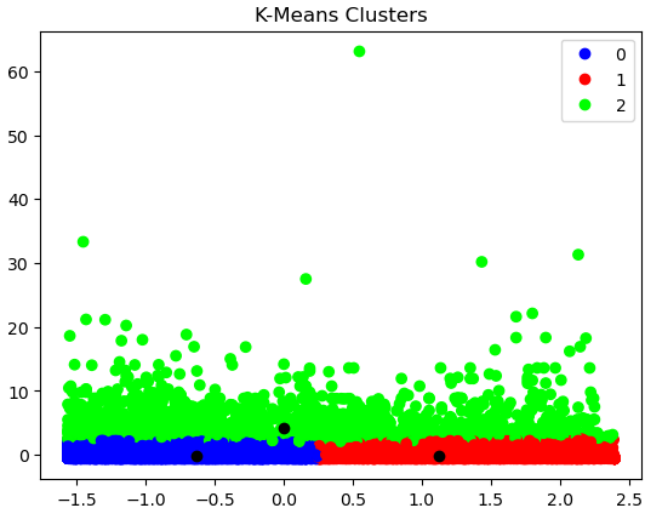
```
# Determine which clusters each data point belongs to
clusters = kmeans.predict(rm_scaled)
```

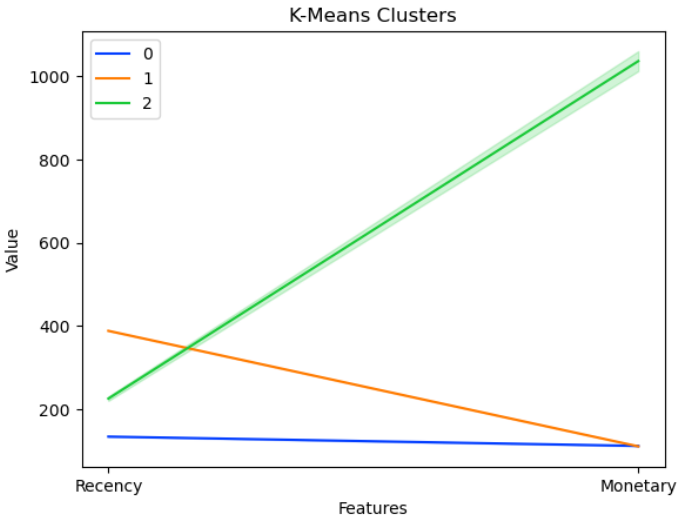
```
# Find the centers of each of the clusters
centers = kmeans.cluster_centers_
```

Convert to a dataset

```
# Add cluster number to the original data
rm_scaled_clustered = pd.DataFrame(rm_scaled, columns = rm_df.columns, index = rm_df.index)
rm_scaled_clustered['cluster'] = clusters

rm_scaled_clustered.head()
```

customer_id	Recency	Monetary	cluster
00012a2ce6f8dcda20d059ce98491703	0.229885	-0.231053	0
000161a058600d5901f007fab4c27140	1.067572	-0.396663	1
0001fd6190edaaf884bcaf3d49edf079	2.022951	0.196921	1
0002414f95344307404f0ace7a26f1d5	0.852958	0.054137	1
000379cdec625522490c315e70c7a9fb	-0.732416	-0.215868	0





Looking at the two graphs, we can see the differences between the different clusters.

- **Cluster 0**: Customers that haven't purchased items in a long time and their purchases amount to a very low dollar amount.
- **Cluster 1**: Customers that have purchased recently but their purchases amount to a very low dollar amount
- **Cluster 2**: These are customers that have purchased items somewhat recently but these customers' purchases amount to a very high dollar amount.

Conclusion

With these 3 groups of customers, the marketing team will be able to target advertising, coupons, and new products to each group individually to help drive higher profits, customer satisfaction with high review scores, and better customer service.

References

Arvai, K. (n.d.). *K-Means Clustering in Python: A Practical Guide*. Retrieved from Real Python: https://realpython.com/k-means-clustering-python/

Begam, S. (2021). *Customer Profiling and Segmentation – An Analytical Approach To Business Strategy In Retail Banking*. Retrieved from Analytics Vidhya: https://www.analyticsvidhya.com/blog/2021/03/customer-profiling-and-segmentation-an-analytical-approach-to-business-strategy-in-retail-banking/

Kaggle. (n.d.). *Brazilian E-Commerce Public Dataset by Olist*. Retrieved from https://www.kaggle.com/datasets/olistbr/brazilian-ecommerce

Pagan Research. (n.d.). Retrieved from olist: https://paganresearch.io/company/olist

Selvaraj, N. (2022). *How to Build Customer Segmentation Models in Python?* Retrieved from 365 Data Science: https://365datascience.com/tutorials/python-tutorials/build-customer-segmentation-models/

Wright, G. (n.d.). *RFM analysis (recency, frequency, monetary)*. Retrieved from TechTarget: https://www.techtarget.com/searchdatamanagement/definition/RFM-analysis