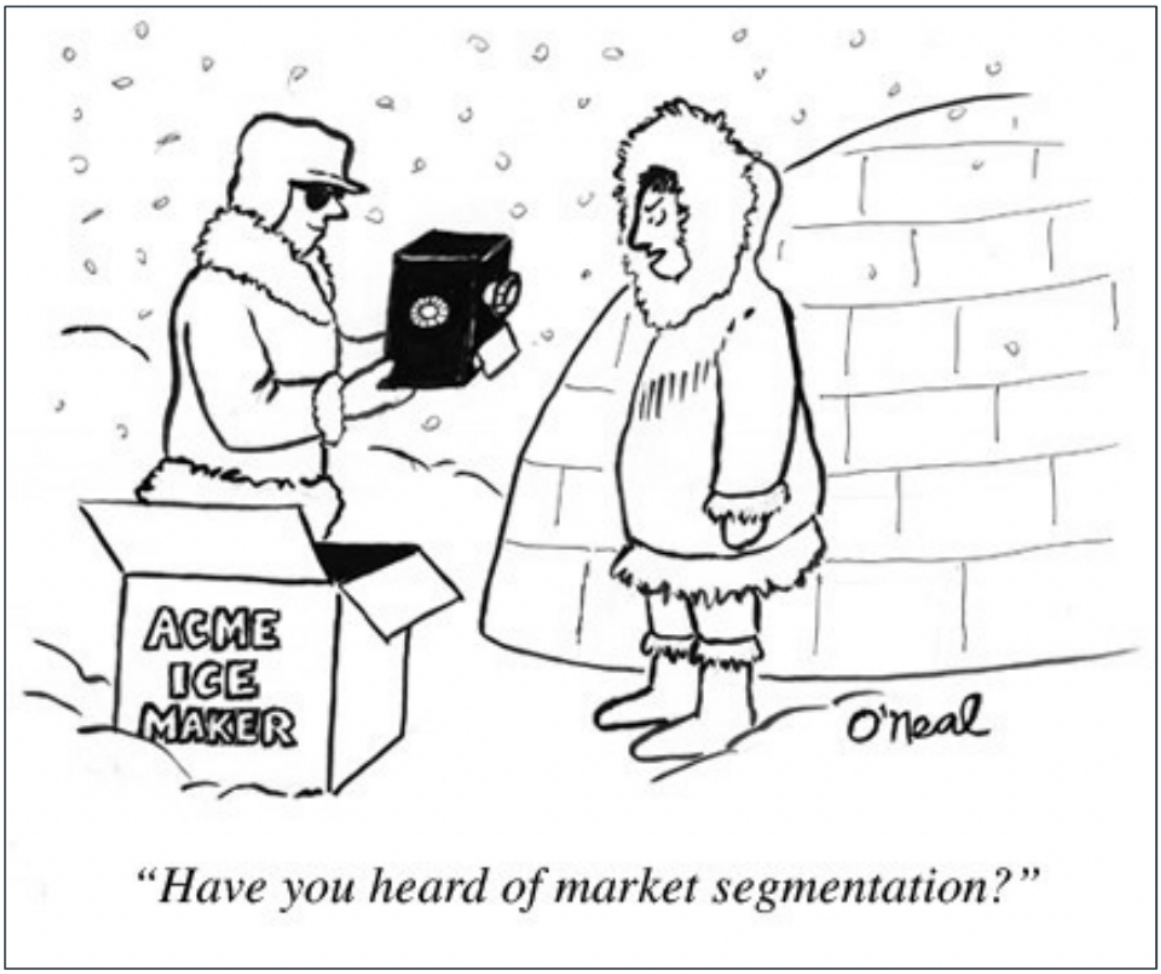
Table of Contents

Customer Segmentation



Problem Statement

We own an ecommerce site, we have some basic data about our customers like Customer ID, age, gender, annual income and spending score. We want a deeper understanding of the customers like who are the target customers so that the marketing team can plan the strategy accordingly.

Customer Segmentation is the strategy that can be used to find discrete customer groups that share similar characteristics. The idea behind customer segmentation is to split the user-base into smaller groups that can be targeted with specialized content and offers. The customer groups are drawn from user behaviour data which gives the business a deeper understanding of the types of users that exist in the system.

Some of the benefits of customer segmentation are:

1. Determine appropriate product pricing.
2. Develop customized marketing campaigns.
3. Design an optimal distribution strategy.
4. Choose specific product features for deployment.
5. Prioritize new product development efforts.

The purpose of the project is to understand:

- How can the data be preprocessed in order to correctly segment users into groups of similar people?
- What is a reasonable cluster size?
- What insights can we get from these cluster segments?
- How can the created clusters be used in the application to target users with specialized content e.g. item recommendations?

Data Acquisition and Data Wrangling

Data set used

I used a public dataset from the UCI Machine Learning Repository on transactions occurring between 01/12/2010 and 09/12/2011 for a UK-based and registered non-store online retail. The dataset can be found here. The company mainly sells unique all-occasion gifts. Many customers of the company are wholesalers.

Data features:

- *InvoiceNo*: Invoice number. Nominal, a 6-digit integral number uniquely assigned to each transaction. If this code starts with letter 'c', it indicates a cancellation
- *StockCode*: Product (item) code. Nominal, a 5-digit integral number uniquely assigned to each distinct product.
- *Description*: Product (item) name. Nominal
- *Quantity*: The quantities of each product (item) per transaction. Numeric
- InvoiceDate: Invoice Date and time. Numeric, the day and time when each transaction was generated
- *UnitPrice*: Unit price. Numeric, Product price per unit in sterling
- *CustomerID*: Customer number. Nominal, a 5-digit integral number uniquely assigned to each customer
- *Country*: Country name. Nominal, the name of the country where each customer resides.

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	2010-12-01 08:26:00	2.55	17850.0	United Kingdom
1	536365	71053	WHITE METAL LANTERN	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom
2	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	2010-12-01 08:26:00	2.75	17850.0	United Kingdom
3	536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom
4	536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom
5	536365	22752	SET 7 BABUSHKA NESTING BOXES	2	2010-12-01 08:26:00	7.65	17850.0	United Kingdom
6	536365	21730	GLASS STAR FROSTED T-LIGHT HOLDER	6	2010-12-01 08:26:00	4.25	17850.0	United Kingdom

Upon investigation, I found that the Quantity field has negative values. Similarly, the unit price field had negative values. It looked like the transactions included returns as well. As our goal is to perform customer segmentation and collaborative filtering, I removed these records.

Further investigation revealed

- There were no records where both quantity and price were negative.

- There were 1336 records where one of them was and the other was 0, for all these records there was no customer ID. The conclusion was to delete all records in that quantity or the price is negative.
- There were also 135,080 records without customer identification that I decided to disregard.

```
details = show_charecteristics(retail_df).sort_values(by='distincts', ascending=False)
```

```
Data shape: (397884, 8)
_____________________________
Data types:
 object            4
float64            2
int64              1
datetime64[ns]     1
Name: types, dtype: int64
_____________________________
```

	types	counts	distincts	nulls	uniques	missing_percent
InvoiceNo	object	397884	18532	0	[536365, 536366, 536367, 536368, 536369, 53637...	0.0
InvoiceDate	datetime64[ns]	397884	17282	0	[2010-12-01T08:26:00.000000000, 2010-12-01T08:...	0.0
CustomerID	float64	397884	4338	0	[17850.0, 13047.0, 12583.0, 13748.0, 15100.0, ...	0.0
Description	object	397884	3877	0	[WHITE HANGING HEART T-LIGHT HOLDER, WHITE MET...	0.0
StockCode	object	397884	3665	0	[85123A, 71053, 84406B, 84029G, 84029E, 22752,...	0.0
UnitPrice	float64	397884	440	0	[2.55, 3.39, 2.75, 7.65, 4.25, 1.85, 1.69, 2.1...	0.0
Quantity	int64	397884	301	0	[6, 8, 2, 32, 3, 4, 24, 12, 48, 18, 20, 36, 80...	0.0
Country	object	397884	37	0	[United Kingdom, France, Australia, Netherland...	0.0

We can see that the stockcode and description do not match up which can be due to data entry issues.

We can see some of the items have multiple descriptions. A simple spelling mistake or an additional space can end up in reducing data quality. I fixed it by using the most frequently used description.

```
unique_desc  = retail_df.groupby('StockCode')['Description'].apply(lambda x: x.mode().iloc[0]).reset_index()
retail_df['Description'] = retail_df['StockCode'].map(unique_desc.set_index('StockCode')['Description'])
```

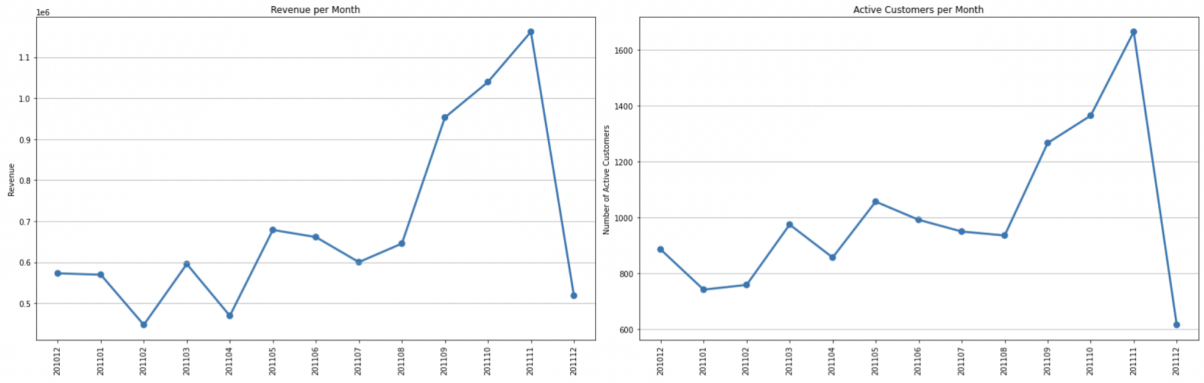
I then add a new column amount obtained by multiplying unit price with quantity.

The cleansed data looked as follows.

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country	Amount
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	2010-12-01 08:26:00	2.55	17850	United Kingdom	15.30
1	536365	71053	WHITE METAL LANTERN	6	2010-12-01 08:26:00	3.39	17850	United Kingdom	20.34
2	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	2010-12-01 08:26:00	2.75	17850	United Kingdom	22.00
3	536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	2010-12-01 08:26:00	3.39	17850	United Kingdom	20.34
4	536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	6	2010-12-01 08:26:00	3.39	17850	United Kingdom	20.34
...	...	...	...	...	...	...	...	...	...

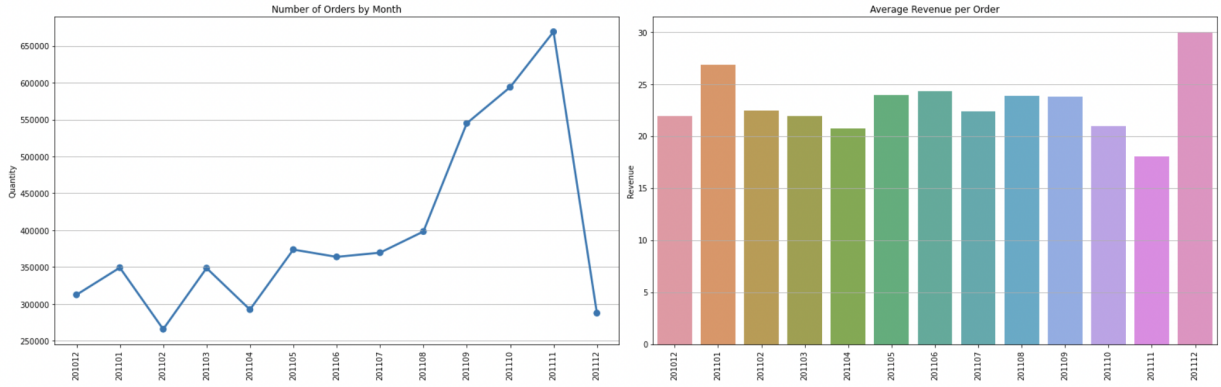
Exploratory Data Analysis

Now I visually analyzed to see if there is any trend in the data. I started by analyzing the total revenue and total active customers by month.



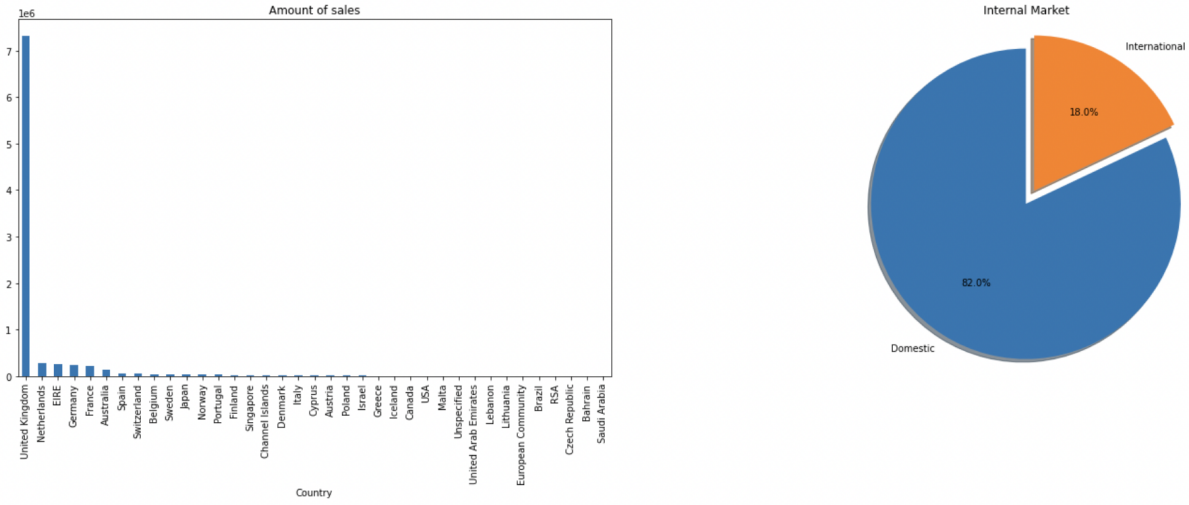
We can see that the revenue is growing especially Aug '11 onwards. The data for December does not appear to be incomplete. We can see that in April, Monthly Active Customer numbers dropped.

Next I looked into trends in monthly order Count & Average Revenue per Order.



As expected, order count also declined in April. Also, Average revenue per order dropped in April.

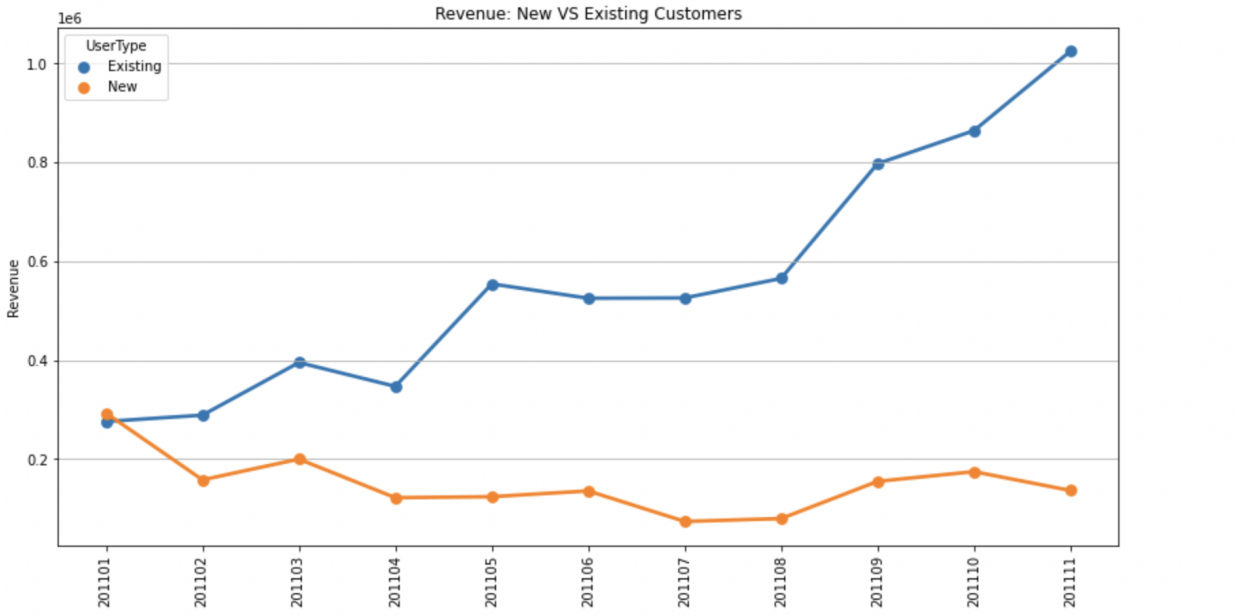
Next, we look at which country accounts for Maximum sales.



It is clear that the majority of our sales are to domestic customers in the United Kingdom.

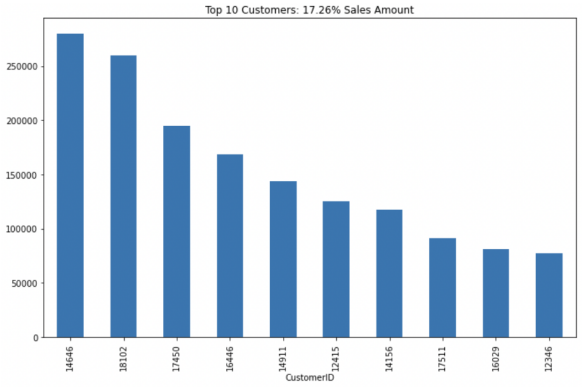
What is the Revenue per Month for New and Existing Customers?

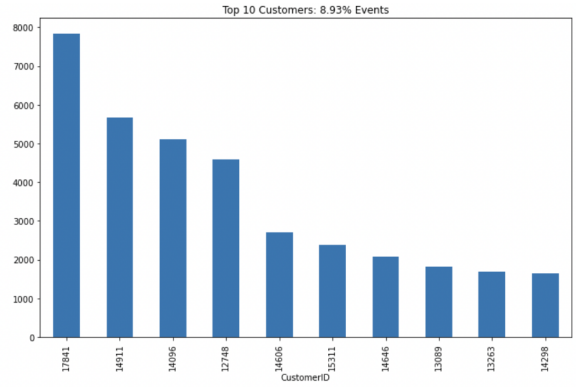
A new customer will be determined by their first purchase in our defined monthly time period.



Existing customers are showing a positive trend. Our customer base is growing but new customers have a slight negative trend.

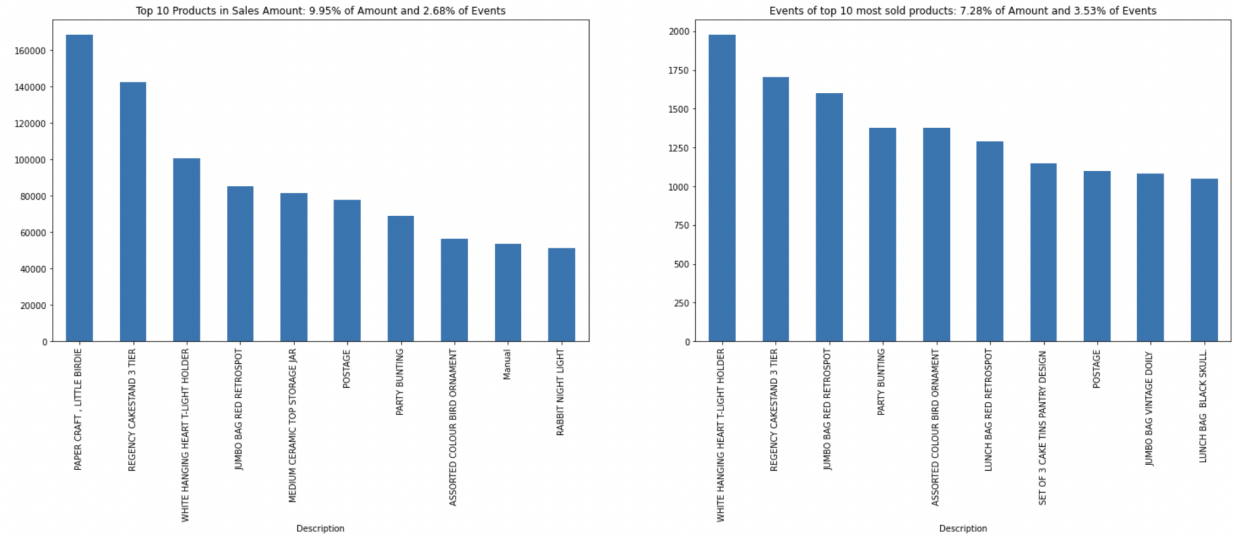
What is the contribution of our top 10 customers to revenue and total orders?





It is seen that the top 10 customers contribute approximately to 18% of revenue and 9.0% of orders .

What are our top 10 products and what is their contribution to revenue and total orders?



Our top 10 product by revenue contributes to 9.95% total revenue and 2.68% of the orders. Top 10 most sold products contribute to 7.28% of revenue and 3.53% of orders.

Data Modeling

Cluster Analysis

In the context of customer segmentation, cluster analysis is the use of a mathematical model to discover groups of similar customers based on finding the smallest variations among customers within each group. These homogeneous groups are known as “customer archetypes” or “personas”.

A common cluster analysis method is a mathematical algorithm known as *k-means cluster analysis*, sometimes referred to as scientific segmentation. The clusters that result assist in better customer modeling and predictive analytics, and are also used to target customers with offers and incentives personalized to their wants, needs and preferences.

The process is not based on any predetermined thresholds or rules. Rather, the data itself reveals the customer prototypes that inherently exist within the population of customers.

Finally, we can find traits of customer segments by analyzing the characteristics of the clusters.

There are many ways to model customer segmentation. Since our dataset is limited to the sales records, and does not include any other information about our customers, I used a **RFM**,***Recency, Frequency and Monetary Value**, based model of customer value for finding our customer segments. The RFM model will take the transactions of a customer and calculate three important informational attributes about each customer:

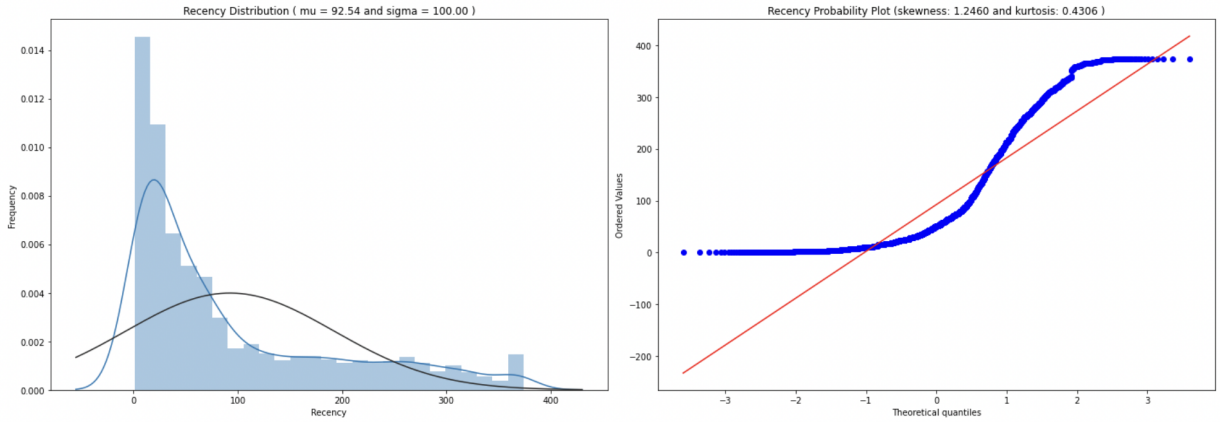
- **Recency**: The value of how recently a customer purchased at the establishment
- **Frequency**: How frequent the customer's transactions are at the establishment
- **Monetary value**: The dollar (or pounds in our case) value of all the transactions that the customer made at the establishment

Recency

To create the recency feature variable, I had to decide on the reference date for our analysis. I used the last transaction date plus one day as our reference date.. Then, I constructed the recency variable as the number of days before the reference date when a customer last made a purchase.

```
ref_date = retail_df.InvoiceDate.max() + timedelta(days=1)
```

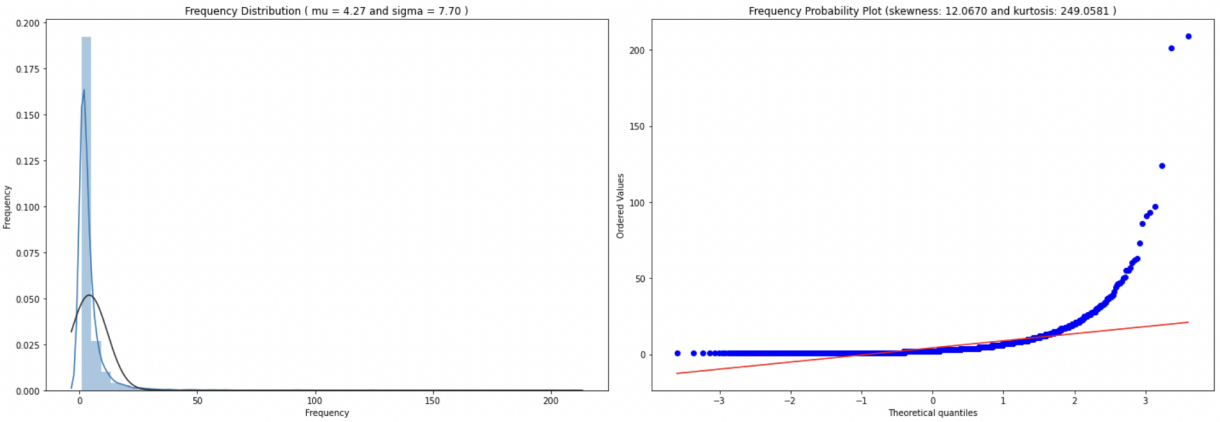
The diagram below shows the histogram output of the distribution of recency across our customers.



The recency distribution graph above shows that the sales recency distribution is skewed, has a peak on the left and a long tail to the right. It deviates from normal distribution and is positively biased.

Probability Plot, shows that the sales recency also does not align with the diagonal red line that represents the normal distribution. The form of its distribution confirms that it is right skewed.

Frequency

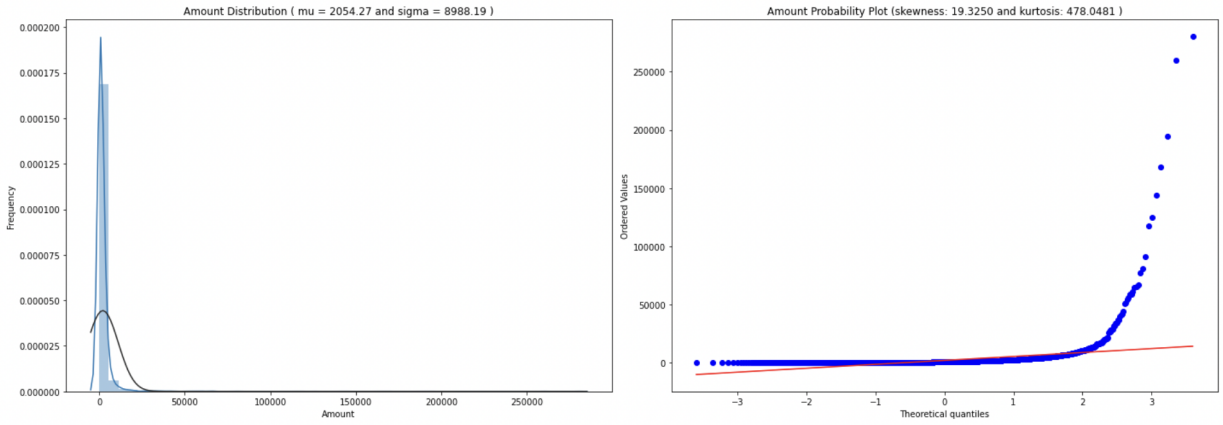


From the graph it is clear that the sales frequency distribution is also skewed,has a peak on the left and a long tail to the right. It deviates from normal distribution and is positively biased.

Probability Plot, shows that the sales frequency also does not align with the diagonal red line that represents the normal distribution. The form of its distribution confirms that it is right skewed.

Revenue

Now I looked into how our customer database looks when we cluster them based on revenue. I calculated the revenue for each customer, then plotted a histogram and applied the same clustering method.



The revenue distribution graph above shows that the distribution is also skewed, has a peak on the left and a long tail to the right. It deviates from normal distribution and is positively biased.

Probability Plot, shows that the sales revenue also does not align with the diagonal red line that represents the normal distribution. The form of its distribution confirms that it is right skewed.

To get a snapshot about what recency,frequency and revenue looks like, I used pandas' **.describe()** method. It shows mean, min, max, count and percentiles of our data.

	CustomerID	Recency	Frequency	Amount
count	4338.000000	4338.000000	4338.000000	4338.000000
mean	15300.408022	92.536422	4.272015	2054.266460
std	1721.808492	100.014169	7.697998	8989.230441
min	12346.000000	1.000000	1.000000	3.750000
25%	13813.250000	18.000000	1.000000	307.415000
50%	15299.500000	51.000000	2.000000	674.485000
75%	16778.750000	142.000000	5.000000	1661.740000
max	18287.000000	374.000000	209.000000	280206.020000

Now we have our customer value dataset. For the customer segmentation,I used the K-means clustering algorithm as mentioned earlier. One of the requirements for proper functioning of the algorithm is the centering of the variable with different means.

Mean centering refers to replacing the actual value of the variable with a standardized value, so that the variable has a mean of 0 and variance of 1. This ensures that all the variables are in the same range and the difference in ranges of values does not degrade the performance of the algorithm. This is similar to feature scaling.

Another problem is the huge range of values each variable takes as seen from the monetary amount variable. To address this problem, I transformed all the variables on the log scale. This transformation, along with the standardization, ensured that the input to the algorithm is a homogenous set of scaled and transformed values.

Sequence of steps

1. Unskew the data - log transformation
2. Standardize to the same average values and scale to the same standard deviation

An important point about the data preprocessing step is that sometimes we need it to be reversible. In this case, the clustering results is in terms of the log transformed and scaled variable. But to make inferences in terms of the original data, I had to reverse

transform all the variables, so that we get back the actual RFM figures. This was done by using the preprocessing capabilities of Python.

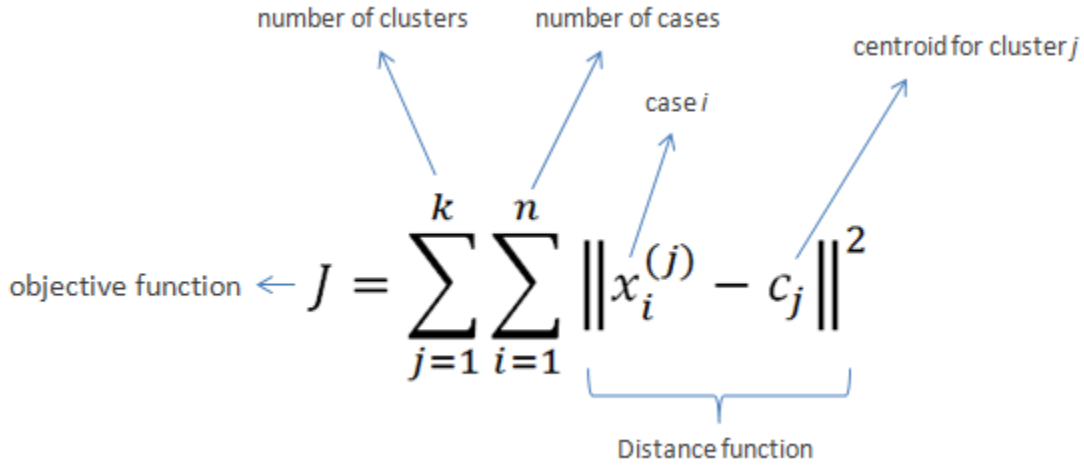
	count	mean	std	min	25%	50%	75%	max
Recency_log	4338.0	-1.027980e-16	1.000115	-2.630445	-0.612424	0.114707	0.829652	1.505796
Frequency_log	4338.0	-2.355833e-16	1.000115	-1.048610	-1.048610	-0.279044	0.738267	4.882714
Amount_log	4338.0	-1.013738e-16	1.000115	-4.179280	-0.684183	-0.060942	0.654244	4.721395

Now it is seen that the values are centred around 0 and have a standard deviation of 1. This data can now be for clustering analysis.

K-Means Clustering

The K-means clustering belongs to the partition based\centroid based hard clustering family of algorithms, a family of algorithms where each sample in a dataset is assigned to exactly one cluster.

Based on this Euclidean distance metric, we can describe the k-means algorithm as a simple optimization problem, an iterative approach for minimizing the within-cluster sum of squared errors (SSE), which is sometimes also called cluster inertia. So, the objective of K-Means clustering is to minimize total intra-cluster variance, or, the squared error function:



The steps that happen in the K-means algorithm for partitioning the data are as given follows:

1. The algorithm starts with random point initializations of the required number of centers. The “K” in K-means stands for the number of clusters.
2. In the next step, each of the data point is assigned to the center closest to it. The distance metric used in K-means clustering is normal Euclidian distance.
3. Once the data points are assigned, the centers are recalculated by averaging the dimensions of the points belonging to the cluster.
4. The process is repeated with new centers until we reach a point where the assignments become stable. In this case, the algorithm terminates.

K-means++

- Place the initial centroids far away from each other via the k-means++ algorithm, which leads to better and more consistent results than the classic k-means.

- To use k-means++ with scikit-learn's KMeans object, we just need to set the init parameter to k-means++ (the default setting) instead of random.

Clustering for Segments

One of the most perplexing issues we face while trying to segment customers is choosing the ideal number of segments. This is a key parameter for multiple clustering algorithms like K means. For our analysis we will use the following techniques to identify the optimal number of clusters.

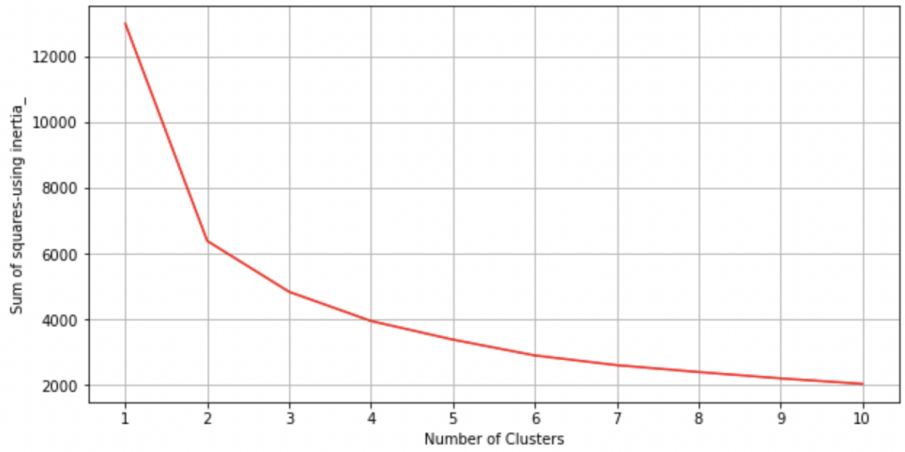
1. Elbow method
2. Gap Statistic
3. Silhouette Coefficient

Elbow Method

It is the most popular method for determining the optimal number of clusters. The method is based on calculating the Within-Cluster-Sum of Squared Errors (WSS) for

different numbers of clusters (k) and selecting the k for which change in WSS first starts to diminish.

The idea behind the elbow method is that the explained variation changes rapidly for a small number of clusters and then it slows down leading to an elbow formation in the curve. The elbow point is the number of clusters we can use for our clustering algorithm.



Elbow Method by calculating sum-of-squares error in each cluster against *K* confirms that the best will be k=4 (plot starts descending much more slowly after k=4)

Gap Statistic

The gap statistic was developed by Stanford researchers Tibshirani, Walther and Hastie in their 2001 paper. The idea behind their approach was to find a way to compare cluster compactness with a null reference distribution of the data, i.e. a distribution with no obvious clustering. Their estimate for the optimal number of clusters is the value for which cluster compactness on the original data falls the farthest below this reference curve. This information is contained in the following formula for the gap statistic:

$$\mathrm{Gap}_n(k) = E_n^*\{\log W_k\} - \log W_k$$

where Wk is measure of the compactness of our clustering based on the Within-Cluster-Sum of Squared Errors (WSS):

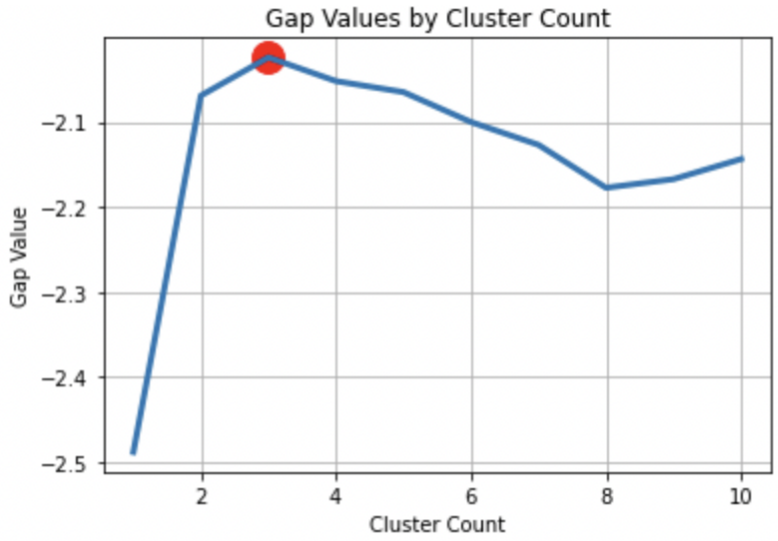
$$D_k = \sum_{\mathrm{x}_i \in C_k} \sum_{\mathrm{x}_j \in C_k} ||\mathrm{x}_i - \mathrm{x}_j||^2 = 2n_k \sum_{\mathrm{x}_i \in C_k} ||\mathrm{x}_i - \mu_k||^2$$

Within-Cluster-Sum of Squared Errors is calculated by the `inertia_` attribute of `KMeans` function as follows:

- The square of the distance of each point from the centre of the cluster (Squared Errors)
- The WSS score is the sum of these Squared Errors for all the points

Calculating gap statistic in python for k means clustering involves the following steps:

- Cluster the observed data on various number of clusters and compute compactness of our clustering
- Generate reference data sets and cluster each of them with a varying number of clusters. The reference datasets are created from a “continuous uniform” distribution using the `random_sample` function.
- Calculate average of compactness of our clustering on reference datasets
- Calculate gap statistics as difference in compactness between clustering on reference data and original data



As seen in Figure , the gap statistics is maximized with 3 clusters and hence, we can choose 3 clusters for our K means.

Silhouette analysis on K-Means clustering

Silhouette analysis can be used to study the separation distance between the resulting clusters, as a strategy to quantify the quality of clustering via a graphical tool to plot a measure of how tightly grouped the samples in the clusters are. The silhouette plot displays a measure of how close each point in one cluster is to points in the neighboring clusters and thus provides a way to assess parameters like number of clusters visually.

Silhouette coefficients has a range of [-1, 1], it calculated by:

Calculate the cluster cohesion a(i)as the average distance between a sample x(i) and all other points in the same cluster. Calculate the cluster separation b(i) from the next closest cluster as the average distance between the sample x(i) and all samples in the nearest cluster. Calculate the silhouette s(i) as the difference between cluster cohesion and separation divided by the greater of the two, as shown here:

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}}$$

Which can be also written as

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}}$$

Where:

- If near +1, it indicates that the sample is far away from the neighboring clusters.
- A high value indicates that the object is well matched to its own cluster and poorly matched to neighboring clusters.
- If most objects have a high value, then the clustering configuration is appropriate.
- If many points have a low or negative value, then the clustering configuration may have too many or too few clusters.
- A value of 0 indicates that the sample is on or very close to the decision boundary between two neighboring clusters
- Negative values indicate that those samples might have been assigned to the wrong cluster.

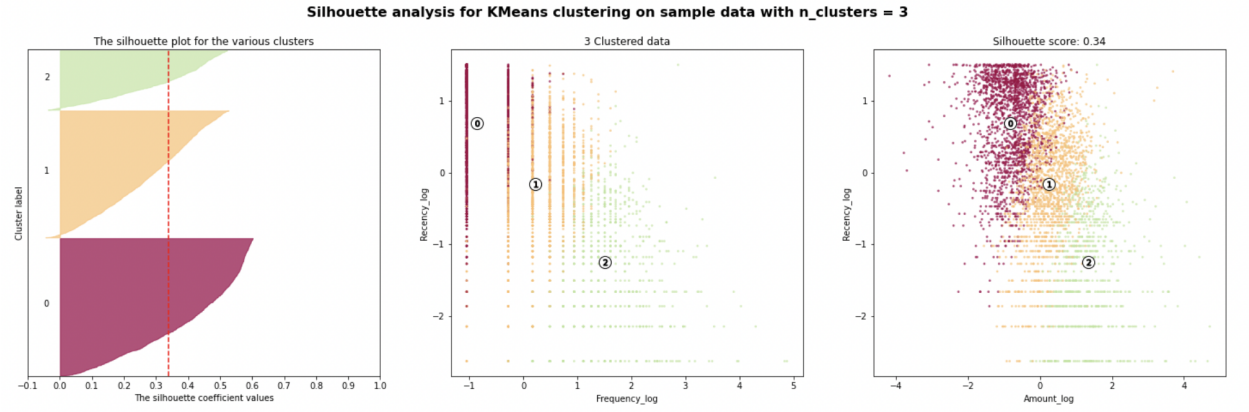
The silhouette plot can show a bad K clusters pick for the given data due to the presence of clusters with below average silhouette scores and also due to wide fluctuations in the size of the silhouette plots. A good k clusters can be found when all the plots are more or less of similar thickness and hence are of similar sizes.

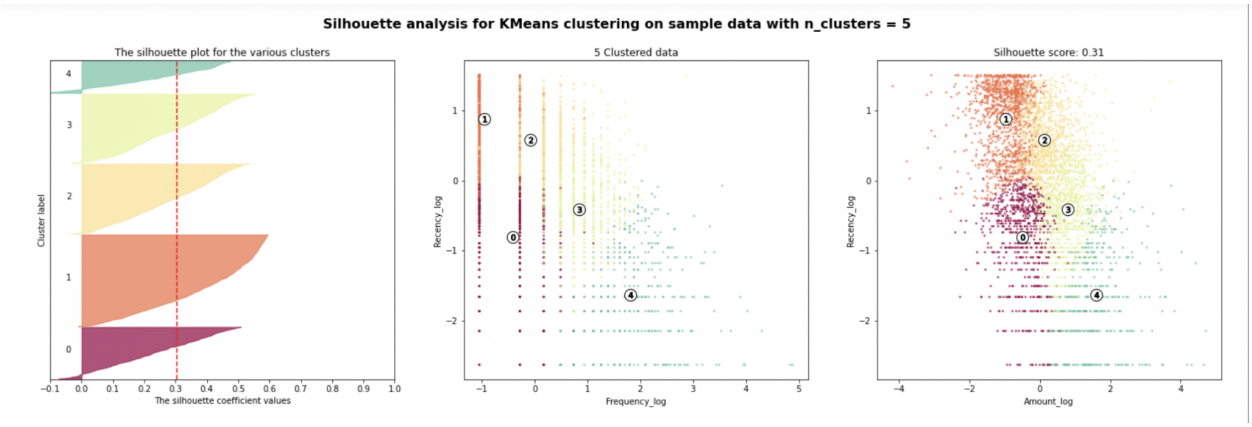
Although we have to keep in mind that in several cases and scenarios, sometimes we may have to drop the mathematical explanation given by the algorithm and look at the business relevance of the results obtained.

Let's see below how our data perform for each K clusters groups (between 3 and 5) in the silhouette score of each cluster, along with the center of each of the clusters discovered in the scatter plots, by amount_log vs recency_log and vs frequency_log.

When we look at the results of the clustering process, we can infer some interesting insights:

All the K clusters options are valid, because there is no presence of clusters with below average silhouette scores. All options had wide fluctuations in the size of the silhouette plots. So, our best choice lies on the option that gives us a simpler business explanation and at the same time target customers in focus groups with sizes closer to the desired.





Profile and interpret segments

Approaches to build customer personas

- Summary statistics for each cluster centers e.g. RFM values
- Relative importance of cluster attributes compared to population
- Snake plots to understand and compare segments

Summary statistics for each cluster centers:

Let's look at the cluster center values after returning them to normal values from the log and scaled version.

```
for 3 clusters the silhouette score is 0.34
Centers of each cluster:
      Recency  Frequency       Amount
0  115.938985   1.185917   260.171159
1   34.517136   3.144894  1005.824234
2    7.230229   9.964304  3924.968748
-------------------------------------------------
for 4 clusters the silhouette score is 0.33
Centers of each cluster:
      Recency  Frequency       Amount
0  153.398108   1.196931   267.202714
1   53.293280   3.772475  1365.818337
2   14.780012   1.750981   395.540264
3    6.502773  10.174447  4000.482026
-------------------------------------------------
for 5 clusters the silhouette score is 0.31
Centers of each cluster:
      Recency  Frequency       Amount
0   13.531089   1.759663   394.447109
1  152.731471   1.085433   213.482985
2   99.524516   2.372810   852.459464
3   24.050122   5.514949  1997.411280
4    4.206772  13.226861  5460.153603
-------------------------------------------------
```

Clusters Insights

Now that the plots and the center are in correct units, let's see some insights by each clusters groups:

Analysis:three-cluster

1. The three clusters appear to have a good difference in the Monetary value of the customer.

2. Cluster 2 is the cluster of high value customers who shop frequently and is the important segment for any business.
3. In the similar way we obtain customer groups with low and medium spends in clusters with labels 2 and 0 respectively.
4. Frequency and Recency correlate perfectly to the Monetary value based on the trend (High Monetary-Low Recency-High Frequency).

Analysis:four-cluster

1. Clusters 0 and 2 look very similar, but when we look deeper we can see that cluster 2 are more recent purchasers and have a slightly higher frequency of buying and monetary value.
2. Cluster 2 customers make low-cost purchases, with a relatively low frequency, but above cluster 0 customers, and made their last purchase more recently. This group of customers may probably respond to price discounts and can be subject to loyalty promotions to try and increase the medium-ticket strategy that can be better defined when we analyze the market basket.
3. Cluster 1 purchases medium amounts, with a relatively low frequency and not very recent Cluster 3 is the cluster of high value customers who shop frequently and is certainly an important segment for each business.
4. The silhouette score matrix says that the 4 cluster segments are less optimal then the three cluster segments.

Analysis:five-cluster

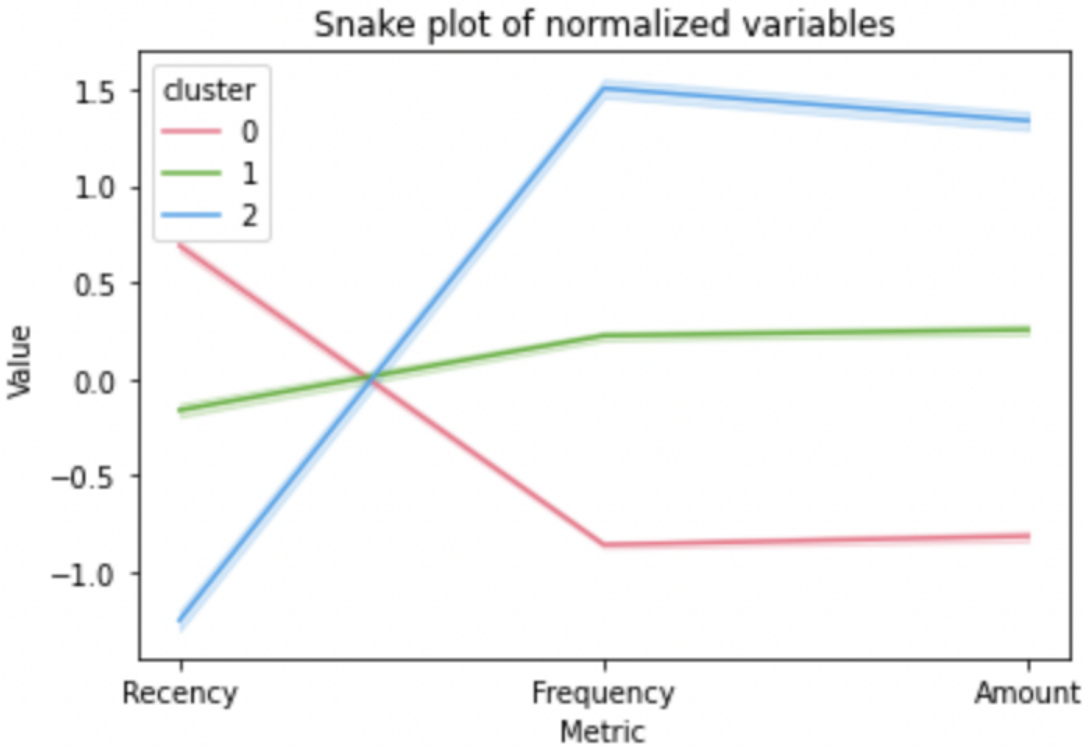
1. Note that clusters 0 and 1 are very similar.
2. The cluster 0 makes low-cost purchases, with a relatively low frequency, but above 0, and made their last purchase more recently
3. The cluster 4 appears more robust on the affirmation of those who shop often and with high amounts.
4. The cluster 2 are those who have a decent spend but are not that frequent.
5. The cluster 3 purchases medium amounts, with a relatively medium frequency and are more recent in purchases than cluster 2.
6. Cluster 0 and cluster 3 groups of customers may probably respond to price discounts and can be subject to loyalty promotions to try to increase the medium-ticket strategy that can be better defined when analyzing the market basket.
7. The silhouette score matrix says that the five cluster segments are less optimal then the three cluster segments.

k_best_silhouette = 3

A snake plot, or line chart, is a market research technique to compare different segments and visually present the attributes of each cluster. I created the plot by

putting Metrics at x-axis and values at the y-axis and grouping the values by K-clusters.

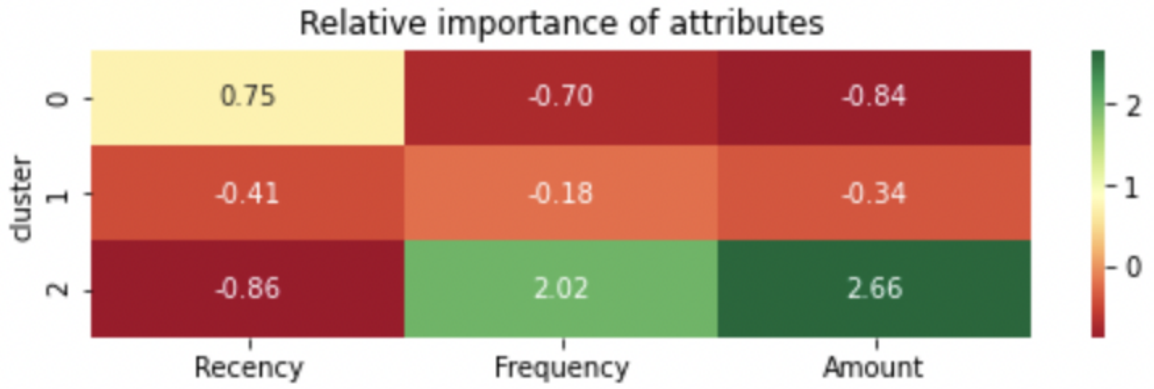
We can now compare the segments and identify insights.



calculate the relative importance of each cluster's attributes compared to the population average. Generally, we would like to have segments to be different from the overall population and have unique properties of their own. I did this by dividing the average value of each segment over the average value of population and

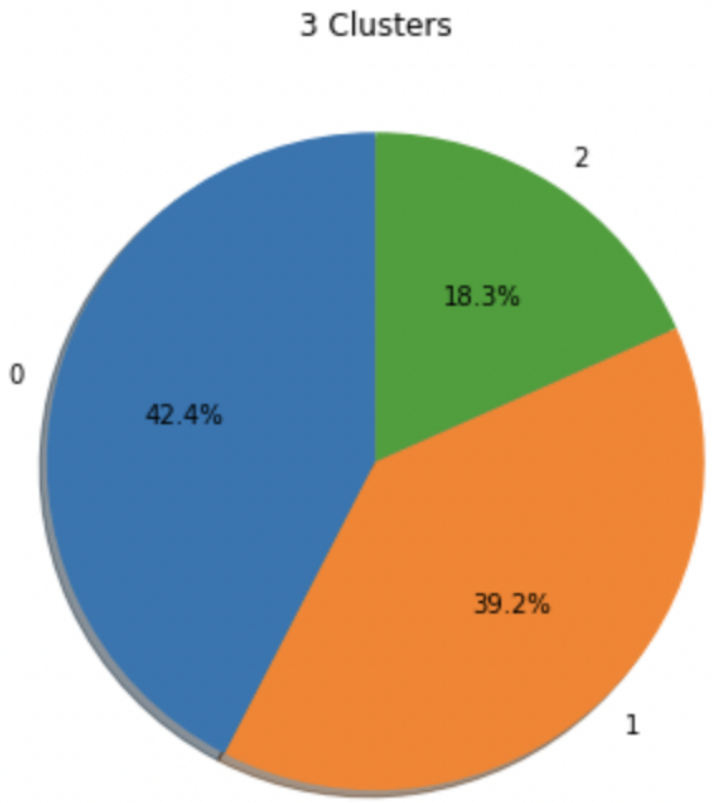
subtracting 1, ensuring 0 is returned when the segment mean equals the population mean.

The result is a relative importance score for each RFM value of the segments. The further that ratio is from zero, the more important that attribute is for defining a specific cluster compared to the population average. We can view it by just looking at rounded values, I plotted a heatmap for better visualization.



One can quickly notice that the **Top** group corresponds to cluster 2 in the heatmap and to what extent the exact numeric values differ in the two groups.

The figure below shows the distribution of our customer population to the three clusters.



Conclusions

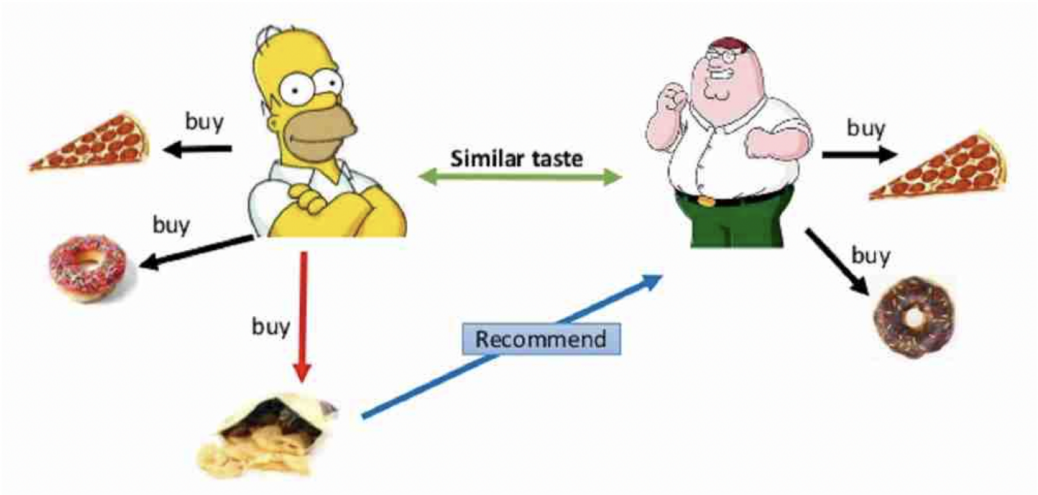
Discovery and the quality of clustering can be improved by adding other customer information and purchases details in this dataset.

For example:

- New indicators, such as customer relationship time, based on the date of first purchase of the client.
- Some group or category of product to be obtained through the SKUs

Another dimension to explore can be trying out different algorithms for performing the segmentation for instance hierarchical clustering.

Recommendation System



Recommendation systems are personalizing our web experience, telling us what to buy **(Amazon)**, which movies to watch **(Netflix)**, whom to be friends with **(Facebook)**, which songs to listen to **(Spotify)** etc. These recommendation systems leverage our shopping/ watching/ listening patterns and predict what we would like in future based on our behavior patterns.

Two most common types of recommender systems are **Content-Based** and **Collaborative Filtering (CF)**. Collaborative filtering produces recommendations based on the knowledge of users' attitude to items, that is it uses the "wisdom of the crowd" to recommend items. In contrast, content-based recommender systems focus on the attributes of the items and give you recommendations based on the similarity between them.

Problem Statement

My purpose here is to find out how we can use the collaborative filtering approach, i.e. the user is recommended items that people with similar tastes and preferences liked in the past. In other words, how we can predict unknown ratings by using the similarities between users.

Data Set Used

I used the same UCI Machine Learning Repository on online retail transactions for this effort to develop recommendation system algorithms, with the Surprise library.

This package has been specially developed to make recommendation based on collaborative filtering easy. It has default implementation for a variety of CF algorithms.

To load a data set from the above pandas data frame, we will use the load_from_df() method, we will also need a Reader object, and the rating_scale parameter must be specified. The data frame must have three columns, corresponding to the user ids, the item ids, and the ratings in this order. Each row thus corresponds to a given rating.

The transaction dataset we are using does not have any ratings and we will build one based on the number of times an item was ordered by a user.

Model Benchmarking

I benchmarked the following algorithms:

- KNNWithZScore - KNNWithZScore is a basic collaborative filtering algorithm, taking into account the z-score normalization of each user.
- KNNWithMeans - KNNWithMeans is basic collaborative filtering algorithm, taking into account the mean ratings of each user.
- KNNBaseline - KNNBaseline is a basic collaborative filtering algorithm taking into account a baseline rating.
- KNNBasic - KNNBasic is a basic collaborative filtering algorithm.

```
benchmark = []
# Iterate over all algorithms
for algorithm in [KNNBaseline(), KNNBasic(), KNNWithMeans(), KNNWithZScore()]:
    # Perform cross validation
    results = cross_validate(algorithm, data, measures=['RMSE'], cv=3, verbose=False)
```

Scoring Metric

I used “rmse” as our accuracy metric for the predictions.

Algorithm	test_rmse	fit_time	test_time
KNNWithZScore	0.050390	1.398475	9.845941
KNNWithMeans	0.051560	1.260634	9.047126
KNNBaseline	0.051933	1.495407	11.370895
KNNBasic	0.054231	1.385951	8.807874

The KNNWithZScore algorithm gave the best rmse.

Tuning the Algorithm Parameters

Surprise provides a GridSearchCV class analogous to GridSearchCV from scikit-learn.

With a dict of all parameters, GridSearchCV tries all the combinations of parameters and reports the best parameters for any accuracy measure

I checked which similarity metric works best with our chosen model.Then built a full Surprise training set from the dataset.

```
sim_options = {
    "name": ["msd", "cosine","pearson_baseline"],
    "min_support": [3, 4, 5],
    "user_based": [False],
}

param_grid = {"sim_options": sim_options}

gs = GridSearchCV(KNNWithZScore, param_grid, measures=["rmse"], cv=3)
gs.fit(data)
```

Making a Recommendation

Finally checked how the final recommendations looks for a specific customer 12680 and got the top 10 recommended items for the user.

When using Surprise there are RAW IDs and INNER IDs. RAW IDs are the IDs from our train dataset. The RAW ID is converted to an unique integer that Surprise can easily manipulate for computations. In order to find a user inside the trainset, I had to convert their RAW ID to INNER ID. For more details on this read: https://surprise.readthedocs.io/en/stable/FAQ.html#what-are-raw-and-inner-ids

```
Item : BATHROOM METAL SIGN
Item : PLAYING CARDS KEEP CALM & CARRY ON
Item : RETRO PLASTIC ELEPHANT TRAY
Item : PETIT TRAY CHIC
Item : PLAYING CARDS JUBILEE UNION JACK
Item : KIDS RAIN MAC PINK
Item : PARTY TIME PENCIL ERASERS
Item : CHILDS GARDEN TROWEL PINK
Item : PACK OF 6 SWEETIE GIFT BOXES
Item : MAGIC TREE -PAPER FLOWERS
Item : MAGIC SHEEP WOOL GROWING FROM PAPER
```

Conclusion

Surprise makes it easy to implement neighborhood and Similarity based recommendation algorithms. There are more sophisticated algorithms based on deep learning that can be used.

Apart from KNN based there are also other types of algorithms like matrix factorization based algorithms that are supported by surprise package for collaborative filtering.

Market Basket Analysis

Cross selling is the ability to sell more products to a customer by analyzing the customer's shopping trends as well as general shopping trends and patterns which are in common with the customer's shopping patterns.

I used association rule-mining, a powerful technique used for cross selling, then I applied the concept of market basket analysis to the retail transactions dataset.

Association analysis is relatively light on the math concepts and easy to explain to non-technical people. In addition, it is an unsupervised learning tool that looks for hidden patterns so there is limited need for data prep and feature engineering. It is a good start for certain cases of data exploration and can point the way for a deeper dive into the data using other approaches.

Association rules are normally written like: {Diapers} -> {Beer} which means that there is a strong relationship between customers that purchased diapers and also purchased beer in the same transaction.

In the above example, the {Diaper} is the antecedent and the {Beer} is the consequent. Both antecedents and consequent can have multiple items. In other words, {Diaper, Gum} -> {Beer, Chips} is **Support** is the relative frequency that the rules show up. We want to look for high support in order to make sure it is a useful relationship. However, there may be instances where a low support is useful if you are trying to find "hidden" relationships.

Confidence is a measure of the reliability of the rule. A confidence of .5 in the above example would mean that in 50% of the cases where Diaper and Gum were purchased, the purchase also included Beer and Chips. For product recommendation, a 50% confidence may be perfectly acceptable but in a medical situation, this level may not be high enough.

Lift is the ratio of the observed support to that expected if the two rules were independent. The basic rule of thumb is that a lift value close to 1 means the rules were completely independent. Lift values > 1 are generally more "interesting" and could be indicative of a useful rule pattern.

Association Rule-based algorithms are viewed as a two-step approach:

1. **Frequent Itemset Generation:** Find all frequent item-sets with support >= predetermined min_support count
2. **Rule Generation:** List all Association Rules from frequent item-sets. Calculate Support and Confidence for all rules. Prune rules that fail min_support and min_confidence thresholds.

Apriori algorithm

Frequent Itemset Generation is the most computationally expensive step because the algorithm scans the database too many times, which reduces the overall performance. Due to this, the algorithm assumes that the database is Permanent in the memory.

Also, both the time and space complexity of this algorithm are very high: O(2^{|D|}), thus exponential, where |D| is the horizontal width (the total number of items) present in the database.

Dataset

I used the online retail data for our analysis.

As mentioned earlier, a major bottleneck with any association rule-mining algorithm is the generation of frequent itemsets. If the transaction dataset is having k unique products, then potentially we have 2**k possible itemsets. We might run into performance with our dataset. I filtered the records for one country(Germany) for our analysis.

This analysis requires that all the data for a transaction be included in 1 row and the items should be 1-hot encoded. This creates a sparse table.

```
# seperating trasanctions for Germany
my_basket = myretail_df[myretail_df['Country'] == 'Germany'].groupby(['InvoiceNo','Description'])['Quantity']\
    .sum().unstack().reset_index().fillna(0).set_index('InvoiceNo')
```

Build the Model

```
#generating frequent item sets
my_frequent_itemsets = apriori(my_basket_sets,min_support=0.03,use_colnames=True)
```

Generate Rules

The generated rules are as given below:

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction
17	(CHARLOTTE BAG SUKI DESIGN)	(WOODLAND CHARLOTTE BAG)	0.045952	0.126915	0.037199	0.809524	6.378489	0.031367	4.583698
19	(CHILDRENS CUTLERY SPACEBOY)	(CHILDRENS CUTLERY DOLLY GIRL)	0.048140	0.050328	0.039387	0.818182	16.256917	0.036965	5.223195
21	(COFFEE MUG PEARS DESIGN)	(COFFEE MUG APPLES DESIGN)	0.039387	0.061269	0.035011	0.888889	14.507937	0.032598	8.448578
26	(JAM JAR WITH GREEN LID)	(JAM JAR WITH PINK LID)	0.035011	0.063457	0.032823	0.937500	14.773707	0.030601	14.984683
32	(JUMBO BAG PINK POLKADOT)	(JUMBO BAG RED RETROSPOT)	0.035011	0.078775	0.032823	0.937500	11.901042	0.030065	14.739606
98	(RED RETROSPOT CHARLOTTE BAG)	(WOODLAND CHARLOTTE BAG)	0.070022	0.126915	0.059081	0.843750	6.648168	0.050194	5.587746
134	(SET/6 RED SPOTTY PAPER CUPS)	(SET/6 RED SPOTTY PAPER PLATES)	0.052516	0.056893	0.045952	0.875000	15.379808	0.042964	7.544858
135	(SET/6 RED SPOTTY PAPER PLATES)	(SET/6 RED SPOTTY PAPER CUPS)	0.056893	0.052516	0.045952	0.807692	15.379808	0.042964	4.926915
138	(SPACEBOY CHILDRENS BOWL)	(SPACEBOY CHILDRENS CUP)	0.041575	0.043764	0.037199	0.894737	20.444737	0.035380	9.084245
139	(SPACEBOY CHILDRENS CUP)	(SPACEBOY CHILDRENS BOWL)	0.043764	0.041575	0.037199	0.850000	20.444737	0.035380	6.389497
178	(RED RETROSPOT CHARLOTTE BAG, ROUND SNACK BOXE...	(WOODLAND CHARLOTTE BAG)	0.030635	0.126915	0.030635	1.000000	7.879310	0.026747	inf

Finally, I looked at how much opportunity there is to use the popularity of one product to drive sales of another.

```
print(my_basket['JUMBO BAG PINK POLKADOT'].sum())
print(my_basket['JUMBO BAG RED RETROSPOT'].sum())
```

```
243.0
522.0
```

We can see that we sell 522 JUMBO BAG RED RETROSPOT but only 243 JUMBO BAG PINK POLKADOT,we can drive more JUMBO BAG PINK POLKADOT sales through recommendations.

Conclusions

Market basket analysis is a set of calculations meant to help businesses understand the underlying patterns in their sales. The sales of certain goods are complementary, they are often bought together and Apriori Algorithm can uncover them.

References:

Build a model based recommendation system:
https://towardsdatascience.com/how-to-build-a-model-based-recommendation-system-using-python-surprise-2df3b77ab3e5

Surprise' documentation:
https://surprise.readthedocs.io/en/stable/index.html

Item-based Collaborative Filtering : Build Your own Recommender System! - https://www.analyticsvidhya.com/blog/2021/05/item-based-collaborative-filtering-build-your-own-recommender-system/

Market basket analysis:
https://www.kdnuggets.com/2019/12/market-basket-analysis.html

K-Means clustering:
https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html

K Means Clustering Simplified in Python:
https://www.analyticsvidhya.com/blog/2021/04/k-means-clustering-simplified-in-python/

K-Means Clustering and the Gap-Statistics:
https://towardsdatascience.com/k-means-clustering-and-the-gap-statistics-4c5d414acd29

Finding Optimal Number Of Clusters for Clustering Algorithm:
https://medium.com/@masarudheena/4-best-ways-to-find-optimal-number-of-clusters-for-clustering-with-python-code-706199fa957c