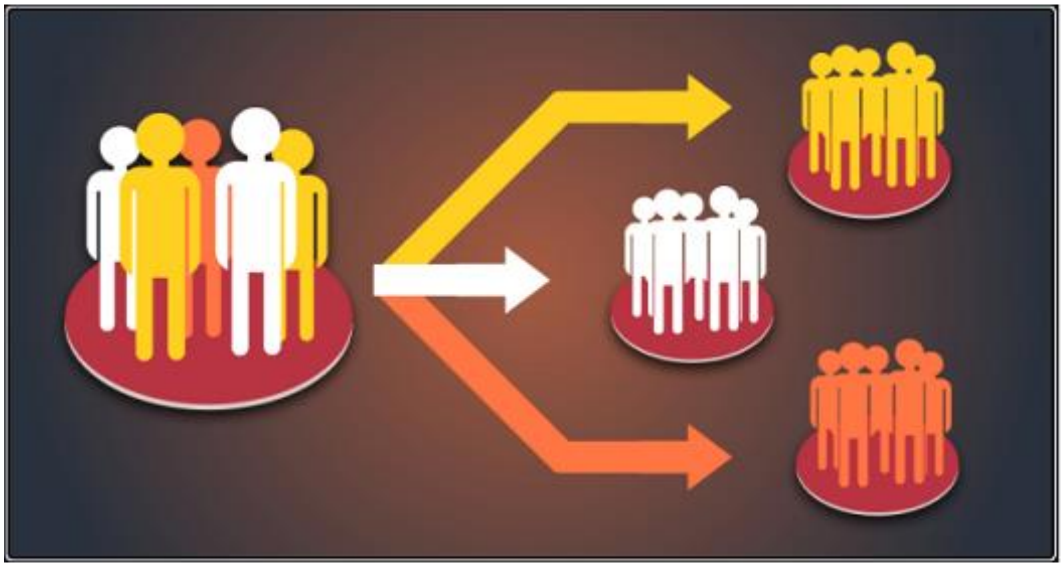
Customer Segmentation using K-Means Clustering

A Machine Learning Project



MSBA 326 Final Project	Dineshkumar Kovai Balasubramanian
Dr. Bahman Zohuri	Fnu Priyansha
Fall 2020	Michael O’Donnell
Golden Gate University	Miriam O'Callaghan
	Reena Sehitya

Table of Contents

Executive Summary

This research paper illustrates the significance and power of customer segmentation by categorizing 495478 customer records at an online retail store into three well-defined clusters. While there are many ways to categorize customers, the segmentation studied here follows the K-Means, machine learning clustering model, and RFM classification. The K-Means model provides information on how many groups should be created and organizes each data point into one of those groups. While K-Means provides the algorithm for classification, the RFM model contains three variables - recency, frequency, and monetary value. It is these three measures that will be used for classification. Recency measures how recently the customer made a purchase. Frequency measures how many purchases the customer made, and monetary value measures how much money the customer spent. Each variable places the customers into one of four groups. As standalone variable groups, RFM provides insight as to which types of customers are most valuable for business. While this is a good insight, it does not maximize the available knowledge.

The true power of the RFM model is leveraged when the three variables (recency, frequency, and monetary value) are combined to create overall business value groups. These final groups consider all three variables and highlight the most valuable customers. Extracting three meaningful segments from 495478 records, allows the online retail company to confidently tailor their business relationships through promotions, services, sales, and feedback with each customer. Customer segmentation creates clarity from chaos and allows a company to have better control of its bottom line.

Introduction

When it comes to achieving optimal results, customer segmentation is one marketing practice that is most essential for a business to operate successfully. According to Harvard Business School professor Clayton Christensen, “as many as 95 percent of annual product launches fail” because of poor market segmentation. While it is not practical to tailor marketing campaigns to each individual customer of interest, it is not wise to attempt a “one-size-fits-all” approach.

Customer segmentation is the practice of partitioning a customer base into groups of individuals that have similar attributes. Each segment has customers with some common characteristics, that influence their buying behavior. For example, one group might contain customers who are high profit and low risk as they are more likely to purchase products or subscribe for a service. While another group might include customers from low profit and high-risk groups. By dividing prospective and current customers into smaller subgroups, it is easier for businesses to effectively allocate marketing resources, to devote more time and attention to dormant customers, and keep offering relevant promotions to the existing ones.

This paper aims to develop a machine learning model to segment customers of an online retail company. There are many ways to segment consumers, such as demographically, geographically, we choose to perform behavioral segmentation using RFM clustering. RFM is based on three pillars of customer attributes: Recency of purchase, Frequency of purchase, and Monetary Value of the purchase. The customers in each cluster are similar to each other with respect to these buying behaviors. To implement the model, we will use an unsupervised machine learning algorithm called K-Means clustering.

K-Means clustering is a very simple and fast algorithm that can efficiently deal with very large data sets such as this one. It classifies objects (customers) in multiple clusters (segments) so that customers within the same segment are as similar as possible, and customers in different segments are as dissimilar as possible. The important steps that we will follow for K-Means clustering include specifying the number of clusters using the elbow method, initialization of centroids by first shuffling the dataset, and then randomly selecting K data points for the centroids without replacement, and then iterating until there is no change to the centroids.

The company in context is an international e-retailer. For this study, we have limited our scope to segmenting customers in the United Kingdom. We chose the UK since, in our database, most of the purchases have been recorded in this country. The details are provided in the analysis section. It is also important to give credit to Shilaja Gupta, whose study available on Kaggle became the very basis of this project (Gupta, n.d.)

Literature Review

The internet revolution and the ongoing digitalization of modern life have revolutionized the retail industry. It is estimated that in 2019, 1.92 billion people purchased goods or services online. Specifically, e-retail sales surpassed 3.5 trillion dollars worldwide during the same year. As the latest calculations inform, e-commerce growth will accelerate even further in the future (Statista, 2020). As good as it sounds, e-retail companies are not always successful in selling their services highly effectively. According to MarketingSignals research reported in Startup 2, 37% of e-commerce startups end up in failure due to poor online marketing (Startup 2, 2019). An effective customer segmentation strategy is paramount for marketing success.

Effective segmentation can help e-retailers in two ways. One, in identifying the most profitable customers that help in maximizing retailer's return on investment. These include the customers who buy the company's products most frequently. Second, in identifying customers' preferences and behaviors. For instance, through segmentation, it is possible to classify specific groups of customers who exhibit similar behaviors such as buying the same products, buying products at the same frequency, and spending similar amounts at purchases (Fuloria, 2011).

In this study, we attempted to identify both – most profitable customers, and their behaviors through the RFM model of customer segmentation. Using RFM, we can segment business customers into various meaningful groups where the main characteristics of each segment could be clearly identified. To do so, we can use one K-Means clustering algorithm to see who our most/least valuable customers are, what are their purchase behaviors in terms of frequency of purchases, and the sales pattern in terms of regions, and time (Chen et al., 2012).

Machine learning and big data have enabled a terrific adoption of automated approaches to customer segmentation. According to a study (Ezenkwu et al., 2015), a group of researchers created customer segments using K-Means clustering with 95% accuracy. They identified segments based on the average amount of goods purchased per month by each customer and the average number of customer visits per month. With the help of such studies, we can be sure that using the K-Means method will help us in creating the right customer segments for our online retail company.

Data Collection

The dataset used for this project is taken from Kaggle. It is a large dataset with 541909 rows and 8 columns. Each row represents one transaction and has the following attributes:

Invoice number – Integer variable that represents transaction id under which the product was purchased.

Stock code – 5-digit integer variable representing each distinct product

Description – Text variable giving a short description of the item

Quantity – The number of each product per transaction

Date – Date and time at which transaction was made

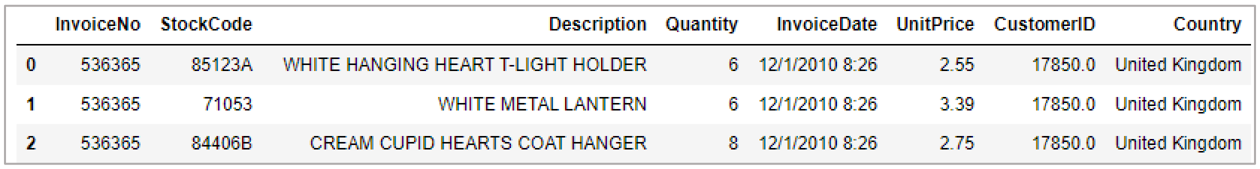
Unit price – Cost of a single unit of the product in sterling

Customer ID – Integer variable representing unique numeric code assigned to each customer

Country of Purchase – Nominal variable. The name of the country where the customer resides

Exploratory Data Analysis and Data Preparation

Figure 1 shows the 8 columns in our dataset representing various attributes of a transaction. Of the eight columns, two can be used for exploratory analysis - Quantity, and Unit Price.



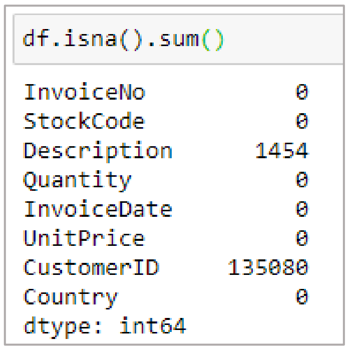
	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	12/1/2010 8:26	2.55	17850.0	United Kingdom
1	536365	71053	WHITE METAL LANTERN	6	12/1/2010 8:26	3.39	17850.0	United Kingdom
2	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	12/1/2010 8:26	2.75	17850.0	United Kingdom

Figure 1

Cleaning the Data

<u>Removing Null Values</u>

We found there were many null values in the description and CustomerID columns (figure 2).



```
df.isna().sum()
```

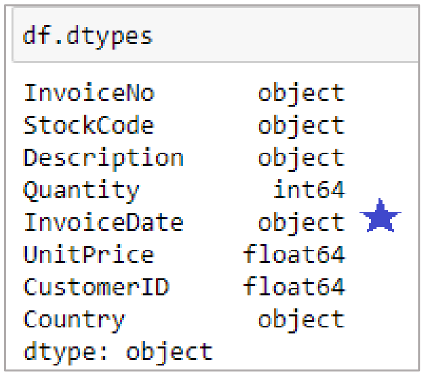
```
InvoiceNo           0
StockCode           0
Description      1454
Quantity            0
InvoiceDate         0
UnitPrice           0
CustomerID     135080
Country             0
dtype: int64
```

Figure 2

Since we have a large dataset, dropping some rows would not make much difference in our analysis. So, we dropped all the rows with null values.

<u>Column Formatting</u>

We found that InvoiceDate column is of "object" type (Figure 3). We converted it to the datetime datatype.

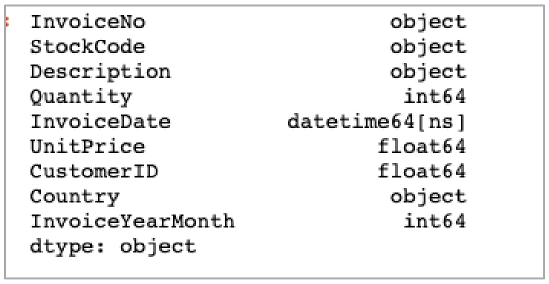


```
df.dtypes
```

```
InvoiceNo       object
StockCode       object
Description     object
Quantity         int64
InvoiceDate     object
UnitPrice      float64
CustomerID     float64
Country         object
dtype: object
```

Figure 3

Next, we dropped the columns that were not relevant for this analysis such as StockCode, and Description. We then created a new InvoiceYearMonth field for the ease of computations, analysis, and visualization as shown in figure 4 below.



```
InvoiceNo                    object
StockCode                    object
Description                  object
Quantity                      int64
InvoiceDate          datetime64[ns]
UnitPrice                   float64
CustomerID                  float64
Country                      object
InvoiceYearMonth              int64
dtype: object
```

Figure 4

Handling Outliers

Further examination of the data shows that anomalies and outliers are present in the dataset. These are rare but influential entries that could negatively affect the accuracy of the final model score, so in the next few steps, we will identify and remove outliers.

Categorical Outliers Country

Figure 5 shows that over 90% of the data transactions come from the United Kingdom. Due to the lack of equal entries by country, the RFM results will be highly skewed toward transactions within the UK. To fix this situation we will narrow down our scope to include only UK transactions.

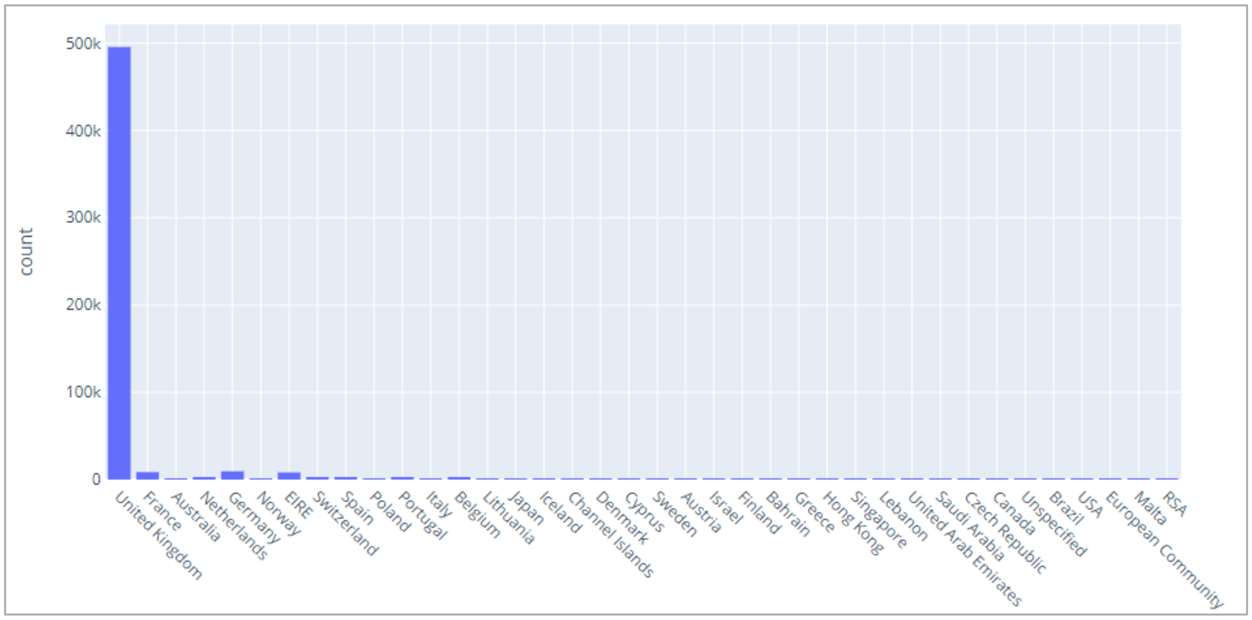
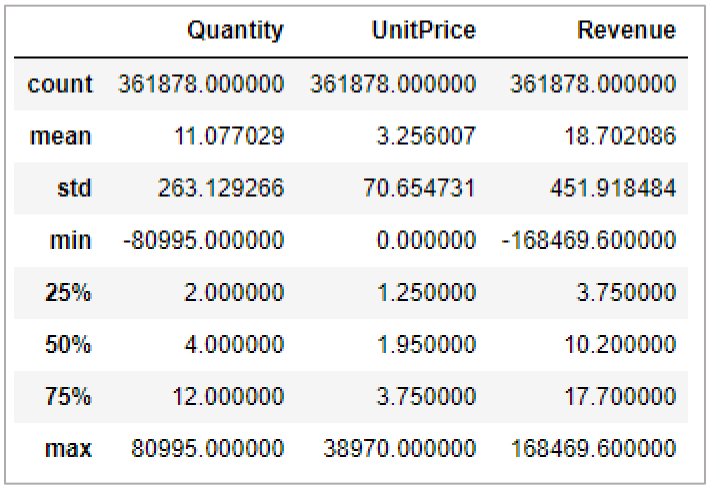


Figure 5

Quantitative Outliers

The descriptive summary of the three quantitative variables, Quantity, Unit Price, and Revenue is listed in figure 6. All three variables show a small interquartile range, but an extremely large range from minimum to the maximum. This observation together with the large standard deviation, illustrates the presence of outliers.



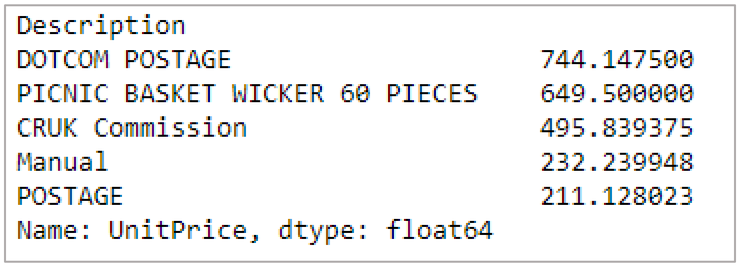
	Quantity	UnitPrice	Revenue
count	361878.000000	361878.000000	361878.000000
mean	11.077029	3.256007	18.702086
std	263.129266	70.654731	451.918484
min	-80995.000000	0.000000	-168469.600000
25%	2.000000	1.250000	3.750000
50%	4.000000	1.950000	10.200000
75%	12.000000	3.750000	17.700000
max	80995.000000	38970.000000	168469.600000

Figure 6

As discussed before, for the exploratory analysis we will be focusing on Unit Price and the Quantity variables. Let us now explore these variables individually and see how they are affecting the overall data.

Unit Price

A filtered observation of the highest mean values brings up some susceptible "products" affecting the spread of the data (Figure 7).



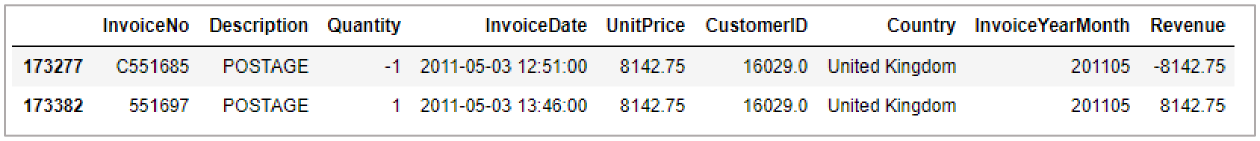
```
Description
DOTCOM POSTAGE                    744.147500
PICNIC BASKET WICKER 60 PIECES    649.500000
CRUK Commission                   495.839375
Manual                            232.239948
POSTAGE                           211.128023
Name: UnitPrice, dtype: float64
```

Figure 7

We found 4 items below have unusual unit price and they do not seem like a specific product:

1. DOTCOM POSTAGE - seems to indicate the amount spent by the customer on postage
2. CRUK Commission - fee paid out to an external cancer research organization
3. Manual - refers to manual services rendered with the purchase like furniture assembly
4. Discount - benefit offered as a reward for purchase

All these items are not a direct indicator of sales and are heavily skewing the data. Furthermore, some suspicious transactions are not apparent from figure 7 above because offsetting costs kept the mean below detection. However, a second filtered search reveals two more "non-products" (Figure 8).



	InvoiceNo	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country	InvoiceYearMonth	Revenue
173277	C551685	POSTAGE	-1	2011-05-03 12:51:00	8142.75	16029.0	United Kingdom	201105	-8142.75
173382	551697	POSTAGE	1	2011-05-03 13:46:00	8142.75	16029.0	United Kingdom	201105	8142.75

Figure 8

The results above inform us that fees, commission, and postage are not actual products but are included within the products section. It was these elements that that were the source of outliers. These elements are removed and, for the same reason as the non-UK countries, will not be a part of our dataset. After removing these elements, our unit prices vary mostly from $0 to $700. From the histogram, in figure 9 we can see that most of the prices fall between $0 to $50 after we focused on unit prices under 100.

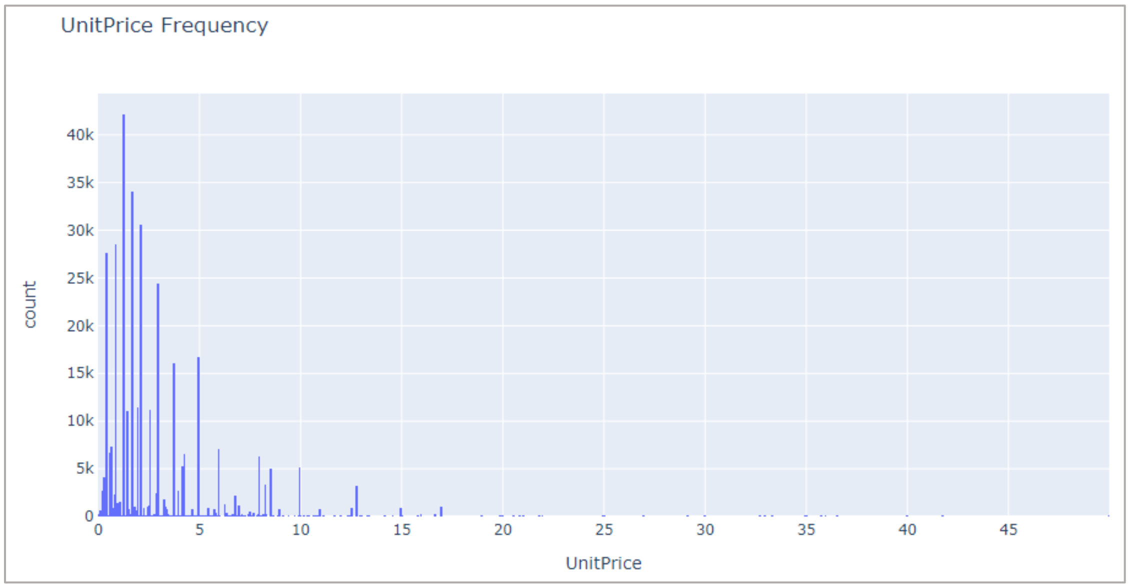


Figure 9

Quantity

After removing the outliers from the "Unit Price", we then focused on "Quantity". By running a quick filter, we found that in our dataset of 490K rows, only 106 transactions are for more than 1000 quantities out of which most of them have refund transactions. This indicates they were bought by mistake, and thus have been removed. The histogram below shows most transactions based on the quantity that falls between 0 to 100 (figure 10).

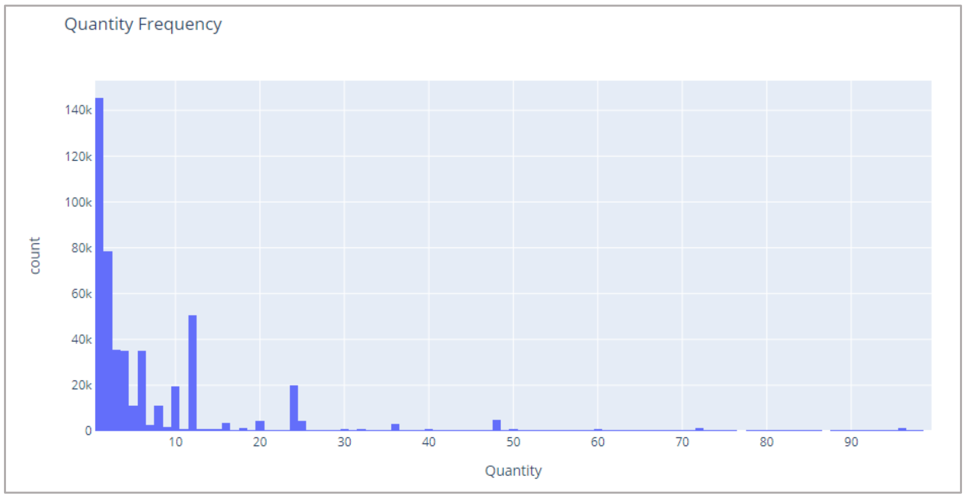


Figure 10

Summarizing EDA

After the outliers were removed, we could see the top ten selling products as shown in figure 11.

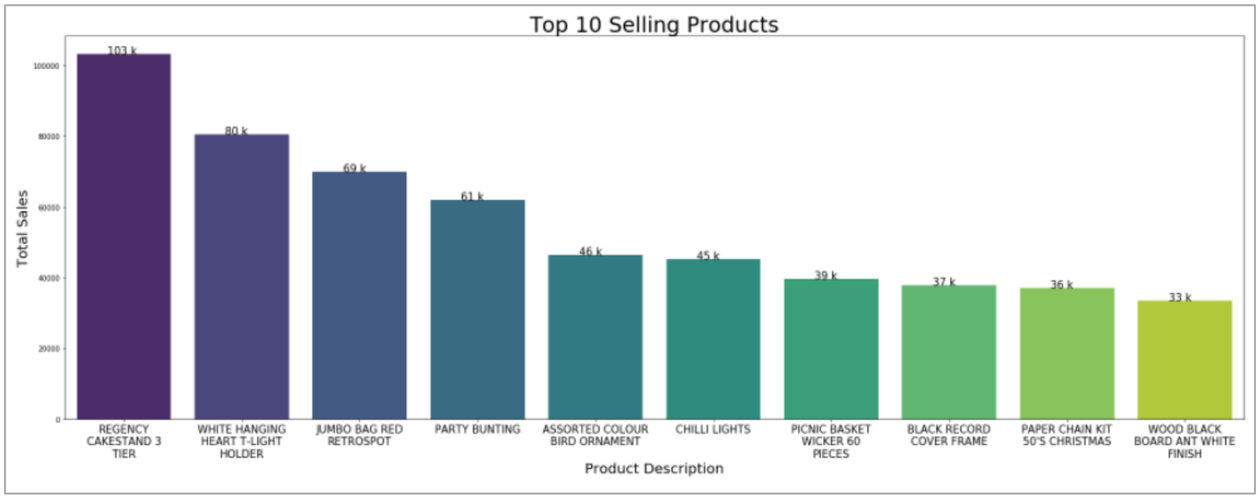
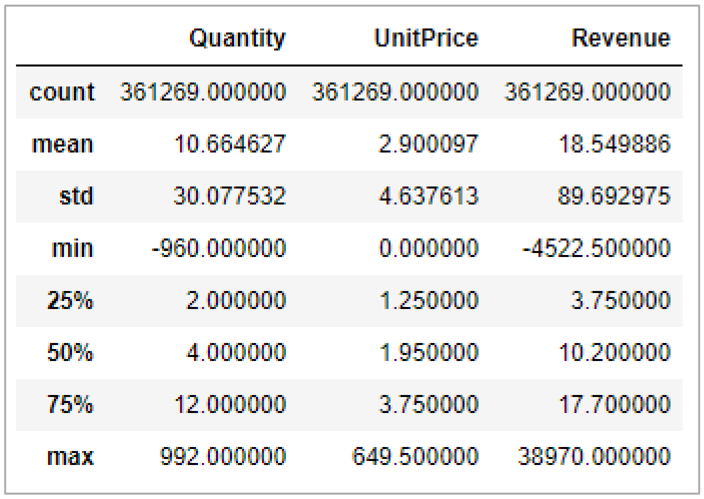


Figure 11

We then calculated the descriptive summary with our clean data that can be seen in figure 12. From the results, it is quite evident that our data looks much more balanced now with low skewness and a more manageable spread (range).



	Quantity	UnitPrice	Revenue
count	361269.000000	361269.000000	361269.000000
mean	10.664627	2.900097	18.549886
std	30.077532	4.637613	89.692975
min	-960.000000	0.000000	-4522.500000
25%	2.000000	1.250000	3.750000
50%	4.000000	1.950000	10.200000
75%	12.000000	3.750000	17.700000
max	992.000000	649.500000	38970.000000

Figure 12

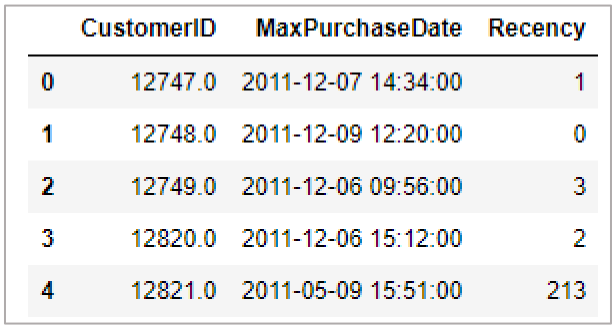
RFM Segmentation using K-Means Clustering

After understanding the nature of our data through EDA and preparing data to be used for further computations, we are now ready to apply RFM segmentation using the K-Means clustering method.

Recency

Below are the steps we took to calculate the Recency score of each customer:

1. We found the latest purchase date of the entire dataset as a baseline (Figure 13).



	CustomerID	MaxPurchaseDate	Recency
0	12747.0	2011-12-07 14:34:00	1
1	12748.0	2011-12-09 12:20:00	0
2	12749.0	2011-12-06 09:56:00	3
3	12820.0	2011-12-06 15:12:00	2
4	12821.0	2011-05-09 15:51:00	213

Figure 13

2. We found the most recent date on which every customer made the purchase and subtracted them from the baseline value found in step 1. We then plotted these recency values on a histogram to see the distribution of the number of customers in each range. From figure 14 we can see that the data is skewed to the right which states that most of the customers have purchased in the last 100 days.

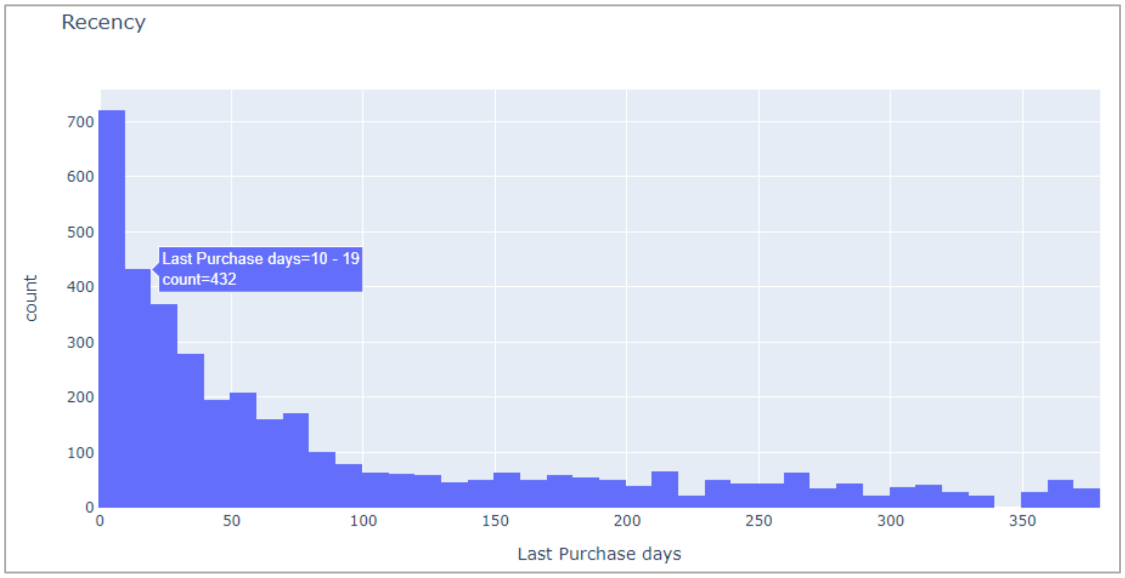


Figure 14

3. The elbow method was then used to choose the number of clusters for K-Means clustering. This method helps us to select the optimal number of clusters by fitting the model with a range of values of K. We can plot a chart and if this line chart resembles an arm, then the elbow which is the point of inflection on the curve is the good indication that this model fits best at that point. In the resulted plot (figure 15), 3 seems the optimum number of clusters. However, we tried to expand our scope and decided to go ahead with 4 clusters of customers to have more defined clusters.

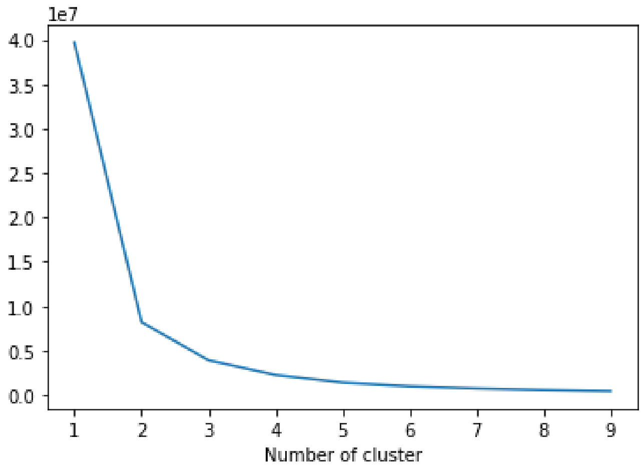
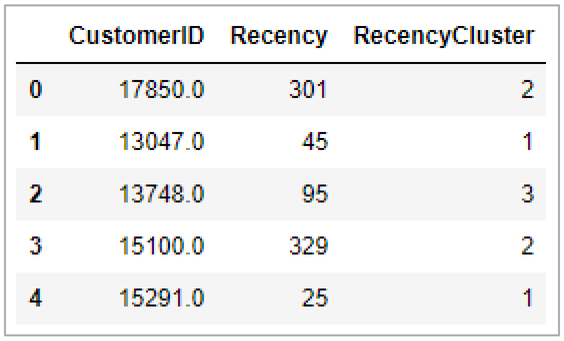


Figure 15

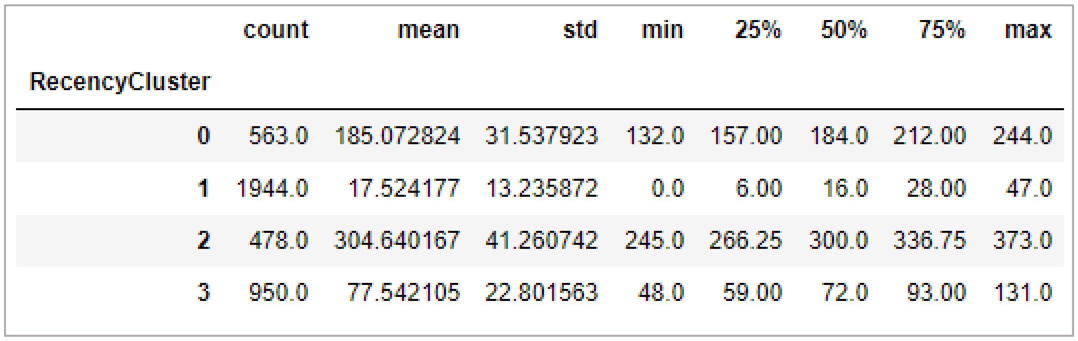
4. We then used K-Means clustering to assign the recency score to each customer (figure 16)



	CustomerID	Recency	RecencyCluster
0	17850.0	301	2
1	13047.0	45	1
2	13748.0	95	3
3	15100.0	329	2
4	15291.0	25	1

Figure 16

Descriptive statistics of recency scores can be seen below in figure 17.



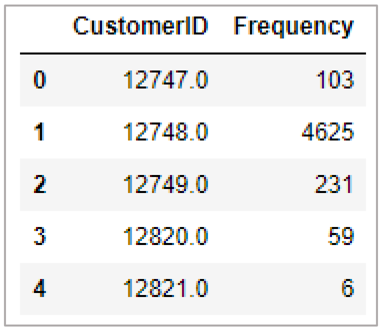
	count	mean	std	min	25%	50%	75%	max
RecencyCluster								
0	563.0	185.072824	31.537923	132.0	157.00	184.0	212.00	244.0
1	1944.0	17.524177	13.235872	0.0	6.00	16.0	28.00	47.0
2	478.0	304.640167	41.260742	245.0	266.25	300.0	336.75	373.0
3	950.0	77.542105	22.801563	48.0	59.00	72.0	93.00	131.0

Figure 17

All the customers are divided into 4 clusters based on their recency score. We can see customers from cluster 1 are the most attractive customers. The most recent purchase made by them was just 0 days ago while the oldest purchase was made within the last 47 days. On the other hand, customers in cluster 2 are the most inactive who have ordered products as earlier as 373 days ago.

Frequency

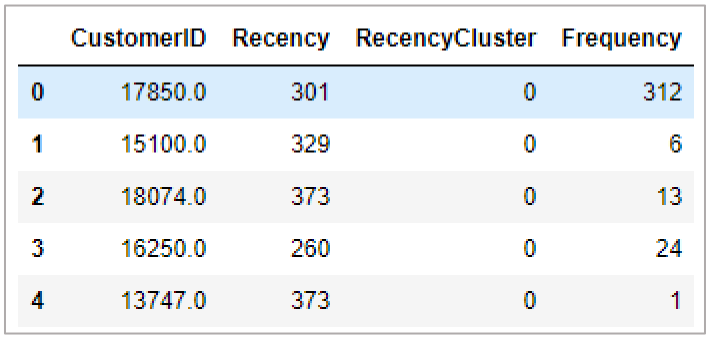
Now that we have calculated Recency, our next step is to perform calculations on the Frequency aspect of RFM. To begin with the frequency cluster, we first find out the total number of orders per customer and the frequency of the orders per customer (Figure 18).



	CustomerID	Frequency
0	12747.0	103
1	12748.0	4625
2	12749.0	231
3	12820.0	59
4	12821.0	6

Figure 18

We ran "uk_frequency.head()" to confirm if we have created the desired dataframe "uk_frequency" by grouping on "CustomersID" using "**count**" aggregate function successfully. Then we merged the uk_frequency dataframe to our generic dataframe "ukdf_user" and checked first five rows (Figure 19).



	CustomerID	Recency	RecencyCluster	Frequency
0	17850.0	301	0	312
1	15100.0	329	0	6
2	18074.0	373	0	13
3	16250.0	260	0	24
4	13747.0	373	0	1

Figure 19

Next, we plotted a histogram of ‘frequency” column of ukdf_user dataframe using Plotly's Python graphing library (Figure 20).

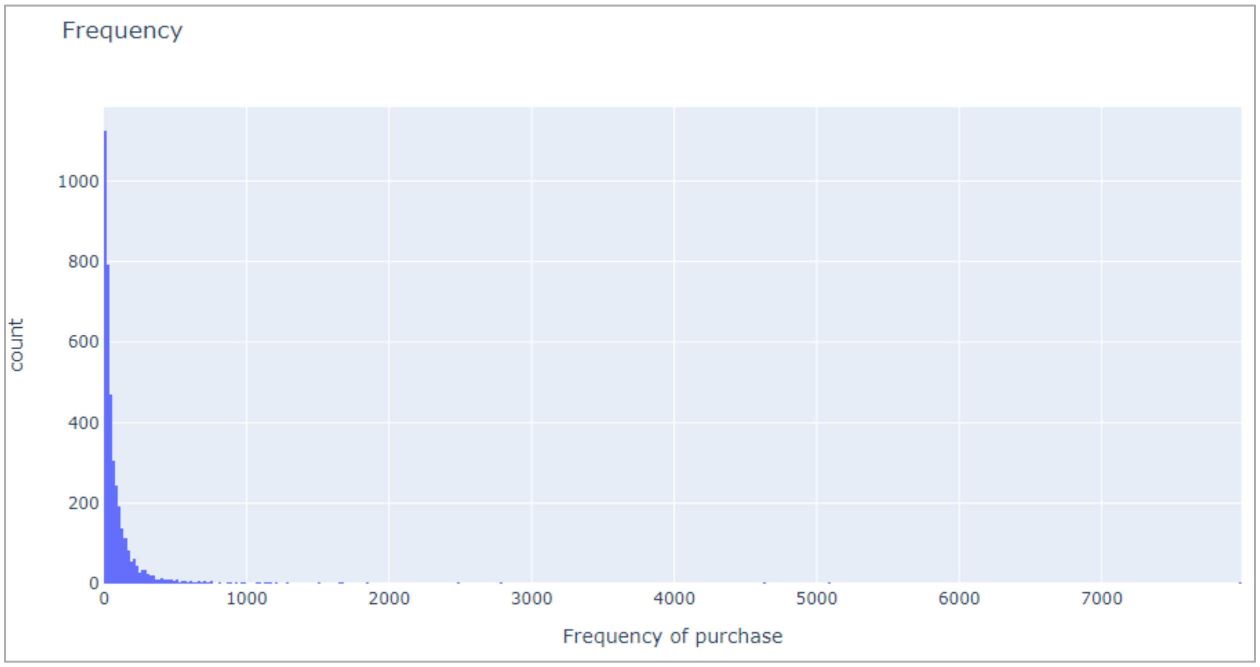


Figure 20

From figure 20, it is evident that we are getting most of the frequency below 1000. We zoomed in within 1000 limit of frequency to get a clear picture of customer’s behavior in terms of frequency of buying (Figure 21)

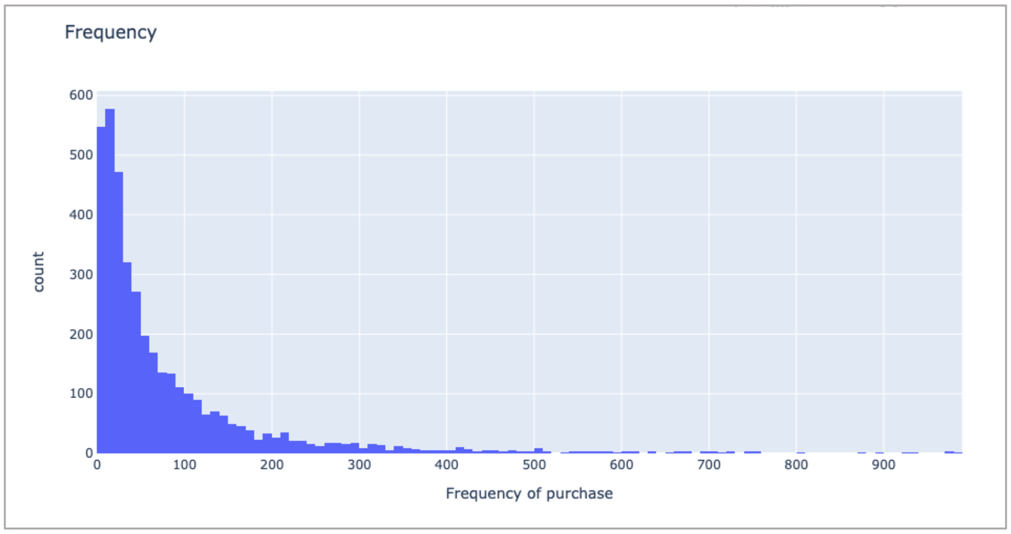


Figure 21

Figure 21 shows that the frequency of the maximum number of customers is within 0-100 and beyond 200, the count of customers is tapering down. We can also see that there are some outliers present. The pattern of frequency after 400 shows that there are only a few customers whose frequency of purchasing is passing beyond 600-800 level.

In the next step, we ran the code to get the right number of clusters for K-Means using the elbow method. The resulted graphic can be seen in figure 22.

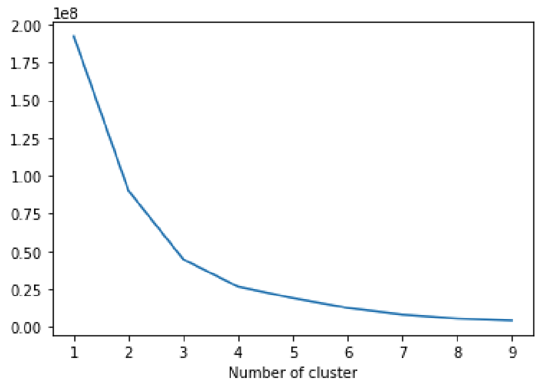
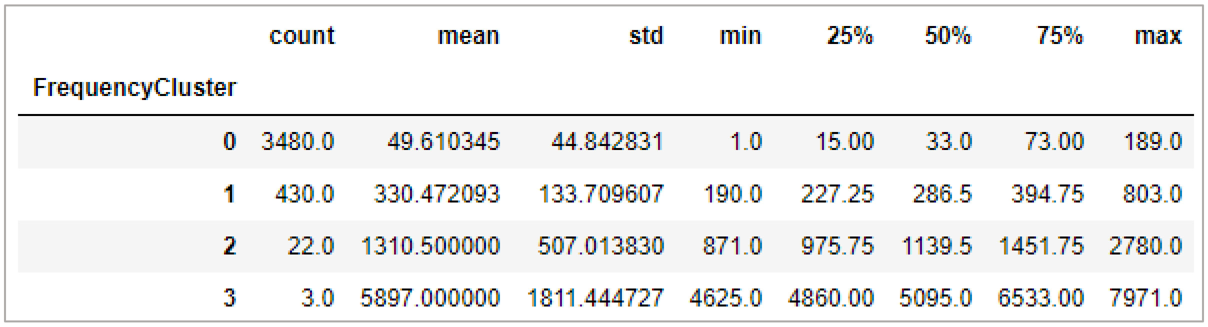


Figure 22

In figure 22, the elbow method suggests that the optimum number of clusters should be 4. Beyond that, the graph is plummeting down. We, therefore, decided to create 4 number of clusters.

We instantiated the instance kmeans of KMeans method and fit our "frequency" column into our model. Then we performed calculations to call frequency clusters grouped by the descriptive statistics of each cluster (figure 23).



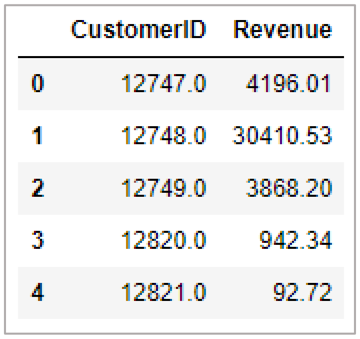
	count	mean	std	min	25%	50%	75%	max
FrequencyCluster								
0	3480.0	49.610345	44.842831	1.0	15.00	33.0	73.00	189.0
1	430.0	330.472093	133.709607	190.0	227.25	286.5	394.75	803.0
2	22.0	1310.500000	507.013830	871.0	975.75	1139.5	1451.75	2780.0
3	3.0	5897.000000	1811.444727	4625.0	4860.00	5095.0	6533.00	7971.0

Figure 23

The results in figure 23 show that the cluster with maximum frequency is cluster 3 and the least frequency cluster is cluster 0. After we found the clusters of frequency, let's see how clusters based on Monetary Value looks like.

Monetary Value

For the sake of convenience, we are using the term Revenue to represent Monetary Value. To begin, we first calculated the revenue for each customer and created a new dataframe "uk_revenue" grouped by CustomersID on Revenue column using "sum" aggregate function. The results can be seen in figure 24.



	CustomerID	Revenue
0	12747.0	4196.01
1	12748.0	30410.53
2	12749.0	3868.20
3	12820.0	942.34
4	12821.0	92.72

Figure 24

After checking the first five rows of uk_revenue dataframe, we merged this dataframe with our main dataframe "ukdf_user". We then plotted the histogram of ukdf_user (Revenue) column on a reduced scale in order to get proper visualization (Figure 25).

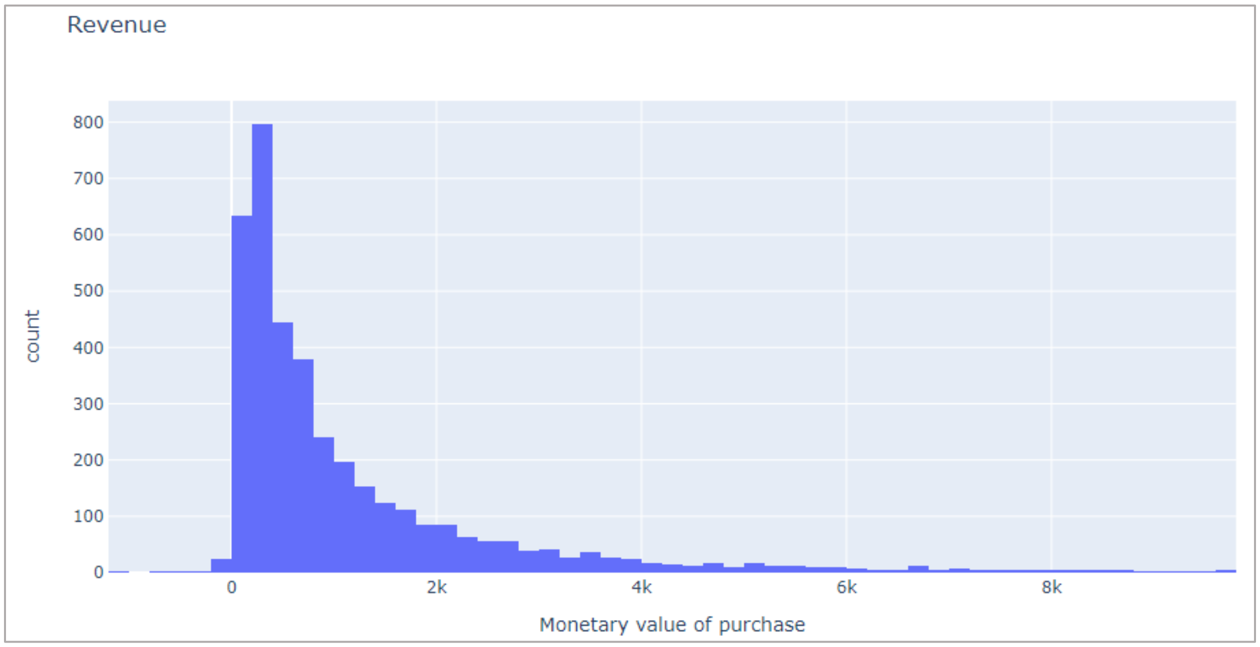


Figure 25

Then we applied the Elbow method to find out the optimum number of clusters for K-Means (Figure 26)

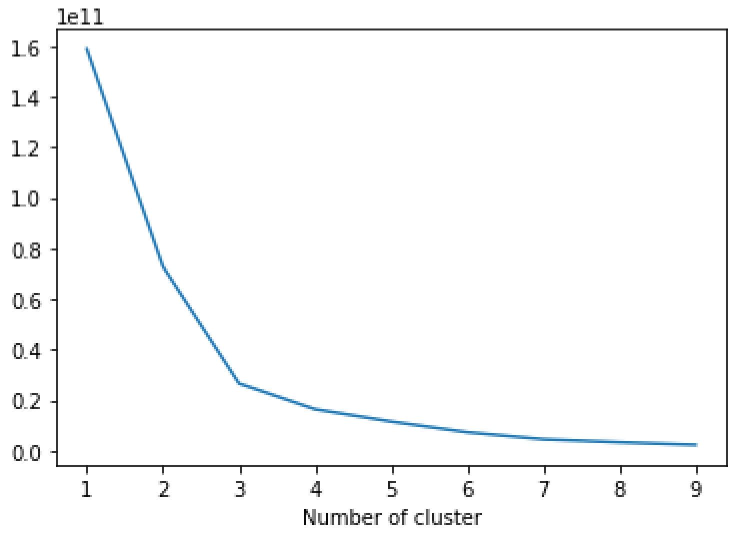
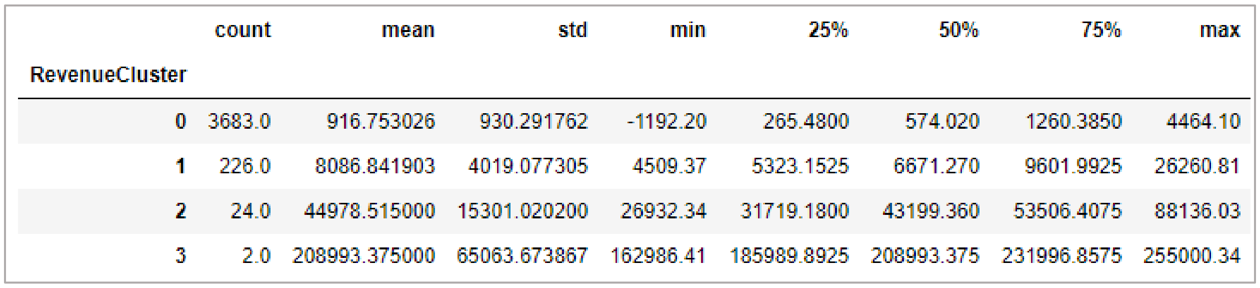


Figure 26

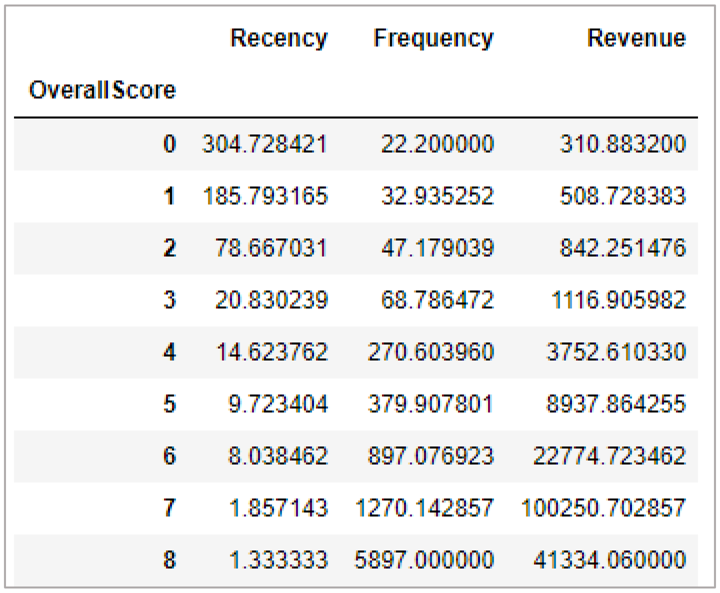
The result derived using the Elbow method above suggests optimal clusters can be 3 or 4. We will create 4 clusters. We instantiated the instance of KMeans algorithm and fit-out ukdf_user (Revenue) variable into our K-Means model. Then finally we ordered the cluster number and grouped it by RevenueCluster on ukdf_user (Revenue) to get statistics shown in figure 27.



RevenueCluster	count	mean	std	min	25%	50%	75%	max
0	3683.0	916.753026	930.291762	-1192.20	265.4800	574.020	1260.3850	4464.10
1	226.0	8086.841903	4019.077305	4509.37	5323.1525	6671.270	9601.9925	26260.81
2	24.0	44978.515000	15301.020200	26932.34	31719.1800	43199.360	53506.4075	88136.03
3	2.0	208993.375000	65063.673867	162986.41	185989.8925	208993.375	231996.8575	255000.34

Figure 27

It seems that cluster 3 has maximum revenue and cluster 0 has the lowest revenue.
Now that we have cluster numbers assigned for recency, frequency & revenue, we then created an overall score for all the clusters. We created a new column "ukdf_user['OverallScore']" and assigned values to this new variable by simply adding ukdf_user['RecencyCluster'] + ukdf_user['FrequencyCluster'] + ukdf_user['RevenueCluster']. Then grouped by "OverallScore" on 'Recency','Frequency','Revenue' using "mean" aggregate function (Figure 28).



Overall Score	Recency	Frequency	Revenue
0	304.728421	22.200000	310.883200
1	185.793165	32.935252	508.728383
2	78.667031	47.179039	842.251476
3	20.830239	68.786472	1116.905982
4	14.623762	270.603960	3752.610330
5	9.723404	379.907801	8937.864255
6	8.038462	897.076923	22774.723462
7	1.857143	1270.142857	100250.702857
8	1.333333	5897.000000	41334.060000

Figure 28

Based on the overall score above, it seems that customers with a score of 7 are our most attractive customers and the ones with a score of 0 are the least attractive ones.

Segment Visualization

Finally, we plotted all these segments on scatter plots to identify customers with the highest and lowest lifetime value. In each of the below visualizations, we can see that customers in the red cluster are the best prospects as they seem to have significant recent purchases and have higher frequency with higher monetary value in their order history. On the contrary, customers in the blue cluster seem to have the oldest last purchase date, the lowest frequency, and the lowest monetary value of the purchase. Figures 29, 30, and 31 show the graphic visualization of our key clusters based upon Revenue vs Frequency (Figure 29), Revenue vs. Recency (Figure 30), and Frequency vs. Recency (Figure 31).

The interpretation of these visuals will be shared in the findings and conclusion section.

Revenue vs Frequency

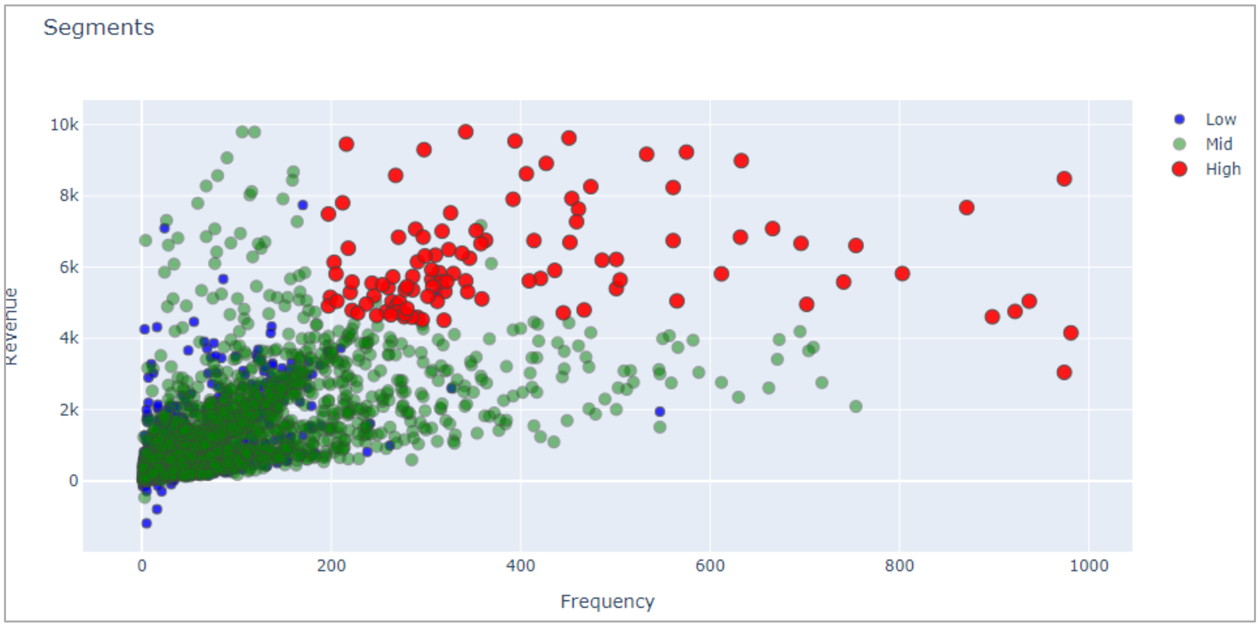


Figure 29

Revenue vs Recency

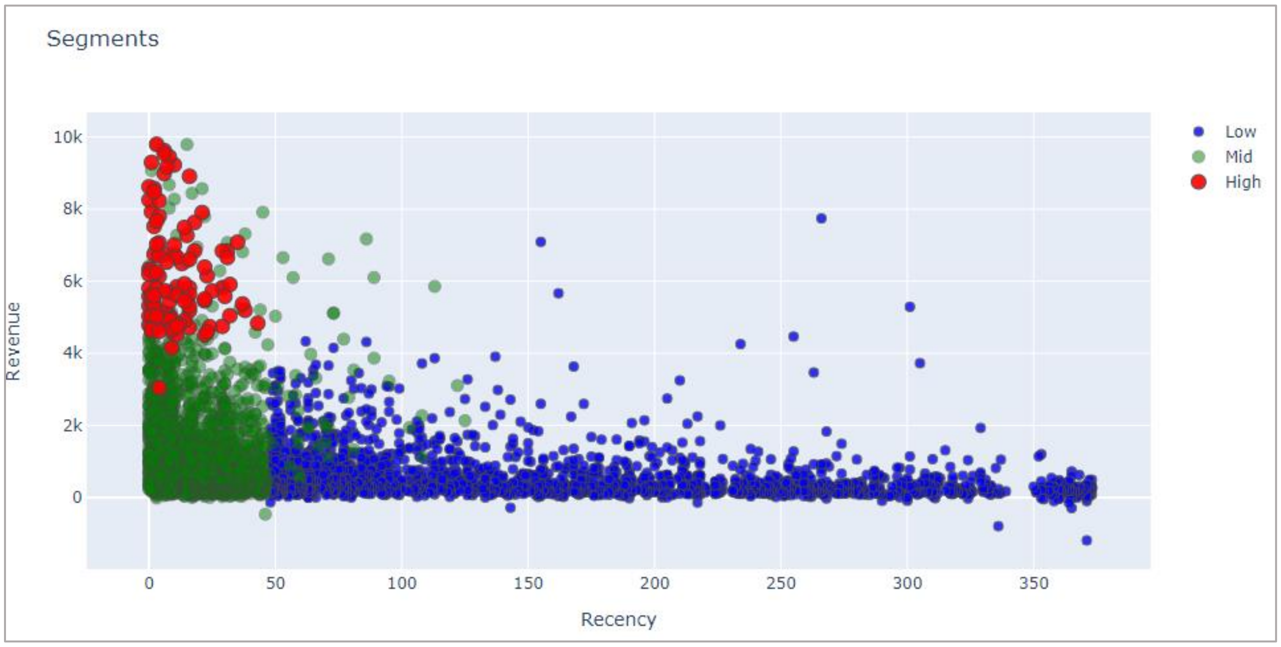


Figure 30

Frequency vs Recency

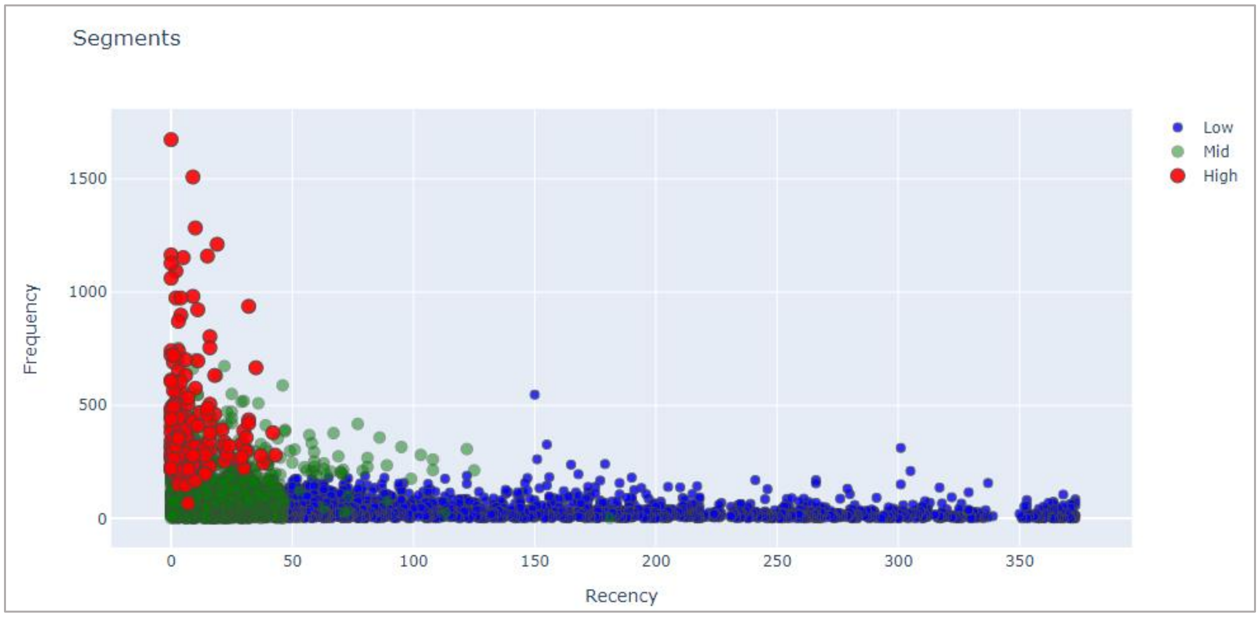


Figure 31

Findings and Conclusion

Before we conclude this study, it is important to interpret and understand each cluster. To generate customer-centric business intelligence, we also need to list down the important findings of this study. As shown in the visualization, we could successfully create clusters based upon customers' recency of purchase, the frequency, and how much money they spent on their purchases. We will compare the visualization information with numerical information in figure 28 to create our market segments. Based on RFM, our study generated 8 customer categories in total (Figure 28). Using these calculations and clusters in the visualization, we can now create our high priority and low priority customer segments.

As consultants, we would claim that customers with an overall score of 7, 8, and 6 in figure 28 should be our top priority – Priority 1 segment for our business (scale: priority 1 being most important to priority 3 being least important). These customers seem to be clustered together as red dots in the visualizations. They are our high priority segment. Together, the customers in this cluster have generated revenue of $164,358 (approximately). Their buying frequency is significantly high with a total of approximately 8064 times. Moreover, this segment seems to have made purchases most recently with a score between 1.33 to 8.

Our Priority 2 segment is the one depicted by green dots on the scatter plots. If we compare this information with figure 28, we can easily say that this cluster comprises of customer categories that scored 3, 4, and 5 as their overall scores. The total revenue of this cluster should be about $13805 which is quite low as compared to the Priority 1 segment but still not insignificant. The frequency of purchases in this cluster is moderate and recency scores are between 9 and 20, not quite attractive. The lowest priority segment, Priority 3 is one that is depicted by blue dots on the scatter plots. In figure 28, this cluster is categorized with three categories that scored 0, 1, and 2 as their overall scores. The total revenue earned by the customers in this cluster is only about $1660 (approximately) and their frequency scores are also significantly low. On the recency scale, this cluster looks even worse with scores as high as between 78 and 304.

The key bottom-line of every business is revenue which is driven by effective marketing. To market our products effectively, we will focus on our top priority segments and use these customers as the top targets for our promotional offers. Through this study, we also identified where not to invest our resources – low priority segment – Priority 3. Most of the promotion budget should be spent on our Priority 1 segment and the left-over resources should be used on the Priority 2 segment.

In conclusion, we can say that K-Means clustering coupled with the RFM technique can help companies identify their high and low priority market segments. Using such methods, businesses can profile their customers based on their demographic and behavioral predictors. Analytics have helped Best Buy identify the fact that 7% of Best Buy customers accounted for 43% of its total sales. Using this information, Best Buy reorganized its stores to address the needs of its high-value customers (Fuloria, 2011). Retailers can maximize their return on investment significantly by correctly identifying the most profitable, high priority customer segments.

For e-retailers, the application of machine learning algorithms is not limited to clustering. It is a lot more that we can do such as analyzing cross-shopping behavior, predicting what products should be sold together, optimizing prices, creating recommendation systems, and analyzing the inventory procurement needs. One advantage of using machine learning algorithms such as K-Means clustering is that businesses can accurately test various hypotheses using quantitative results generated by algorithms.

References

Chen, D., Sain, S. L., & Guo, K. (2012). Data mining for the online retail industry: A case study of RFM model-based customer segmentation using data mining. *Journal of Database Marketing & Customer Strategy Management*, *19*(3), 197–208. https://doi.org/10.1057/dbm.2012.17

E-commerce worldwide. (2020). Statista. https://www.statista.com/topics/871/online-shopping/

Elbow Method — Yellowbrick v1.2 documentation. (n.d.). https://www.scikityb.org/en/latest/api/cluster/elbow.html

Ezenkwu, C. P., Ozuomba, S., & Kalu, C. (2015). Application of K-Means Algorithm for Efficient Customer Segmentation: A Strategy for Targeted Customer Services. *International Journal of Advanced Research in Artificial Intelligence*, *4*.

Fuloria, S. (2011) How Advanced Analytics Will Inform and Transform U.S. Retail. Cognizant Reports, July, http://www.cognizant.com/InsightsWhitepapers/How-Advanced-Analytics-Will-Inform-and-Transform-US-Retail.pdf.

Gupta, S. (n.d.). Customer Segmentation: RFM Clustering. https://www.kaggle.com/shailaja4247/customer-segmentation-rfm-clustering

Kocina, L. (2017, May). What percentage of new products fail and why? https://www.publicity.com/marketsmart-newsletters/percentage-new-products-fail/

Medvedev, S. (n.d.). Customer segmentation dataset. https://www.kaggle.com/sergeymedvedev/customer_segmentation

Qiao, F. (2018, Aug). Cakestands & Paper Birdies: E-Commerce Cohort Analysis in Python. https://towardsdatascience.com/cakestands-paper-birdies-e-commerce-cohort-analysis-in-python-e33d0cf70dfc

Startup 2, M. G. I. (2019, July 21). Why Ecommerce Fails - The Top 10 Reasons and How to Avoid Them. *Small Business Trends*. https://smallbiztrends.com/2019/07/why-ecommerce-fails.html

Appendix

The Code

Note: various results of this code are pasted as figures throughout the paper.

```
# Importing libraries
from __future__ import division
from IPython.display import Image
import warnings
warnings.filterwarnings('ignore')
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import pandas as pd
import seaborn as sb
from textwrap import wrap
import statsmodels.api as sm

from datetime import datetime, timedelta,date
from sklearn.metrics import classification_report,confusion_matrix
from sklearn.model_selection import KFold, cross_val_score, train_test_split
import seaborn as sns
from sklearn.cluster import KMeans
import plotly as py
import plotly.express as px
import plotly.offline as pyoff
import plotly.graph_objs as go
import matplotlib.pyplot as plt
import xgboost as xgb
import plotly.graph_objs as go
import plotly.offline as py

pd.options.mode.chained_assignment = None  # default='warn'

# Loading the data
filename = 'customer_segmentation.csv'
df = read_csv(filename, encoding="ISO-8859-1")

Data Exploration and Preparation

df.head(3)
df.isna().sum()
df = df.dropna()
df.isna().sum()
df.dtypes

#converting the type of Invoice Date Field from string to datetime.
df['InvoiceDate'] = pd.to_datetime(df['InvoiceDate'])

#creating YearMonth field for the ease of reporting and visualization
df['InvoiceYearMonth'] = df['InvoiceDate'].map(lambda date: 100*date.year + 
date.month)
df.dtypes

# Let's see which country gets maximum number of orders
```

```
fig = px.histogram(df, x="Country")
fig.update_layout(xaxis = go.layout.XAxis(tickangle = 45))
fig.show()
ukdf=df.loc[df['Country'] == 'United Kingdom']

Finding Outliers
# Creating a new column for Total amount per transaction
ukdf['Revenue'] = ukdf['Quantity']*ukdf['UnitPrice']
# Taking a look at negative values.
NegTrans = ukdf[ukdf['Revenue'] < 0]
NegTrans.head(5)
# Dropping unnecessary column
ukdf = ukdf.drop(['StockCode'], axis =1)
ukdf.shape

# Descriptive Analysis before removing outliers
ukdf[['Quantity', 'UnitPrice','Revenue']].describe()

   1. Unit Price

ukdf.groupby('Description').mean()['UnitPrice'].nlargest()

df[df['Description']== 'DOTCOM POSTAGE']['UnitPrice'].describe()
df[df['Description']== 'Manual']['UnitPrice'].describe()
df[df['Description']== 'CRUK Commission'].head()
df[df['Description']== 'Discount'].head()
removeitems = ['DOTCOM POSTAGE', 'CRUK Commission', 'Manual','Discount']
ukdf = ukdf[~ukdf['Description'].isin(removeitems)]

# Let's check if we still have any rows with unusual unitprice i.e. greater
than 2K
ukdf[ukdf.UnitPrice > 2000]
removeinvoice1 = ['C551685', '551697']
ukdf = ukdf[~ukdf['InvoiceNo'].isin(removeinvoice1)]

# So, we deleted 1315 rows, but since we still have 494K of data it should be
enough.
ukdf.shape
ukdf.head()

Let's plot frequency distribution of some of the relevant columns¶
1. UnitPrice

fig = px.histogram(ukdf, x="UnitPrice",
                 labels={
                     "UnitPrice": "UnitPrice",
                     },
                title="UnitPrice Frequency")
fig.show()

# As we see most of the unit prices are under 50, we will take a closer look
ukdf1=ukdf.query('UnitPrice < 50')['UnitPrice']
fig = px.histogram(ukdf1, x="UnitPrice",
                 labels={
                     "UnitPrice": "UnitPrice",
                     },
                title="UnitPrice Frequency")
```

```
fig.show()
```

2. Quantity

```
ukdf[(ukdf['Quantity'] > 1000) | (ukdf['Quantity'] < -1000)]

# We will remove all the quantities above 1K and their corresponding return
transaction of less than -1K.
ukdf = ukdf[(ukdf['Quantity'] > -1000)]
ukdf = ukdf[(ukdf['Quantity'] < 1000)]
ukdf2 = ukdf[ukdf.Quantity > 0]        # leaving out negative values as they
are return items
fig = px.histogram(ukdf2, x="Quantity",
                 labels={
                     "Quantity": "Quantity",
                     },
                title="Quantity Frequency")
fig.show()

# Most number of quantities lie below 100, so taking a closer look at the
histogram
ukdf3=ukdf2.query('Quantity < 100')['Quantity']
fig = px.histogram(ukdf3, x="Quantity",
                 labels={
                     "Quantity": "Quantity",
                     },
                title="Quantity Frequency")
fig.show()
```

Finding top ten selling items by their total sales

```
sales_order = ukdf.groupby('Description').sum()['Revenue'].nlargest(10)

plt.figure(figsize = (30,10))
ax = sb.barplot(x = sales_order.index, y = sales_order.values, palette =
'viridis')
ax.set_xlabel('Product Description', fontsize = 20)
ax.set_ylabel('Total Sales', fontsize = 20)
ax.set_title('Top 10 Selling Products', fontsize = 30)

labels = [ '\n'.join(wrap(l, 15)) for l in sales_order.index ]
ax.set_xticklabels(labels, fontsize = 15)

value_labels = []
for x in sales_order.values:
    value_labels.append(str(int(x/1000))+' k')

for p, label in zip(ax.patches, value_labels):
    ax.annotate(label, (p.get_x() + 0.26, p.get_height() + 2), fontsize = 15)

ukdf.shape   # Even after cleaning up the outliers, we still have 490K
records so we should be good to go.

# Descriptive Analysis of relevant columns after removing outliers
ukdf[['Quantity', 'UnitPrice','Revenue']].describe()
```

Customer segmentation using RFM Clustering

```
#creating a generic user dataframe to keep CustomerID and new segmentation
scores
ukdf_user = pd.DataFrame(ukdf['CustomerID'].unique())
ukdf_user.columns = ['CustomerID']
ukdf_user.head()

Recency Analysis
# Creating a dataframe with max purchase date for each customer
uk_max_purchase = ukdf.groupby('CustomerID').InvoiceDate.max().reset_index()
uk_max_purchase.columns = ['CustomerID','MaxPurchaseDate']
uk_max_purchase.head(3)

# Comparing the last transaction of the dataset with last transaction dates
of the individual customer IDs.
uk_max_purchase['Recency'] = (uk_max_purchase['MaxPurchaseDate'].max() -
uk_max_purchase['MaxPurchaseDate']).dt.days
uk_max_purchase.head()

#merging this dataframe to the new user dataframe
ukdf_user = pd.merge(ukdf_user, uk_max_purchase[['CustomerID','Recency']],
on='CustomerID')
ukdf_user.head()

# Plotting Recency histogram
fig = px.histogram(ukdf_user, x="Recency",
                 labels={
                     "Recency": "Last Purchase days",
                     },
                title="Recency")
fig.show()

# Applying K-Means and using elbow method

from sklearn.cluster import KMeans
sse={} # error
uk_recency = ukdf_user[['Recency']]
for k in range(1, 10):
    kmeans = KMeans(n_clusters=k, max_iter=1000).fit(uk_recency)
    uk_recency["clusters"] = kmeans.labels_  #cluster names corresponding to
recency values
    sse[k] = kmeans.inertia_ #sse corresponding to clusters
plt.figure()
plt.plot(list(sse.keys()), list(sse.values()))
plt.xlabel("Number of cluster")
plt.show()

#building 4 clusters for recency and add it to dataframe
kmeans = KMeans(n_clusters=4)
ukdf_user['RecencyCluster'] = kmeans.fit_predict(ukdf_user[['Recency']])
ukdf_user.head()
ukdf_user.groupby('RecencyCluster')['Recency'].describe()

# Building the function
def order_cluster(cluster_field_name, target_field_name,df,ascending):
    new_cluster_field_name = 'new_' + cluster_field_name
    df_new =
df.groupby(cluster_field_name)[target_field_name].mean().reset_index()
    df_new =
df_new.sort_values(by=target_field_name,ascending=ascending).reset_index(drop
=True)
    df_new['index'] = df_new.index
    df_final = pd.merge(df,df_new[[cluster_field_name,'index']],
on=cluster_field_name)
    df_final = df_final.drop([cluster_field_name],axis=1)
    df_final = df_final.rename(columns={"index":cluster_field_name})
    return df_final
ukdf_user = order_cluster('RecencyCluster', 'Recency',ukdf_user,False)
ukdf_user.head()
ukdf_user.groupby('RecencyCluster')['Recency'].describe()

Frequency Analysis

# To create frequency clusters, we need to find total number of orders for
each customer.
#get order counts for each user and create a dataframe with it
uk_frequency = ukdf.groupby('CustomerID').InvoiceDate.count().reset_index()
uk_frequency.columns = ['CustomerID','Frequency']
uk_frequency.head() # number of orders per customer
# We will add this data to our generic dataframe
ukdf_user = pd.merge(ukdf_user, uk_frequency, on='CustomerID')
ukdf_user.head()

# Plotting Recency histogram
fig = px.histogram(ukdf_user, x="Frequency",
                 labels={
                     "Frequency": "Frequency of purchase",
                     },
                title="Frequency")
fig.show()

# The maximum frequencies are below 1000. Visualizing frequencies below 1000.
df2=ukdf_user.query('Frequency < 1000')['Frequency']
fig = px.histogram(df2, x="Frequency",
                 labels={
                     "Frequency": "Frequency of purchase",
                     },
                title="Frequency")
fig.show()

# Frequency clusters
# Determining the right number of clusters for K-Means by elbow method
from sklearn.cluster import KMeans
sse={} # error
uk_recency = ukdf_user[['Frequency']]
for k in range(1, 10):
    kmeans = KMeans(n_clusters=k, max_iter=1000).fit(uk_recency)
    uk_recency["clusters"] = kmeans.labels_  #cluster names corresponding to
recency values
    sse[k] = kmeans.inertia_ #sse corresponding to clusters
plt.figure()
plt.plot(list(sse.keys()), list(sse.values()))
plt.xlabel("Number of cluster")
plt.show()

# Applying k-Means
kmeans=KMeans(n_clusters=4)
ukdf_user['FrequencyCluster']=kmeans.fit_predict(ukdf_user[['Frequency']])

#ordering the frequency cluster
ukdf_user = order_cluster('FrequencyCluster', 'Frequency', ukdf_user, True )
ukdf_user.groupby('FrequencyCluster')['Frequency'].describe()

Monetary Value
# We will now cluster our customers based on revenue.
#calculate revenue for each customer
uk_revenue = ukdf.groupby('CustomerID').Revenue.sum().reset_index()
uk_revenue.head()

# merging uk_revenue with our main dataframe
ukdf_user = pd.merge(ukdf_user, uk_revenue, on='CustomerID')
ukdf_user.head(3)

# Visualizing with the histogram on a reduced scale
#plot the histogram
df4=ukdf_user.query('Revenue < 10000')['Revenue']
fig = px.histogram(df4, x="Revenue",
                 labels={
                     "Revenue": "Monetary value of purchase",
                     },
                title="Revenue")
fig.show()

# Using elbow method to find out the optimum number of clusters for K-Means
from sklearn.cluster import KMeans
sse={} # error
uk_recency = ukdf_user[['Revenue']]
for k in range(1, 10):
    kmeans = KMeans(n_clusters=k, max_iter=1000).fit(uk_recency)
    uk_recency["clusters"] = kmeans.labels_  # cluster names corresponding to
recency values
    sse[k] = kmeans.inertia_ # sse corresponding to clusters
plt.figure()
plt.plot(list(sse.keys()), list(sse.values()))
plt.xlabel("Number of cluster")
plt.show()

# Elbow method suggests optimal clusters can be 3 or 4. We will take 4 as the
number of clusters

#apply clustering
kmeans = KMeans(n_clusters=4)
ukdf_user['RevenueCluster'] = kmeans.fit_predict(ukdf_user[['Revenue']])

#order the cluster numbers
ukdf_user = order_cluster('RevenueCluster', 'Revenue',ukdf_user,True)

#show details of the dataframe
ukdf_user.groupby('RevenueCluster')['Revenue'].describe()
```

```
# Overall scores

# Now we have scores (cluster numbers) for recency, frequency & revenue. We
will now create an overall score.
# Calculating overall score and use mean() to see details
ukdf_user['OverallScore'] = ukdf_user['RecencyCluster'] +
ukdf_user['FrequencyCluster'] + ukdf_user['RevenueCluster']
ukdf_user.groupby('OverallScore')['Recency','Frequency','Revenue'].mean()

# It seems that customer with score 7 is our best customer and one with score
0 is least attractive customer.
# Analysing low value and high value customers
ukdf_user['Segment'] = 'Low-Value'
ukdf_user.loc[ukdf_user['OverallScore']>2,'Segment'] = 'Mid-Value'
ukdf_user.loc[ukdf_user['OverallScore']>4,'Segment'] = 'High-Value'
ukdf_user.head()

Visualizing segments with scatter plots
#Revenue vs Frequency
uk_graph = ukdf_user.query("Revenue < 10000 and Frequency < 1000")

plot_data = [
    go.Scatter(
        x=uk_graph.query("Segment == 'Low-Value'")['Frequency'],
        y=uk_graph.query("Segment == 'Low-Value'")['Revenue'],
        mode='markers',
        name='Low',
        marker= dict(size= 7,
            line= dict(width=1),
            color= 'blue',
            opacity= 0.8
           )
    ),
        go.Scatter(
        x=uk_graph.query("Segment == 'Mid-Value'")['Frequency'],
        y=uk_graph.query("Segment == 'Mid-Value'")['Revenue'],
        mode='markers',
        name='Mid',
        marker= dict(size= 9,
            line= dict(width=1),
            color= 'green',
            opacity= 0.5
           )
    ),
        go.Scatter(
        x=uk_graph.query("Segment == 'High-Value'")['Frequency'],
        y=uk_graph.query("Segment == 'High-Value'")['Revenue'],
        mode='markers',
        name='High',
        marker= dict(size= 11,
            line= dict(width=1),
            color= 'red',
            opacity= 0.9
           )
    ),
]
```

```
plot_layout = go.Layout(
        yaxis= {'title': "Revenue"},
        xaxis= {'title': "Frequency"},
        title='Segments'
    )
fig = go.Figure(data=plot_data, layout=plot_layout)
pyoff.iplot(fig)

#Revenue vs Recency

uk_graph = ukdf_user.query("Revenue < 10000 and Frequency < 1000")

plot_data = [
    go.Scatter(
        x=uk_graph.query("Segment == 'Low-Value'")['Recency'],
        y=uk_graph.query("Segment == 'Low-Value'")['Revenue'],
        mode='markers',
        name='Low',
        marker= dict(size= 7,
            line= dict(width=1),
            color= 'blue',
            opacity= 0.8
           )
    ),
        go.Scatter(
        x=uk_graph.query("Segment == 'Mid-Value'")['Recency'],
        y=uk_graph.query("Segment == 'Mid-Value'")['Revenue'],
        mode='markers',
        name='Mid',
        marker= dict(size= 9,
            line= dict(width=1),
            color= 'green',
            opacity= 0.5
           )
    ),
        go.Scatter(
        x=uk_graph.query("Segment == 'High-Value'")['Recency'],
        y=uk_graph.query("Segment == 'High-Value'")['Revenue'],
        mode='markers',
        name='High',
        marker= dict(size= 11,
            line= dict(width=1),
            color= 'red',
            opacity= 0.9
           )
    ),
]

plot_layout = go.Layout(
        yaxis= {'title': "Revenue"},
        xaxis= {'title': "Recency"},
        title='Segments'
    )
fig = go.Figure(data=plot_data, layout=plot_layout)
fig.show(renderer="notebook")
```

```
# Recency vs Frequency Visualization

uk_graph = ukdf_user.query("Revenue < 50000 and Frequency < 2000")

plot_data = [
    go.Scatter(
        x=uk_graph.query("Segment == 'Low-Value'")['Recency'],
        y=uk_graph.query("Segment == 'Low-Value'")['Frequency'],
        mode='markers',
        name='Low',
        marker= dict(size= 7,
            line= dict(width=1),
            color= 'blue',
            opacity= 0.8
           )
    ),
        go.Scatter(
        x=uk_graph.query("Segment == 'Mid-Value'")['Recency'],
        y=uk_graph.query("Segment == 'Mid-Value'")['Frequency'],
        mode='markers',
        name='Mid',
        marker= dict(size= 9,
            line= dict(width=1),
            color= 'green',
            opacity= 0.5
           )
    ),
        go.Scatter(
        x=uk_graph.query("Segment == 'High-Value'")['Recency'],
        y=uk_graph.query("Segment == 'High-Value'")['Frequency'],
        mode='markers',
        name='High',
        marker= dict(size= 11,
            line= dict(width=1),
            color= 'red',
            opacity= 0.9
           )
    ),
]

plot_layout = go.Layout(
        yaxis= {'title': "Frequency"},
        xaxis= {'title': "Recency"},
        title='Segments'
    )
fig = go.Figure(data=plot_data, layout=plot_layout)
fig.show(renderer="notebook")
```

*** End of the Project***