Contents

List of Figures

1 Background

With customers spanning multiple geographies and demographics, proposing an effective marketing strategy requires a thorough understanding of their needs. Customers often do not like being treated as one monolithic group but rather prefer a more personalized experience while making purchases. For such a use case, past purchase history of customer has proven to be an imperative source of information which if mined correctly can lay the foundation of many a successful marketing campaigns.

Understanding the customer spending habits has multiple advantages. For one, it can be used to group customers into different categories and recommend products based on such categories. Being able to predict the customer's future purchases can lead to increased customer retention with an increased net cart value. Second, it can also be used to mail discount coupons and vouchers just before an expected purchase is to be made(say before the holiday season begins) to draw more repeated customers. Third, it can be used to recommend related products when other products of similar categories are already added to the cart e.g. recommending a shaving brush on adding a shaving cream to the cart. Lastly, having a record of past purchases can help in better inventory management for the business as they restock the highest selling products more often than the less popular ones.

As with any such exercise, a crucial question of ethicality of the whole process often arises. Is it morally correct to nudge a customer (often without his/her knowledge) to be influenced by a product recommendation and make an otherwise unrelated purchase? Do customers like being prodded on to purchase related products and do they have the option of turning them down? Is it right for a business to get the customers to spend more overall on their store just to increase the annual business revenue? All these questions and more give rise to a cogent argument against such a practice which is so widespread across online businesses. That being said, the scope of this project is just to analyze the different ways in which customer segmentation can be performed based on available data. It is up to the individual businesses to ensure that customers are not exploited based on their past choices and have some degree of control over what data the customer shares with the business. Only that can help alleviate the ethical downsides of this practice to an extent.

2 Customer Segmentation : An Ethical Angle

The field of Data Analytics with its ubiquitous data access raises rather poignant ethical dilemmas. For a data scientist, profiling the customer based on the spending habits seems like a rather harmless activity. However, from the customer's perspective it appears a tad bit exploitative and falls within the realm of a privacy breach. In order to resolve such conflicts, we must be able to come up with a sound rationale on why the analysis of data is essential and to what extent is it justifiable while respecting the privacy of the customer.

2.1 POTIO Principles

The following principles encompass the ethical obligations of the data owning entity while dealing with customer data:

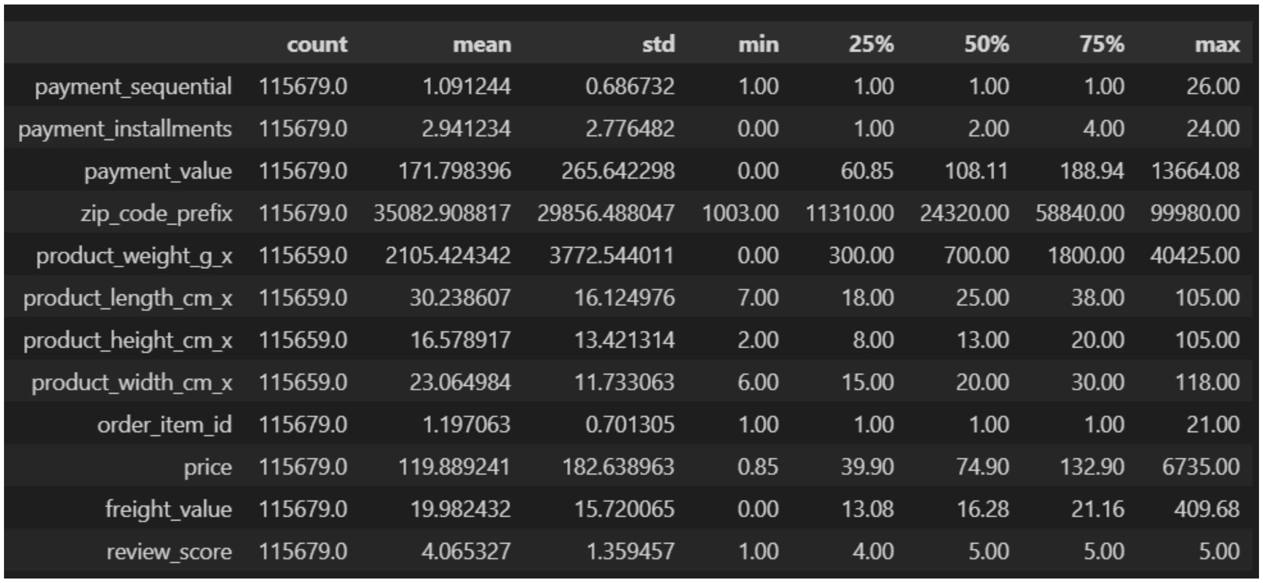
- Privacy : The business should take measures to ensure the customer's privacy. Customer must be asked to opt-in to the privacy terms with a choice to refuse storage of their data should they wish to. Customer should be regularly updated about the Privacy policy changes, if any. Appropriate security measures must be taken to prevent potential data breach. Datasets must be scrubbed for any traces of PII prior to storage.
- Ownership : Explicit consent must be taken from the customer before assuming ownership of their data. If the customer refuses the agreement, then it should be honored by the business.
- Transparency : Customers have the right to know how their data is being stored, collected and used. It must be mentioned in the privacy policy of the business, a document which must be easily accessible to the customer. Withholding such information can be deemed unlawful and / or unethical.
- Intent : Intentions form the basis of many ethical theories while deciding if an action is deemed ethical or otherwise. Deontology is one such example where the intent of the action is more important and decisive of the act being ethical rather than the outcome. If the intent is just, then the act can be deemed as ethical. If a business is storing transactional data, then the purpose must be restricted to creating user personas and high level marketing strategies. It must not be used to hurt others, persuade them psychologically or exploit their weaknesses. Every piece of data which is stored must have an explicit reason behind it and not just for the sake of having more information on the customer.
- Outcome : Another popular ethical theory(Utilitarianism) attributes greater importance to the outcome of the act rather than the intent. If the outcome of performing technical data analysis proves to be helpful for the customers e.g. by providing better recommendations and offers for future purchases, then it can be considered an ethical one.

3 Data preprocessing

This dissertation utilises an anonymized Brazilian ecommerce public dataset of orders made at Olist Store. The dataset has information of 100k orders from 2016 to 2018 made at multiple marketplaces in Brazil. It is split across several smaller tables e.g. customers dataset, orders dataset, order payments dataset, products dataset etc. Each of them providing a different lens through which to look at the data. In order to understand customer spending behavior and cluster them, we will combine the data from all of these tables based on order_id and product_id incrementally.

```python
#payment merging with order
payment_order     = pd.merge(order_payments,orders,on="order_id")

#product merging with item
product_item     = pd.merge(order_products,order_items,on="product_id")
```



	count	mean	std	min	25%	50%	75%	max
payment_sequential	115679.0	1.091244	0.686732	1.00	1.00	1.00	1.00	26.00
payment_installments	115679.0	2.941234	2.776482	0.00	1.00	2.00	4.00	24.00
payment_value	115679.0	171.798396	265.642298	0.00	60.85	108.11	188.94	13664.08
zip_code_prefix	115679.0	35082.908817	29856.488047	1003.00	11310.00	24320.00	58840.00	99980.00
product_weight_g_x	115659.0	2105.424342	3772.544011	0.00	300.00	700.00	1800.00	40425.00
product_length_cm_x	115659.0	30.238607	16.124976	7.00	18.00	25.00	38.00	105.00
product_height_cm_x	115659.0	16.578917	13.421314	2.00	8.00	13.00	20.00	105.00
product_width_cm_x	115659.0	23.064984	11.733063	6.00	15.00	20.00	30.00	118.00
order_item_id	115679.0	1.197063	0.701305	1.00	1.00	1.00	1.00	21.00
price	115679.0	119.889241	182.638963	0.85	39.90	74.90	132.90	6735.00
freight_value	115679.0	19.982432	15.720065	0.00	13.08	16.28	21.16	409.68
review_score	115679.0	4.065327	1.359457	1.00	4.00	5.00	5.00	5.00

Figure 1: Numerical features with their individual statistics

Apart from the de-duplication of tuples and removal of null values in each columns, we performed the following:

- The given product names were in Portuguese, so we translated them to English
- We filter for only those orders which were successfully delivered
- Removed non relevant columns containing seller information and product review comments

Thus, we were left with the following numerical features listed with their statistics:

4 Exploratory Data Analysis

We will now look at some of the observations derived from the final merged dataset to further segregate and understand our customers.

4.0.1 Customer demographics

```python
#Let us find out where our customers are located
from matplotlib.pyplot import figure
figure(figsize=(8, 6), dpi=80)
ax = final_data["customer_state"].value_counts().plot(kind='bar')
ax.set_yscale('log')
```

4.0.2 Highest Spenders

```python
# Let's see where do the highest spenders come from
cat=final_data[["customer_state", "payment_value"]].groupby(['customer_state'],
↪  as_index=False).sum().sort_values(by='payment_value', ascending=False)
```

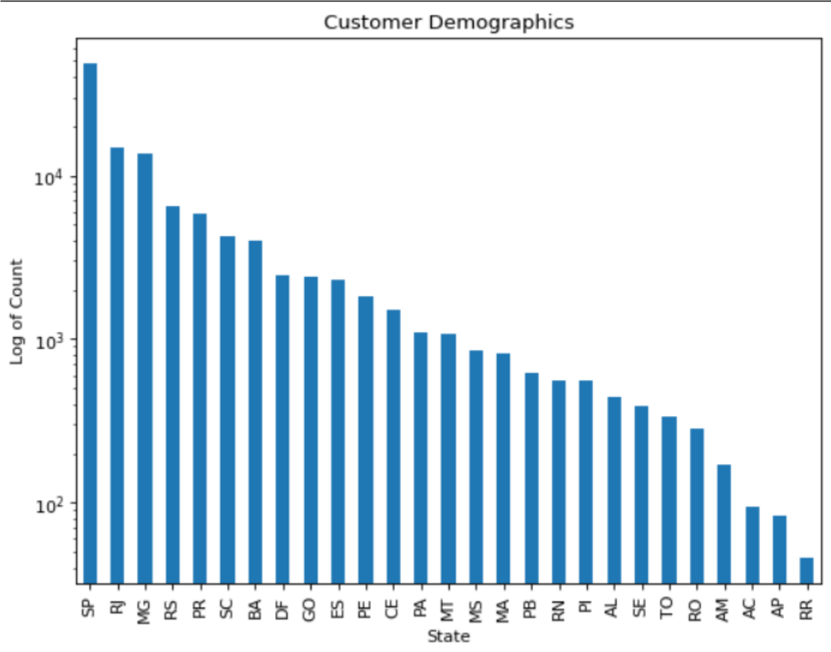


Figure 2: Customer distribution across states

```
plt.figure(figsize=(10,5))
sns.barplot(x='customer_state', y='payment_value', data=cat)
```

4.0.3 Product Popularity

```
top_products = final_data['product_category_name'].value_counts()
.reset_index().nlargest(10,'product_category_name')
lowest_products = final_data['product_category_name'].value_counts()
.reset_index().nsmallest(10,'product_category_name')
plt.figure(figsize = (15,12))
red_color = sns.color_palette()[3]
green_color = sns.color_palette()[2]

plt.subplot(211)
sns.barplot(data = top_products, x = 'product_category_name', y = 'index', color =
↪  green_color)
plt.title('Top 10 Products Ordered')
plt.xlabel('Number of Orders')
plt.ylabel('Product');

plt.subplot(212)
sns.barplot(data = lowest_products, x = 'product_category_name', y = 'index', color
↪  = red_color)
plt.title('Lowest 10 Products Ordered')
```

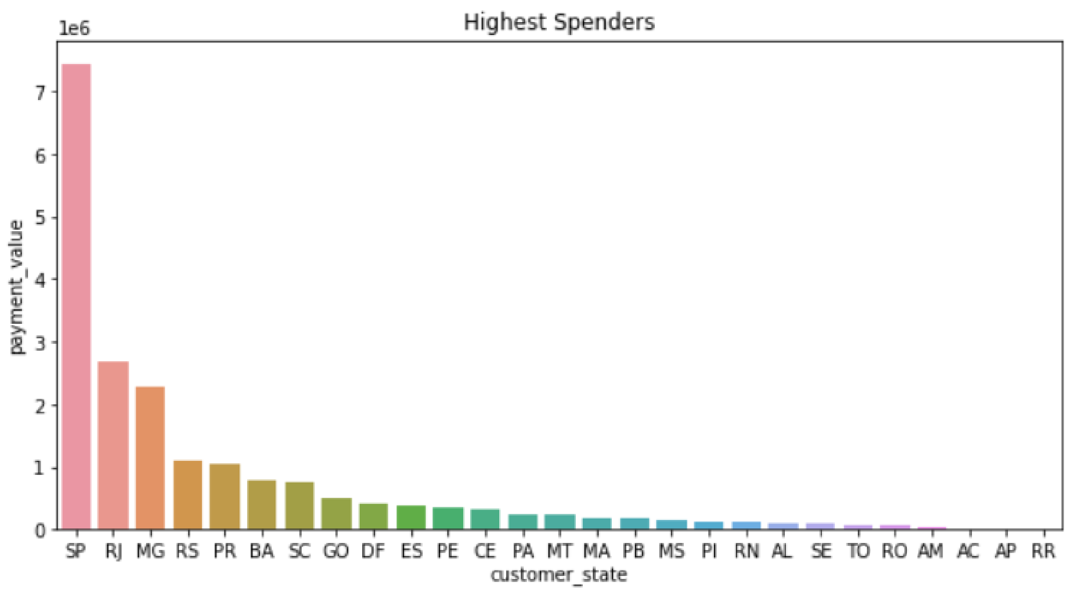


Figure 3: Highest spending customers by State

```
plt.xlabel('Number of Orders')
plt.ylabel('Product')
```

4.0.4 Payments and Popular shopping days

```
sns.countplot(data=final_data, x="payment_type").set_title('Mode of Payments')
```

```
sns.countplot(data=final_data, x="order_day", order=["Sunday", "Monday", "Tuesday",
↪  "Wednesday", "Thursday", "Friday", "Saturday"]).set_title('Order Days')
```

Some of the valuable insights obtained from this preliminary analysis include:

- Majority of customers reside in Sao Paolo and Rio De Janeiro(2 of the most populous states in Brazil)
- Even the top spenders tend to be from the above two major states
- The most popular mode of purchase is Credit Card
- Product orders tend to be the highest on Mondays and Tuesdays and gradually taper down toward the weekend

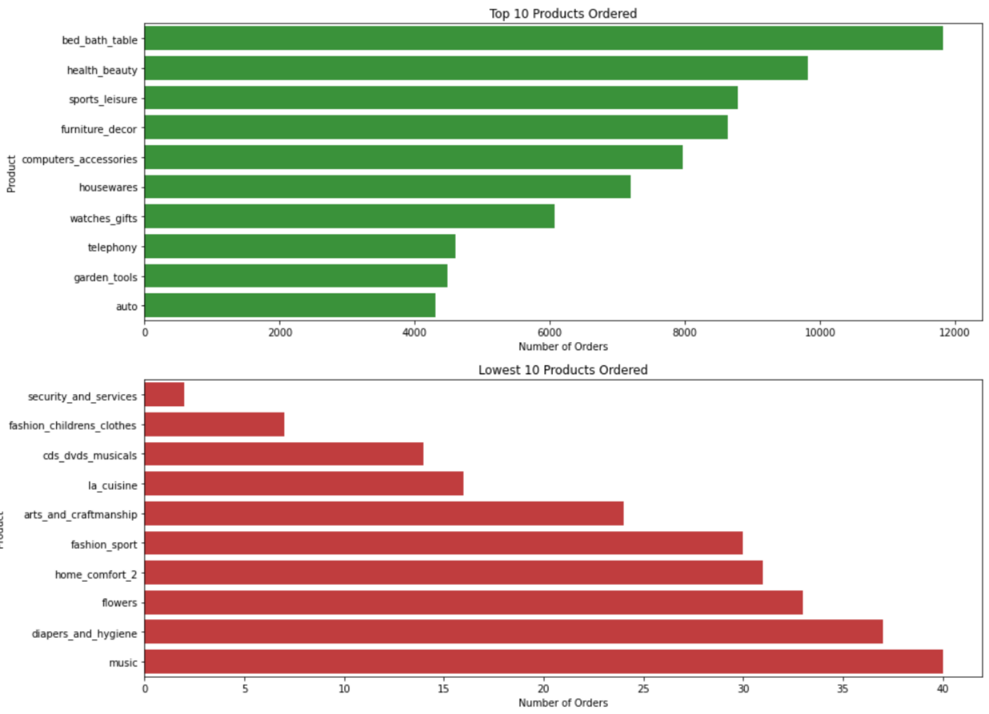


Figure 4: Most and Least popular product categories

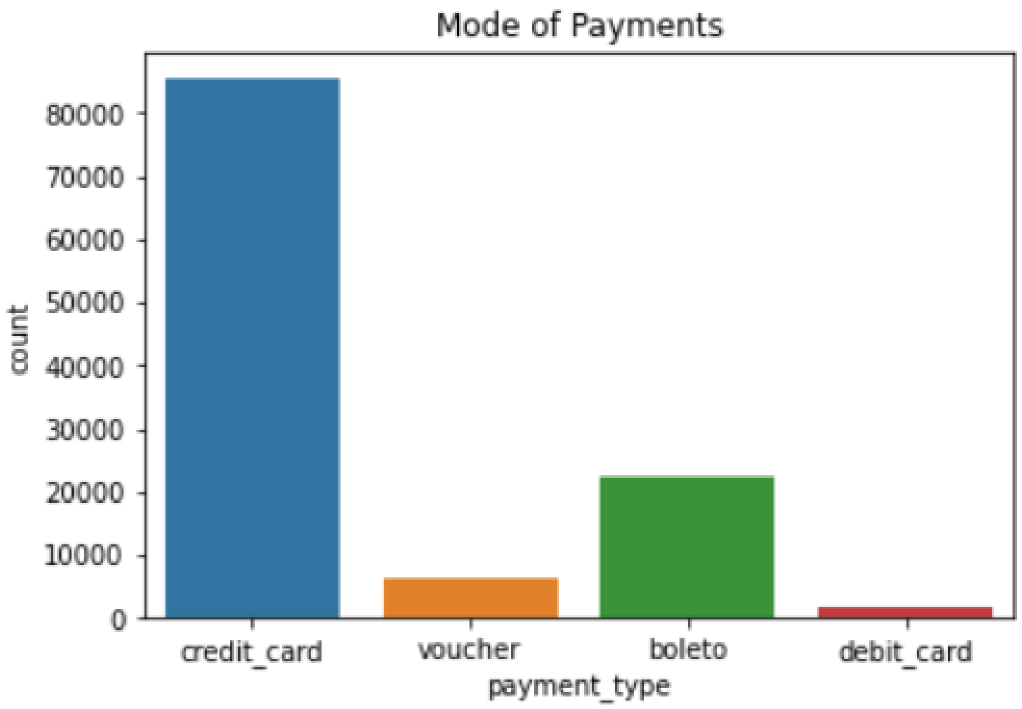


Figure 5: Popular Payment Modes by count

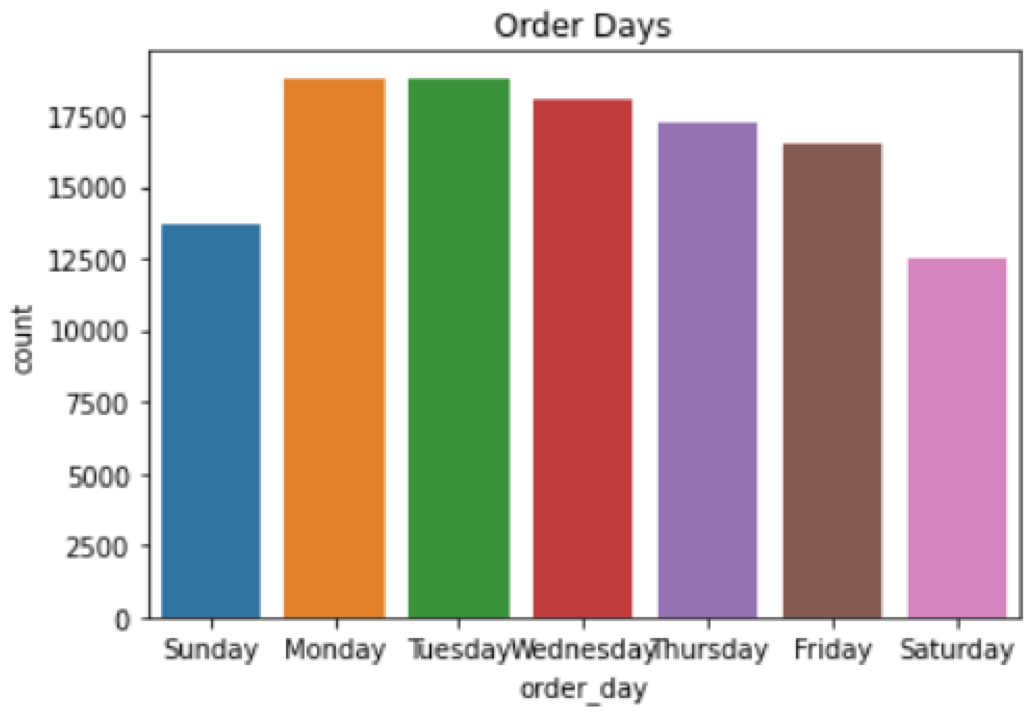


Figure 6: Number of orders per Weekday

5 RFM Model

Clustering algorithms are used to group a set of objects in such a way that objects within the same cluster are more similar to each other and more dissimilar from those in other clusters.In this dissertation, we will first use a very popular model of customer segmentation called RFM Model and then apply KMeans algorithm on the scaled RFM Data to identify clusters.

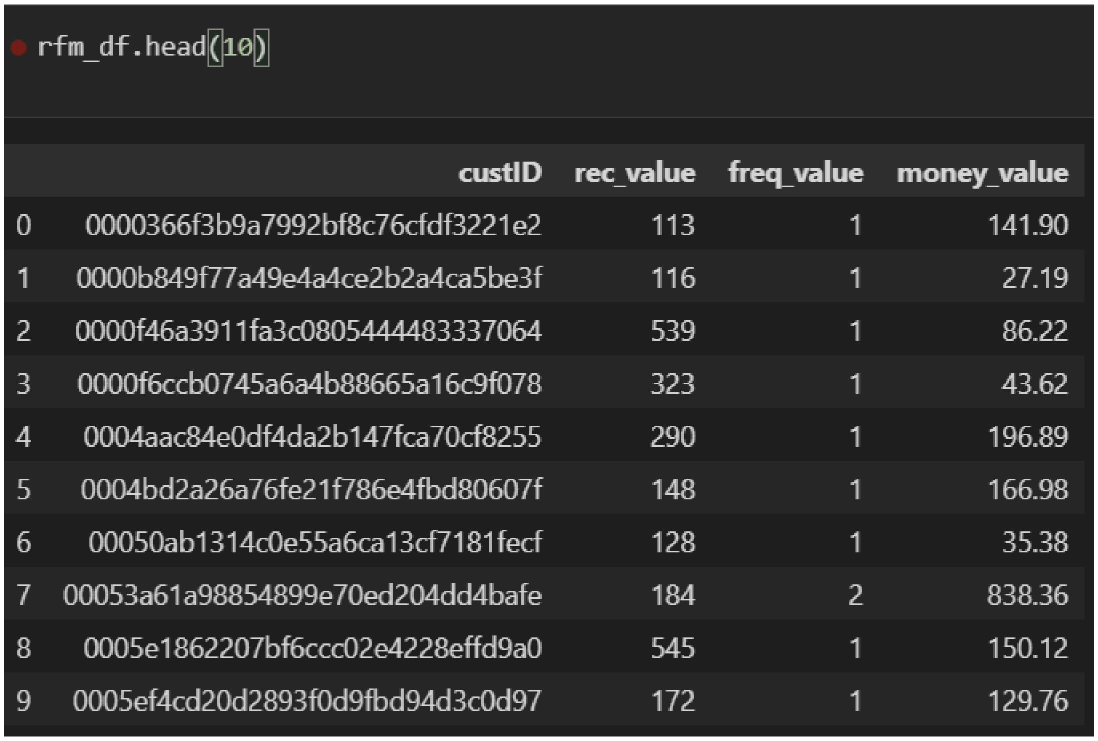
RFM involves classifying customers based on their Recency, Frequency and Monetary values. Recency refers to the number of days between the customer latest purchase and the current date. Frequency refers to how often has he/she placed an order throughout the observation period and Monetary value is obtained from the cumulative sum of the purchases made throughout the period. I have taken Sept 1st, 2018 to be the value for current date since the dataset is from 2018 and the last order is from August 28th.

```
#For RFM , we try to fetch the latest date at which each customer performed a
↪  transaction, this gives you the R(Recency)

t1 = final_data.groupby(['customer_unique_id'])
d={}
findate = datetime.strptime('2018-09-01 00:00:00', '%Y-%m-%d %H:%M:%S')
for name,group in t1:
    finval = 10000
    for index, value in group['order_approved_at'].items():
            if name in d:
                finval = (findate - value).days
                if(finval<d[name]):
                    d[name]=finval
            else:
                d[name]=(findate - value).days

#Next we will calculate the F(Frequency)
frq={}
uvals = final_data['customer_unique_id'].value_counts()
for index,value in uvals.items():
    frq[index]=value

#Lastly, the M(Monetory) value is calculated for each customer
t2 = final_data.groupby(['customer_unique_id'])
m1 = {}
for name,group in t2:
    sum=0
    for index,value in group['payment_value'].items():
        sum+=value
    m1[name]=sum
```



```
rfm_df.head(10)
```

	custID	rec_value	freq_value	money_value
0	0000366f3b9a7992bf8c76cfdf3221e2	113	1	141.90
1	0000b849f77a49e4a4ce2b2a4ca5be3f	116	1	27.19
2	0000f46a3911fa3c0805444483337064	539	1	86.22
3	0000f6ccb0745a6a4b88665a16c9f078	323	1	43.62
4	0004aac84e0df4da2b147fca70cf8255	290	1	196.89
5	0004bd2a26a76fe21f786e4fbd80607f	148	1	166.98
6	00050ab1314c0e55a6ca13cf7181fecf	128	1	35.38
7	00053a61a98854899e70ed204dd4bafe	184	2	838.36
8	0005e1862207bf6ccc02e4228effd9a0	545	1	150.12
9	0005ef4cd20d2893f0d9fbd94d3c0d97	172	1	129.76

Figure 7: Snapshot of Dataframe contents

We create a new dataframe by combining these three metrics along with custID as seen above.

In order to scale the R/F/M values, we will divide them into 4 intervals and label each value with an integer identifying the interval which it belongs to. Next, we will combine the R/F/M values to get a scaled RFM Score.

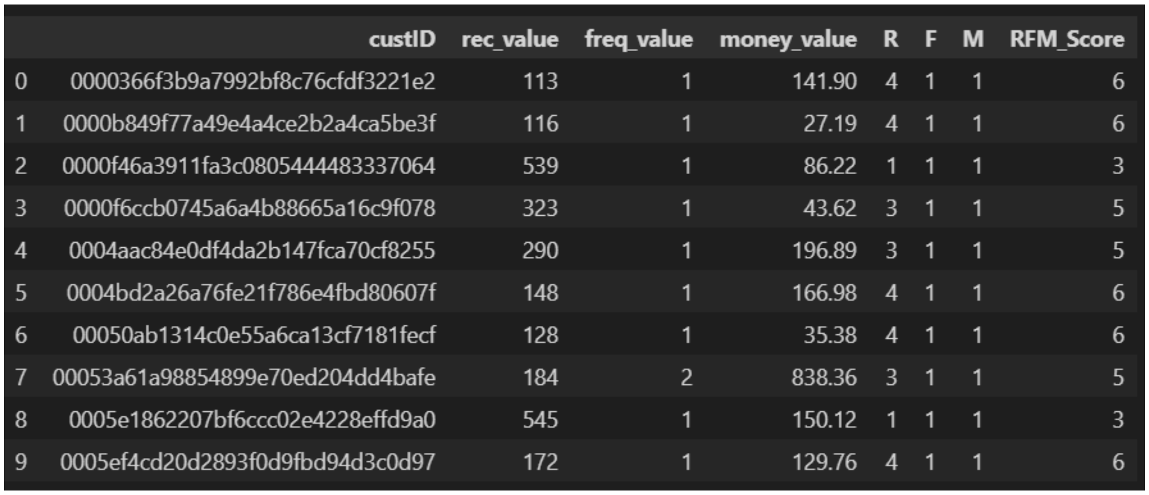
```
#Recency
rec_labels = range(4,0,-1)
rec_groups = pd.cut(rfm_df.rec_value, 4, labels = rec_labels)

#Frequency
freq_labels = range(1,5)
freq_groups = pd.cut(rfm_df.freq_value, 4, labels = freq_labels)

#Monetary
money_labels = range(1,5)
money_groups = pd.cut(rfm_df.money_value, 4, labels = money_labels)

rfm_df['R'] = rec_groups.values
rfm_df['F'] = freq_groups.values
rfm_df['M'] = money_groups.values

rfm_df['RFM_Score'] = rfm_df[['R', 'F', 'M']].sum(axis = 1)
rfm_df.head(10)
```



	custID	rec_value	freq_value	money_value	R	F	M	RFM_Score
0	0000366f3b9a7992bf8c76cfdf3221e2	113	1	141.90	4	1	1	6
1	0000b849f77a49e4a4ce2b2a4ca5be3f	116	1	27.19	4	1	1	6
2	0000f46a3911fa3c0805444483337064	539	1	86.22	1	1	1	3
3	0000f6ccb0745a6a4b88665a16c9f078	323	1	43.62	3	1	1	5
4	0004aac84e0df4da2b147fca70cf8255	290	1	196.89	3	1	1	5
5	0004bd2a26a76fe21f786e4fbd80607f	148	1	166.98	4	1	1	6
6	00050ab1314c0e55a6ca13cf7181fecf	128	1	35.38	4	1	1	6
7	00053a61a98854899e70ed204dd4bafe	184	2	838.36	3	1	1	5
8	0005e1862207bf6ccc02e4228effd9a0	545	1	150.12	1	1	1	3
9	0005ef4cd20d2893f0d9fbd94d3c0d97	172	1	129.76	4	1	1	6

Figure 8: RFM Values bucketized to get RFM Score

5.1 Customer Segment Nomenclature

The model allows us to label customers into different tiers according to their spending habits. For the purpose of this project, I have chosen the following nomenclature for the various customer segments based on their overall RFM Score:

- Standard - These customers have purchased products recently but their overall cart value is less per order
- Premium - These customers are less recent than Standard but their cart values are higher than Standard per order
- Premium Plus - These customers are even less recent than Premium but with transaction values higher than Standard per order
- Ultra Premium - These are our top spenders spanning across a range of time periods

We have 93316 customers in total and this is how their individual counts look like:

```
RFM_score_labels = ['Standard','Premium','Premium Plus','Ultra Premium']
RFM_score_groups = pd.cut(rfm_df.RFM_Score, 4, labels = RFM_score_labels)
rfm_df['RFM_Level'] = RFM_score_groups.values

rfm_df['RFM_Level'].value_counts()
"""
Premium Plus     36682
Premium          33554
Standard         23071
Ultra Premium        9
"""
```

5.2 K-Means Algorithm

K-Means algorithm is one of the most popular unsupervised clustering algorithms. It is a Centroid-based approach where we identify an initial set of clusters and incrementally measure the distance between this centroid and all other points in that cluster. Cluster boundaries are refined in each iteration and new centroids identified.

5.2.1 Elbow Method

Elbow method is one of the most popular method used to select the optimal number of clusters by fitting the model with a range of values for K in K-means algorithm. Elbow method requires drawing a line plot between SSE (Sum of Squared errors) vs number of clusters and finding the point representing the "elbow point" (the point after which the SSE or inertia starts decreasing in a linear fashion). We are going to perform a log transformation on the R/F/M values and then apply Elbow method to find the optimal number of clusters, which turns out to be 4 in our case.

5.2.2 Silhouette Score

Silhouette Score is a metric which is used to determine how similar are the individual data points between clusters. The silhouette ranges from -1 to +1, where a high value indicates that the object is well matched to its own cluster and poorly matched to neighboring clusters.

5.2.3 Calinski-Harabasz Index

The Calinski-Harabasz Index is the ratio of the Within Cluster Dispersion(WCD) and Between Cluster Dispersion(BCD) for all clusters. The higher the value , the better the clustering performance.

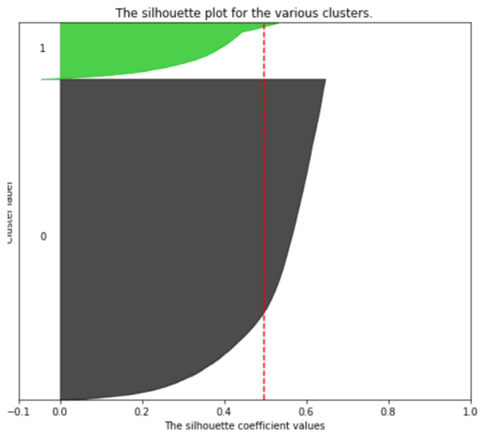
5.2.4 Davies Bouldin Score

The score is defined as the average similarity measure of each cluster with its most similar cluster. Similarity is defined as the ratio of within cluster distances to between cluster distances.Thus, the clusters which are farther apart and less dispersed will result in a better score. Lower values indicate better clustering.

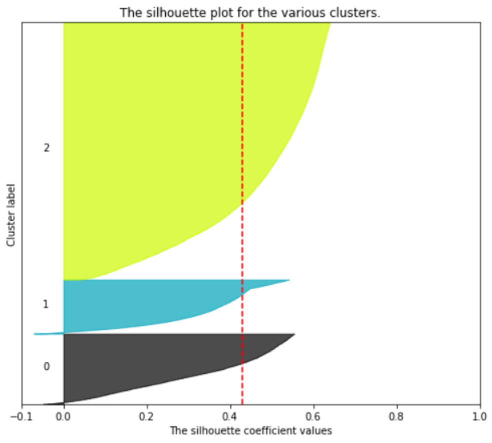
```
"""
For n_clusters = 2 The average silhouette_score is : 0.49767781727402205
CHIndex :  48533.262078010244
DB Score : 0.9486215753695908
For n_clusters = 3 The average silhouette_score is : 0.42916625965237104
CHIndex :  54865.40986568162
DB Score : 0.906773487694912
For n_clusters = 4 The average silhouette_score is : 0.3592548589494817
CHIndex :  58146.44457658749
DB Score : 0.8909444832255029
For n_clusters = 5 The average silhouette_score is : 0.3652525436330711
CHIndex :  53994.37641278736
DB Score : 0.9927396304663049
"""
```

5.2.5 Observation

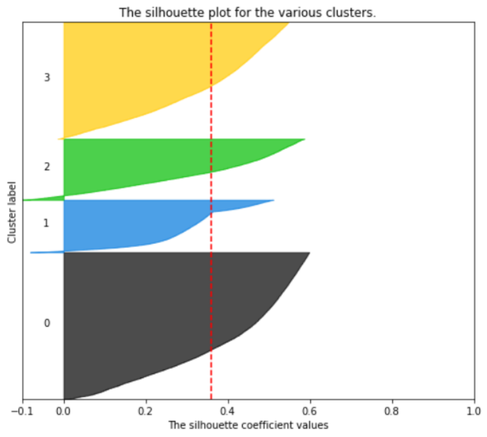
- As the number of clusters increases, the Silhouette score decreases until the number is 4 and then rises
- The CHI is the highest when the number of clusters is 4
- The DBS is lowest when the number of clusters is 4
- The ideal number of clusters should be 4 which provides the best separation of points and cluster boundaries



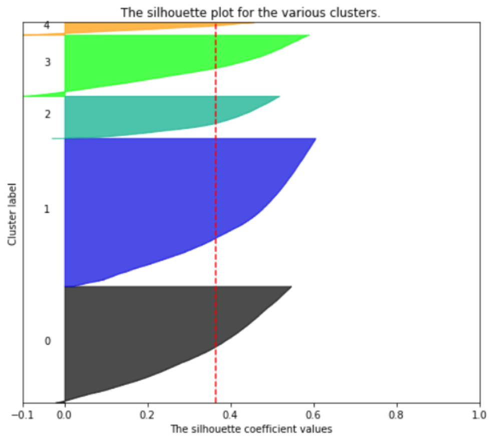
(a) Number of clusters = 2



(b) Number of clusters = 3



(c) Number of clusters = 4



(d) Number of clusters = 5

Figure 9: A Visualization of clusters with the red line depicting the average value for the Silhouette Score across all clusters

- If we consider the Silhouette Score separately then the ideal cluster count appears to be 2 but this observation is invalidated by the fact that both CH Index and DB score are ideal for a cluster count of 4
- Using a single metric for evaluating cluster performance can provide a biased result and therefore using a combination of atleast 3 different evaluation measures provides a clearer picture of the performance

5.3 K-Means Clustering Vs RFM Model

The RFM Model provides a straightforward approach to divide the customers based on the three parameters only viz. Recency, Frequency and Monetary value. In this case, we chose to calculate the RFM Score after dividing the values into 4 buckets explicitly which resulted in 4 tiers. However, the same calculation could have been done with more than 4 buckets as well and we would have arrived at as many clusters. Thus, RFM model is a great starting point to begin exploring customer segmentation.
Incidentally, the reliance of the model on specific parameters prevents it from capturing the nuance of large datasets with multiple other features(similar to the one we are dealing with now). This is where K-Means excels. Although, we saw the ideal number of clusters in both the algorithms to be the same value(=4) in this case, we can still explore the dataset further with K-Means and other clustering algorithms to better understand the distribution of customer segments based on more features. Let us look at some more of these in the next section.

6 Other Clustering Algorithms

We will now explore what other features can be considered from the given dataset for creating the model apart from the RFM Values.

6.1 Feature Engineering

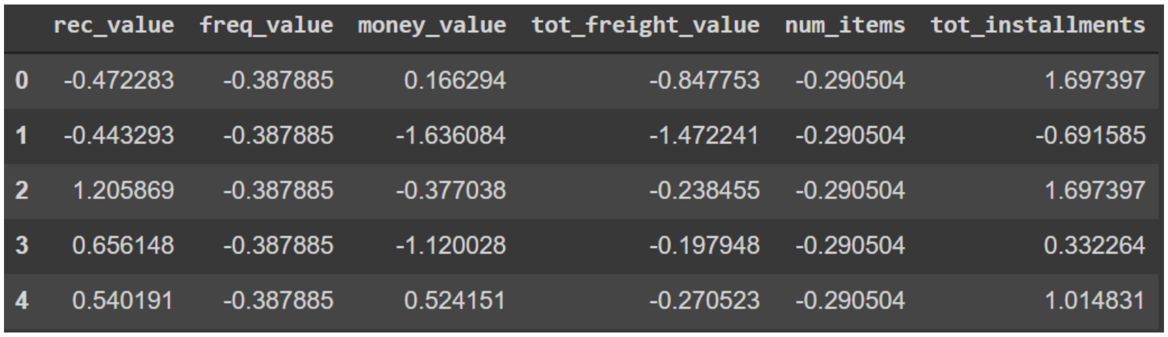
Here is a brief explanation on why we chose the next set of features and dropped the rest:

- **Product Category Name** : This cannot be considered because the dataset is for orders and several customers have ordered from multiple categories over the span of time
- **Customer City** : This also cannot be considered because the same customer has ordered from different cities for different orders
- **Product Dimensions(W/H/L)** : Products from different categories can have similar dimensions and therefore might lead to an incorrect inference so we dropped this as well
- **Payment installments** : This feature was chosen because the higher the number of payment installments, the higher would be the monetory value for the customer
- **Number of items ordered** : This feature was chosen because it draws parallel to the overall transaction value per order
- **Freight Value** : This feature also was found to be a major contributor to the revenue as the size of the orders increased so it was selected

We calculate the payment installments, items ordered and Freight value per customer to add to the RFM model as follows:

```
#Only 0.04% of items dont have freight attached so a significant portion do have it
↪  and we cant ignore that

numitems =
↪  final_data.groupby(['customer_unique_id','order_id']).max('order_item_id')
numinstalls =
↪  final_data.groupby(['customer_unique_id','order_id']).max('payment_installments')
numfreight = final_data.groupby(['customer_unique_id']).sum('freight_value')
numitems.loc['9a736b248f67d166d2fbb006bcb877c3']['order_item_id'].sum()
numfreight.loc['9a736b248f67d166d2fbb006bcb877c3']['freight_value']
numinstalls.loc['9a736b248f67d166d2fbb006bcb877c3']['payment_installments'].sum()
```



	rec_value	freq_value	money_value	tot_freight_value	num_items	tot_installments
0	-0.472283	-0.387885	0.166294	-0.847753	-0.290504	1.697397
1	-0.443293	-0.387885	-1.636084	-1.472241	-0.290504	-0.691585
2	1.205869	-0.387885	-0.377038	-0.238455	-0.290504	1.697397
3	0.656148	-0.387885	-1.120028	-0.197948	-0.290504	0.332264
4	0.540191	-0.387885	0.524151	-0.270523	-0.290504	1.014831

Figure 10: RFM Model with added features - A sample

```
#Calculating total items ordered
itemcount={}
custs = final_data['customer_unique_id'].value_counts()
for index,value in custs.items():
    itemcount[index]=numitems.loc[index]['order_item_id'].sum()

#Calculating total freight value
freightcount={}
for index,value in custs.items():
    freightcount[index]=numfreight.loc[index]['freight_value']

#Calculating payment installments
pymt_installs_total={}
for index,value in custs.items():
    pymt_installs_total[index]=numinstalls.loc[index]['payment_installments'].sum()
```

We will now apply log transformation and scaling to these features as follows:

```
rfm_log_df = fin_rfm_df[['rec_value', 'freq_value', 'money_value',
↪  'tot_freight_value']].apply(np.log, axis = 1).round(3)
rfm_log_df['num_items'] = fin_rfm_df['num_items']
rfm_log_df['tot_installments'] = fin_rfm_df['tot_installments']

# scale the data
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
rfm_scaled_df = scaler.fit_transform(rfm_log_df)
# transform into a dataframe
rfm_scaled_df = pd.DataFrame(rfm_scaled_df, index = fin_rfm_df.index, columns =
↪  rfm_log_df.columns)
```

Here is how the model looks like after adding the new features:

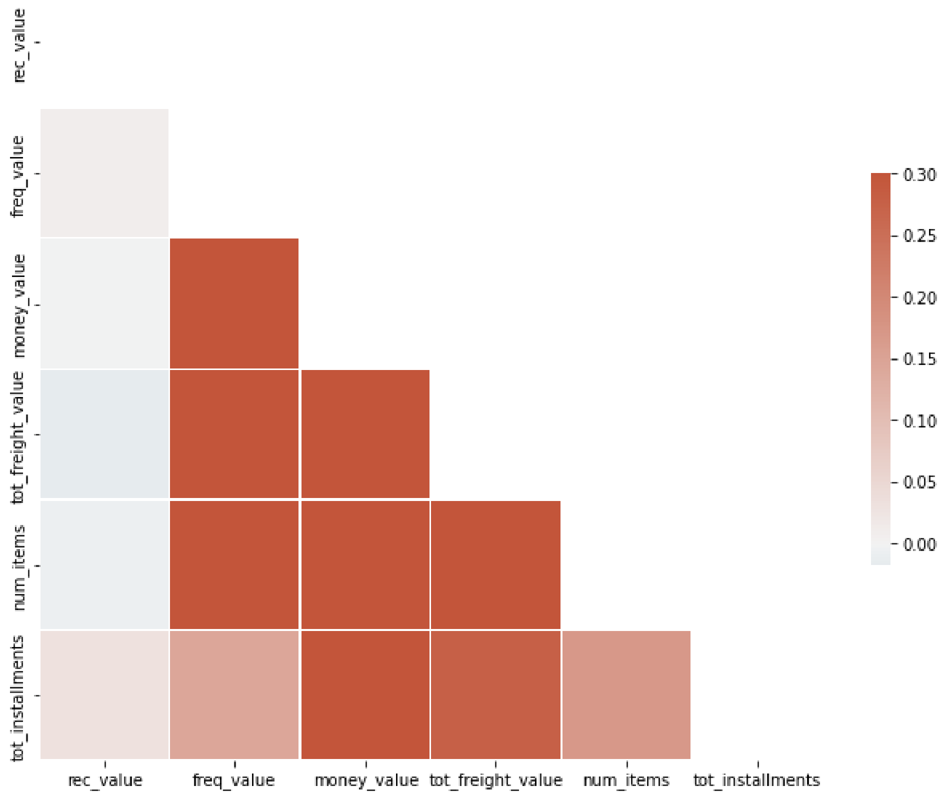


Figure 11: Correlation matrix of the enhanced RFM Model features

6.1.1 Feature Correlation

Let us first check the correlation between the new features to ensure that they are not highly correlated, in which case we may have to drop some of the features:

From the above fig, we do not see any highly correlated features, so we are okay to proceed with all of them in the model.

6.2 DBSCAN

DBSCAN is a density based clustering algorithm which considers points huddled closely together(or densely packed) to belong to the same cluster. A Core point has atleast MinPts number of points around its Eps-neighbourhood. A border point is itself within the Eps-neighbourhood of a Core Point but has less than MinPts number of points within its Eps-neighbourhood. A noise point is neither a Core nor a Border Point.
MinPts refers to Minimum number of points to qualify Core points. Eps-neighbourhood refers to the distance(referred as Epsilon) within which we consider the total number of points to determine Core and Border points.
MinPts and Eps values can be arbitrarily set and tweaked until we get appropriately spaced out clusters. DBSCAN also takes into account Noise points which are treated the same as other points in case of K-Means Algorithm.
Applying DBSCAN on our new model with 6 features:

```
from sklearn.cluster import DBSCAN
from sklearn import metrics
from sklearn.datasets import make_blobs
from sklearn.preprocessing import StandardScaler

db = DBSCAN(eps=0.5, min_samples=50).fit(rfm_scaled_df)
core_samples_mask = np.zeros_like(db.labels_, dtype=bool)
core_samples_mask[db.core_sample_indices_] = True
labels = db.labels_

# Number of clusters in labels, ignoring noise if present.
n_clusters_ = len(set(labels)) - (1 if -1 in labels else 0)
n_noise_ = list(labels).count(-1)

print('Estimated number of clusters: %d' % n_clusters_)
print('Estimated number of noise points: %d' % n_noise_)
print("Silhouette Coefficient: %0.3f"
      % metrics.silhouette_score(rfm_scaled_df, labels))
print("CHIndex : ",metrics.calinski_harabasz_score(rfm_scaled_df, labels))
from sklearn.metrics import davies_bouldin_score
print("DB Score :",davies_bouldin_score(rfm_scaled_df, labels))

#Output
Estimated number of clusters: 4
Estimated number of noise points: 11160
Silhouette Coefficient: 0.220
CHIndex :  10406.866912053476
DB Score : 2.0565158284637035
```

6.3 OPTICS

Ordering points to identify the clustering structure (OPTICS) is another density based clustering algorithm. It addresses a major weakness of DBSCAN which involves detecting meaningful clusters in data of varying density. It starts with finding core samples of high density and expanding clusters from them.
Applying OPTICS to our model:

```
from sklearn.cluster import OPTICS
from sklearn import metrics

db = OPTICS(min_samples=70).fit(rfm_scaled_df)
labels = db.labels_

# Number of clusters in labels, ignoring noise if present.
n_clusters_ = len(set(labels)) - (1 if -1 in labels else 0)
n_noise_ = list(labels).count(-1)

print('Estimated number of clusters: %d' % n_clusters_)
print('Estimated number of noise points: %d' % n_noise_)
print("Silhouette Coefficient: %0.3f"
      % metrics.silhouette_score(rfm_scaled_df, labels))
print("CHIndex : ",metrics.calinski_harabasz_score(rfm_scaled_df, labels))
from sklearn.metrics import davies_bouldin_score
print("DB Score :",davies_bouldin_score(rfm_scaled_df, labels))

#OPTICS with MinSamples = 5
Estimated number of clusters: 4738
Estimated number of noise points: 54974
Silhouette Coefficient: -0.322
CHIndex :  8.429050135215213
DB Score : 1.3708174831498234

#OPTICS with MinSamples = 10
Estimated number of clusters: 1175
Estimated number of noise points: 73634
Silhouette Coefficient: -0.557
CHIndex :  12.68121611722482
DB Score : 1.3922428959834339

#OPTICS with Minsamples = 70
Estimated number of clusters: 3
Estimated number of noise points: 92092
Silhouette Coefficient: -0.362
CHIndex :  257.57732878262345
DB Score : 1.5979244916561808
```

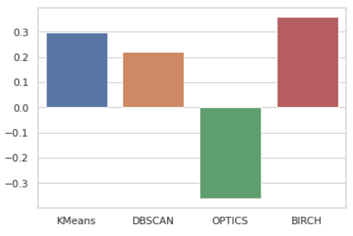
6.4 BIRCH

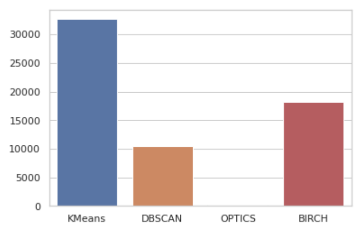
BIRCH is an unsupervised clustering algorithm which performs hierarchical clustering over large datasets. It creates a tree structure in-memory which is balanced as new items are scanned from the dataset. Branching Factor specifies the number of items in each of the leaf nodes. If the number of items exceeds a Threshold value , the tree is re-balanced by grouping smaller sub-clusters into larger ones.
Applying BIRCH to our dataset:

```
brc = Birch(n_clusters=4)
brc.fit(rfm_scaled_df)

output = brc.predict(rfm_scaled_df)

#Output
Silhouette Coefficient: 0.360
CHIndex :  18186.059451465997
DB Score : 1.3969639210951768
```


(a) Silhouette Score


(b) Calinski-Harabasz Index

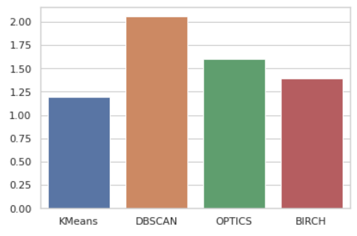

(c) Davis-Bouldin Score

Figure 12: Performance measures compared

6.5 K-means with 6 features

We finally apply K-Means Algorithm again on this new model with added features:

```
    clusterer = KMeans(n_clusters=4, random_state=10)
    cluster_labels = clusterer.fit_predict(rfm_scaled_df)
    # 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 = metrics.silhouette_score(rfm_scaled_df, cluster_labels)
    print("For n_clusters =", n_clusters,
         "The average silhouette_score is :", silhouette_avg)
    # Compute the silhouette scores for each sample
    sample_silhouette_values = silhouette_samples(rfm_scaled_df, cluster_labels)

#Output for number of clusters = 4
The average silhouette_score is : 0.2986975535055303
CHIndex :  32672.080148616238
DB Score : 1.1933283019655088
```

7 Performance Comparison

Clustering performance evaluation is a non trivial exercise. It cannot be measured in terms of an absolute number like supervised algorithm measures e.g. Precision or Recall. Rather, it measures how similar are elements from the same cluster and how dissimilar are elements among different clusters. We used 3 performance measures to compare the algorithms seen so far: Silhouette Score, Calinski-Harabasz Index and Davis-Bouldin Score.

8 Summary

- Density based algorithms identify too many clusters which is not suitable for effective customer segmentation strategy
- Performance measures for OPTICS imply there is no significant difference between members of different clusters(Negative Silhouette Coefficient and low CHI)
- BIRCH performs better than DBSCAN and OPTICS albeit with an explicit input for number of clusters
- K-Means offers the most balanced performance with a positive Silhouette Score and high CHI coupled with lowest DBS

- DBSCAN and OPTICS are best suited for applications where we need to identify arbitrary clusters of points for initial exploratory analysis
- BIRCH is best suited for creating a structured summary of the dataset

Therefore, KMeans is the most appropriate algorithm which can be applied to the problem of customer segmentation as it is solely based on the distance metric and includes every possible customer from the dataset. In contrast, a large number of noise points identified by DBSCAN and OPTICS are dropped and not added to any cluster.

9 Classification using Multilayer Perceptron Model - A NN Approach

Unsupervised algorithms discussed so far do not require labelled data. They can identify clusters from the given dataset based on a certain metric e.g. distance or density. We will take this study a step further by checking how can an Artificial Neural Network based Classifier be used for the problem of customer segmentation.
We will use the output of K-Means Algorithm from the previous section as the ground truth to build a basic MultiLayer Perceptron Model for classifying customers. We will use Keras which is a high level API for creating MLP models and capable of handling large datasets like ours.

9.1 Label Encoding

Since we are dealing with categorical variables(Labels), we need to covert them into numeric variable using One-Hot Encoding as below.

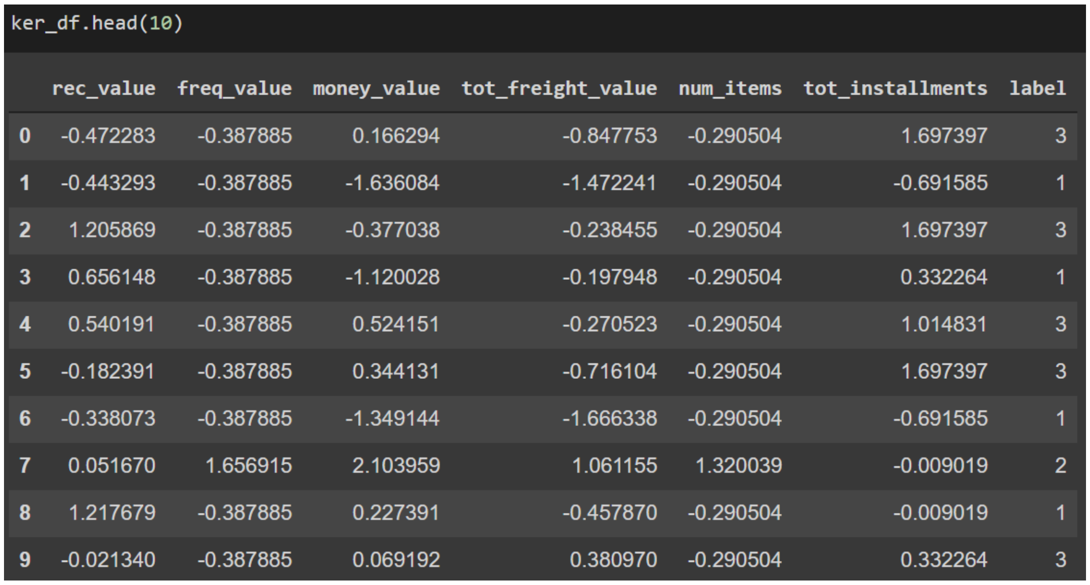
```
labels = ker_df['label']
features = ker_df.iloc[:,0:6]

encoder = LabelEncoder()
encoder.fit(labels)
encoded_Y = encoder.transform(labels)
# convert integers to dummy variables (i.e. one hot encoded)
dummy_y = np_utils.to_categorical(encoded_Y)
```

9.2 Hyperparameter Tuning

MLP consists of more than one layer of neurons which are activated using an activation function. The combined output of these neurons over multiple epochs determines the output class. In our case, the segment which a customer belongs to.
The number of layers, epochs, batch size, validation ratio and learning rate are called Hyperparameters. We use a 2 layer model with the first layer containing 8 and second layer containing 4 neurons each. We use 33 percent of data as test set for validation purposes. Ideally, we should use Grid Search algorithm to ascertain the appropriate values for the Hyperparameters. The last column of the dataframe contains the label assigned to it by the K-Means Algorithm.
The model's performance can be improved further by using different ANN structures with a multitude of activation functions.

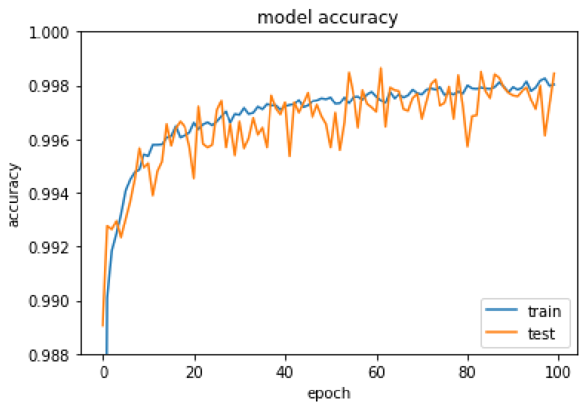
```
# create model
model = Sequential()
model.add(Dense(8, input_dim=6, activation='relu'))
model.add(Dense(4, activation='sigmoid'))
```



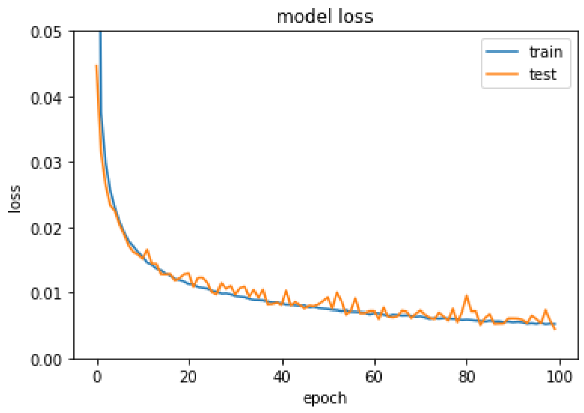
```
ker_df.head(10)
```

	rec_value	freq_value	money_value	tot_freight_value	num_items	tot_installments	label
0	-0.472283	-0.387885	0.166294	-0.847753	-0.290504	1.697397	3
1	-0.443293	-0.387885	-1.636084	-1.472241	-0.290504	-0.691585	1
2	1.205869	-0.387885	-0.377038	-0.238455	-0.290504	1.697397	3
3	0.656148	-0.387885	-1.120028	-0.197948	-0.290504	0.332264	1
4	0.540191	-0.387885	0.524151	-0.270523	-0.290504	1.014831	3
5	-0.182391	-0.387885	0.344131	-0.716104	-0.290504	1.697397	3
6	-0.338073	-0.387885	-1.349144	-1.666338	-0.290504	-0.691585	1
7	0.051670	1.656915	2.103959	1.061155	1.320039	-0.009019	2
8	1.217679	-0.387885	0.227391	-0.457870	-0.290504	-0.009019	1
9	-0.021340	-0.387885	0.069192	0.380970	-0.290504	0.332264	3

Figure 13: The last column contains the labels assigned by Kmeans



(a) Accuracy over epochs



(b) Loss over epochs

Figure 14: Accuracy vs Loss

```
# Compile model
model.compile(loss='categorical_crossentropy', optimizer='adam',
↪ metrics=['accuracy'])
# Fit the model
history = model.fit(features, dummy_y, validation_split=0.33, epochs=100,
↪ batch_size=4, verbose=0)
```

The model uses categorical cross entropy as the loss function with the Adam optimizer. To conclude, we summarize the model accuracy and loss over 100 epochs:

```
# summarize history for accuracy
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
# summarize history for loss
```

```
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
```

10 Conclusion and Future Work

Businesses who are able to personify their customers the fastest have an advantage in the established online e-commerce space. Customer Segmentation allows businesses to target novel marketing campaigns and advertisements to the right set of customers at the right time which results in increased revenue and customer retention.
Democratization of Machine learning has made accessible to the masses, a plethora of tools which can be used to analyse and derive meaningful hidden insights from the wealth of data collected for every customer action.
We started off with the RFM Model. We discussed DBSCAN, OPTICS, BIRCH and KMeans algorithms to carry out customer segmentation on our dataset. Later, we used the output of KMeans as the ground truth and created an ANN based classifier on it. Although, we have chosen KMeans to be the algorithm of choice for this exercise, in the real world a lot of external factors need to be taken into account such as the customer's age and income before dividing them into clusters. KMeans is also not resistant to outliers which could cause them to be misinterpreted as part of a different cluster.
Lastly, we discarded several features such as customer city , product categories and zip code to simplify our model. Those can also be incorporated along with more descriptive features to get a holistic customer segmentation model.

10.1 Future Work

As we have seen, customer segmentation using unsupervised clustering is a non trivial problem which can be solved using a variety of tools and techniques. Even more so, analysing the clustering performance is difficult owing to the nature of the problem. Some of the other advanced algorithms which we can apply to get more sophisticated segmentation include:

- Meanshift Algorithm, Gaussian Mixture Model with Expectation Maximization
- Discriminative clustering for market segmentation
- Spectral clustering with dimensionality reduction
- Incorporating sentiment analysis into product review comments and take pre-emptive actions to improve the quality

To conclude, the complexity of the algorithm used for segmentation would depend on the resources available for the business and the experience level of the Data Scientist/Analyst tasked with the job.

11 Literature References

There has been considerable research in the field of data mining to carry out customer segmentation. The following are some of the resources referred for preliminary literature review:

References

[1] Boyu Shen, *E-commerce Customer Segmentation via Unsupervised Machine Learning* CONFCDS 2021, January 28–30, 2021, Stanford, CA, USA

[2] Sukru Ozan and Leonardo O. Iheme, *Artificial Neural Networks in Customer Segmentation* Adres-Gezgini Inc. Research and Development Center,Izmir, Turkey

[3] Qiasi, Razieh, et al. *Developing a model for measuring customer's loyalty and value with RFM technique and clustering algorithms.* The Journal of Mathematics and Computer Science 4.2 (2012): 172-181.

[4] Jing Wu, Zheng Lin, *Research on Customer Segmentation Model by Clustering,* School of Information, Central University of Finance and Economics

[5] Michel Wedel ,Wagner A. Kamakura *Market Segmentation: Conceptual and Methodological Foundations,* SOM Marketing

[6] Daqing Chen ,Sai Laing Sain ,Kun Guo *Data mining for the online retail industry: A case study of RFM model-based customer segmentation using data mining* 2012 The Journal of Database Marketing & Customer Strategy Management Volume: 19, Issue: 3, pp 197-208

[7] Jon Kleinberg 1,Christos Papadimitriou 2,Prabhakar Raghavan 3 *Segmentation problems,* 2004 Journal of the ACM Volume: 51, Issue: 2, pp 263-280