**Introduction**

In today's data-driven world, businesses are constantly seeking innovative ways to optimize product offerings and enhance customer segmentation in the retail industry. This pressing challenge can be effectively addressed through the utilization of big data analytics. To tackle this problem head-on, we have leveraged the UK’s retailer’s transactional dataset, a rich and real-world collection of transactional data obtained from an online retail store. This dataset serves as a valuable source of insights into customer behavior, purchasing patterns, and detailed product information, enabling us to gain a comprehensive understanding of the retail landscape.

Our analysis encompasses two pivotal techniques: product clustering using Yules coefficient and RFM customer segmentation. The utilization of Yules coefficient facilitates the identification of associations between diverse products based on their co-occurrence in customer transactions. This enables us to group together similar products and uncover valuable patterns, thereby aiding inventory management and revealing lucrative cross-selling opportunities.

Moreover, we have employed RFM customer segmentation, a powerful methodology that evaluates customers based on three essential dimensions: recency, frequency, and monetary value of purchases. By categorizing customers into distinct groups through this segmentation technique, we can obtain profound insights into their purchasing behavior, allowing for the development of tailored and personalized marketing strategies.

By applying these cutting-edge methodologies to the leveraged dataset, our primary objective is to extract actionable insights that will empower retailer to optimize their product assortments, enhance customer satisfaction, and elevate overall business performance in the highly competitive retail industry. Through this comprehensive analysis, we aim to equip business with the tools and knowledge needed to thrive in an ever-evolving marketplace.

**Initial Analysis**

1. A notable count of approximately 1.4k records lacks a comprehensive description of the corresponding stockkeeping unit. We can curtail the occurrence of such missing descriptions by exploring the mapping between SKUcodes and descriptions derived from alternative records.
2. Invoices distinguished by the prefix 'C' signify canceled orders, delineating a distinct subset within the dataset.
3. Our dataset encompasses an array of entries encompassing manual adjustments, damages, postage charges, dotcom charges, and Amazon fees. Notably, their SKUcodes do not commence with a digit, necessitating their exclusion from the product clustering analysis.
4. Remarkably, the top 10 SKUs collectively contribute to approximately 10% of the overall sales, revealing their significant influence on the revenue landscape. It is worth highlighting that both Dotcom Postage and Postage hold a prominent position within this esteemed roster.
5. The United Kingdom emerges as a dominant force, accounting for a striking 84% of the total sales. This underscores the imperative to accord the utmost priority to customers hailing from the UK, as they constitute the majority within the customer base.
6. September 2011 witnessed an impressive 40% surge in sales, more remarkable considering the absence of any newly added SKUs during that period. This notable growth could be attributed to strategic collaborations forged with additional wholesalers, thereby fortifying the revenue streams.
7. An intriguing observation unveils that the top 20% of SKUs epitomize a staggering 80% of the total sales, exemplifying the Pareto principle in the dynamic retail realm. This realization unveils an opportunity to eliminate underperforming SKUs, streamlining the product portfolio for enhanced profitability.

Notably, SKU number 23084 emerges as a notable presence within both the most frequently purchased units as well as the esteemed roster of revenue-generating SKUs, attesting to its prominence and commercial significance within the dataset.

**Analysis**

In this section, we will cover different aspects of data analysis such as data cleaning, data preparation, models’ building, and models’ evaluation.

Here is an outline of the steps performed:

* **Data Cleaning:**
  + Removed canceled orders and adjusted transactions from the dataset.
  + Filtered out irrelevant columns and retained the necessary ones for analysis.
* **Yules Coefficient and Product Clustering:**
  + Prepared the data by creating product combinations that appeared in the dataset.
  + Applied product clustering using both k-means and hierarchical clustering techniques.
  + Examined the results to identify clusters and hierarchical relationships between products.
* **RFM Customer Segmentation:**
  + Conducted customer segmentation based on three dimensions: recency, frequency, and monetary value.
  + Calculated recency by determining the latest day a customer visited the retailer.
  + Calculated frequency by counting the number of visits made by each customer.
  + Calculated monetary value by aggregating the total transaction amount for each customer.
* **K-means and DBSCAN Clustering for Customer Segmentation:**
  + Calculated the recency, frequency and monetary values of customers and treated it as input for the k-means algorithm.
  + Applied k-means and DBSCAN clustering to group customers into distinct clusters based on their similarity.
  + Analyzed the results to understand different customer segments and their characteristics.

Through this analysis, we learned the following insights beyond the basic analysis:

* **Hidden Patterns:** Despite the initial appearance of a relatively shallow dataset, our analysis using Yules coefficient and clustering techniques revealed hidden patterns and relationships among products. This demonstrates the importance of exploring deeper insights beyond surface-level observations.
* **Product Hierarchy:** The hierarchical clustering approach allowed us to create a hierarchy of products based on their relationships. This insight can be leveraged for inventory management, cross-selling strategies, and optimizing product assortments.
* **Customer Segmentation:** By employing RFM customer segmentation and k-means clustering, we successfully grouped customers into distinct clusters based on their recency, frequency, and monetary value. This enables targeted marketing strategies, personalized promotions, and tailored customer retention initiatives.

During our preliminary analysis, we discerned that the transactional data harbored pertinent information pertaining to adjustments made on the balance sheet. Amongst these adjustments were distinct components such as .com postage fee, amazon fee, and debts, which necessitated meticulous removal before proceeding with any transformative processes. Furthermore, we expunged the inclusion of gift cards, as their association with specific “StockCodes” was nonexistent. Subsequently, we embarked upon the computation of the Yules coefficient of association, a statistical measure that quantifies the degree of dependence between categorical variables, facilitating the subsequent implementation of diverse clustering methodologies.

Moreover, in our pursuit of customer segmentation, we initially opted to employ the k-means clustering algorithm. This choice was guided by the inherent characteristics of the dataset, wherein one of the categorical attributes, namely "Country," displayed a diminished variability, predominantly comprised of customers originating from the United Kingdom. Consequently, the utilization of k-means clustering as an initial approach would facilitate the formation of distinct customer segments, enabling enhanced targeting and tailored marketing strategies.

**Data Cleaning**

After initializing Pyspark, we performed data cleaning using PySpark to remove canceled orders and adjusted transactions. This step helps in ensuring the accuracy and integrity of the data, providing a reliable foundation for further analysis. You can refer figure 1 for seeing the data cleaning.

A screenshot of a computer program

Description automatically generated with medium confidence

Figure : Data Cleaning code

**Product Clustering**

Figure 2 presents an insightful overview of the Yules coefficient association technique employed for product clustering. It offers a visual representation of the intricate interplay between two products, highlighting their associations through the analysis of transactional combinations. By delving into this illustration, we gain a deeper understanding of the underlying relationships and connections between various products within the dataset. Check figure 2 and 3.

A picture containing text, screenshot, line, diagram

Description automatically generated

Figure : Yules coefficient demonstration

A picture containing text, font, line, white

Description automatically generated

Figure : Yules coefficient formula

The underlying concept behind the utilization of the Yules coefficient was to discern and aggregate products based on their inherent similarity. To achieve this objective, we computed a square matrix encompassing the Yules coefficient for every conceivable pair of Stock codes within the dataset. Subsequently, we performed the k-means algorithm iteratively, employing varying values of k, with the aim of identifying the optimal number of product clusters. The outcomes derived from the k-means clustering were subject to analysis through two distinct methodologies: silhouette analysis and the elbow curve method. These techniques provided valuable insights into the effectiveness and cohesion of the clustering results. Refer figure 4.

A graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of

Description automatically generated with low confidence

Figure : Silhouette and Elbow curve analysis for product clustering

After performing the methods mentioned above, we inferred that there is a discrepancy between the optimal value of k suggested by the elbow method and the decreasing silhouette scores with an increasing k value. Such discrepancies can make it challenging to determine the appropriate number of clusters. In such cases, it is important to consider additional factors and interpret the results cautiously. Here are a few possible interpretations:

* **Complex Data Structure:** The dataset may have a complex structure that does not conform well to the assumptions of the k-means algorithm. The decreasing silhouette scores could indicate that the data points are not well-separated into distinct clusters or that the clusters become more overlapping as the number of clusters increases. In such cases, k-means might not be the most suitable algorithm, and alternative clustering methods or dimensionality reduction techniques could be explored.
* **Overfitting with Elbow Method:** The elbow method examines the rate of decrease in inertia as the number of clusters increases. It is possible that the elbow method suggests a higher k value (e.g., 5) due to the decrease in inertia, but the silhouette scores indicate that the resulting clusters are not well-separated. This could be an indication of overfitting, where the clusters become too specific and do not capture the underlying patterns or true structure of the data.

Keeping in mind the above reasons we chose Hierarchical clustering.

**Customer Segmentation**

**A picture containing footwear, person, cartoon

Description automatically generated**

Figure : Customer Segmentation

Figure 5 provides a comprehensive depiction of the customer segmentation analysis conducted on the dataset under investigation. The objective of this study was to ascertain an effective approach for segmenting customers, thereby enabling the retailer to devise tailored marketing campaigns and strategies tailored to their distinct target audiences. By delving into this analysis, we aimed to deliver profound insights into the underlying dynamics of the customer base, facilitating the identification of specific customer segments characterized by unique needs and behaviors. Armed with these data-driven findings, the retailer can make informed decisions to amplify customer satisfaction, bolster customer retention, and elevate the overall performance of the business.

To conduct customer segmentation, we initially determined the recency values, which indicate the number of days since customers' most recent transactions with the retailer. Additionally, we computed the frequency column, representing the count of purchases made by each customer, and the monetary value, denoting the total amount of money spent by customers at the store. Subsequently, in the first round of the k-means algorithm, we used standardized (normalized) values of the actual continuous attributes - recency, frequency, and monetary. We iteratively used the k-means algorithm by varying the k value parameter from 2 to 11 and plotted the silhouette score and inertia to identify the optimal value of k. Figure 6 shows the results of the output.

A picture containing line, plot, screenshot

Description automatically generated

Figure : Optimal clusters choosing output.

From the figure 6, we observed that the optimum value of k is 2, if the clustering is performed using normalized actual attributes. Upon further examination of data by splitting them into two clusters, we noticed that the presence of outliers was heavily influencing the creation of clusters. All the outliers were getting clubbed into one cluster while the algorithm was failing to identify the patterns amongst other data points. Figure 7 are the cluster sizes when k was chosen to be 2.

A screen shot of a computer

Description automatically generated with low confidence

Figure : Clustering result with standardized data as input

Hence, to mitigate the impact of outliers, we created pentiles of RFM and performed clustering on those pentiles rather than those actual continuous variables. The clustering outcomes were then subjected to analysis using two established methodologies: silhouette analysis and the elbow curve method. These techniques furnished valuable insights into the coherence and efficacy of the clustering results, aiding in the interpretation of the segmentation process. For a visual representation of this analysis, kindly refer to Figure 8.

A picture containing line, plot, slope, diagram

Description automatically generated

Figure : Silhouette and Elbow curve analysis for customer segmentation

From the figure 8, we inferred that the optimal number of clusters to be chose for clustering is 4.

In our exploration of DBSCAN, we conducted a thorough examination of various parameter combinations. Initially, when we set the minimum sample size to 100 and experimented with different epsilon values ranging from 0.01 to 0.1, the algorithm suggested a staggering 14 clusters, which seemed implausible given the dataset. However, upon increasing the minimum sample size to 300, the algorithm recommended a more reasonable 2 clusters. However, 3.7k out of 4.3k customers were not assigned to any cluster as these were considered as noisy points. Additionally, we observed that the silhouette score did not exhibit significant improvement with variations in the epsilon value. Conversely, in the case of k-means clustering, we witnessed an enhancement in the silhouette score with 4 clusters, prompting us to proceed with this configuration. Refer figure 9.

A picture containing text, screenshot, line, plot

Description automatically generated

Figure : DBSCAN output for Customer segmentation

To tackle the drawbacks of DBSCAN and k-means (with normalized actual data), we decided to form our marketing strategies for the four clusters that were identified using k-means algorithm based on pentiles of RFM metrics.

**Conclusion**

During the process of product clustering, we successfully identified five significant categories that emerged from the analysis:

* **The Eclectic Gift Assortment:** This category encompasses a diverse range of products tailored for various occasions such as birthdays, Valentine's Day, and more.
* **The Christmas and Vintage Collection:** This cluster specifically consists of Christmas-themed products, forming a small but distinct group.
* **The Vibrant and Whimsical Gift Collection:** This category comprises an array of fancy and exotic items, including unique pieces like Chinese dragon portraits and Cinderella key rings.
* **The Gift Wrap and Decor Collection:** Within this cluster, customers can find an assortment of home decor products, including wall clocks and pillow covers, offering an opportunity to enhance the ambiance of living spaces.
* **The Kitchen and Stationery Essentials with Sweet Delights:** This particular category is dedicated to kitchen-related products, such as scissors, knives, kitchen towels, alongside delightful sweet treats.

Now armed with these clearly defined clusters, the retailer can effectively categorize and optimize their product catalog on their website. Furthermore, by delving into the performance of each SKU within these clusters, the retailer can address assortment optimization concerns, identifying and potentially eliminating underperforming items. Refer figure 10 for the visual.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Level1 | Gifts (4045 SKUs) | | | | |
| Level2 | Merry Vintage Whimsy Collection (2618 SKUs) | | | Charming Treasures: Gift Wrap, Kitchen, Stationery & Sweet Delights Collection (1427 SKUs) | |
| Level3 | Festive Vintage Gift Melange (1504 SKUs) | | Vibrant and Whimsical Gift Collection (1114 SKUs) | Charming Treasures: Gift Wrap, Kitchen, Stationery & Sweet Delights Collection (1427 SKUs) | |
| Level4 | Festive Vintage Gift Melange (1504 SKUs) | | Vibrant and Whimsical Gift Collection (1144 SKUs) | Gift Wrap and Decor Collection (687 SKUs) | Kitchen and Stationery Essentials with Sweet Delights (740 SKUs) |
| Level5 | Eclectic Gift Assortment (1110 SKUs) | Christmas and Vintage Collection (394 SKUs) | Vibrant and Whimsical Gift Collection (1114 SKUs) | Gift Wrap and Decor Collection (687 SKUs) | Kitchen and Stationery Essentials with Sweet Delights (740 SKUs) |

Figure : Product hierarchy using clustering

During the comprehensive customer segmentation analysis, we employed meticulous techniques to identify and classify customers into four distinct clusters, each characterized by their unique traits and purchasing patterns:

* **Platinum Elite Customers:** This cluster consists of loyal, valuable customers who regularly engage with the online store. They epitomize brand loyalty, making frequent visits and high-value transactions, driving substantial sales. To enhance brand advocacy and loyalty, offer exclusive rewards, personalized recommendations, and VIP perks.
* **Occasional/Dormant Shoppers:** Contrasting the Platinum Elite Customers, this cluster comprises infrequent visitors who make minor purchases. Their sporadic engagement and limited spending minimally contribute to sales. To re-engage them, use targeted promotions, limited time offers, and personalized incentives to increase their visit and purchase frequency.
* **Moderate Engagers/New Customers:** This cluster includes new customers showing promising involvement and making noteworthy transactions. They offer growth potential and require targeted marketing. Nurture their interest with personalized onboarding, tailored recommendations, and promotions to accelerate their transition to loyal customers.
* **Dedicated Shoppers:** In this cluster, frequent visitors show consistent engagement. While their transactions may be smaller than Platinum Elite Customers, their presence generates steady sales. Maintain their loyalty with loyalty programs, discounts, and personalized communication to maximize lifetime value and encourage repeat purchases.

Platinum Elite Customers: Dominant segment (28% of customer base, 73% sales). Occasional/Dormant Shoppers: 30% customer base, 5% sales. Tailor initiatives, allocate resources for deepening engagement with Platinum Elite Customers and re-engaging Occasional/Dormant Shoppers. Figure 9 shows cluster differences in recency and frequency.

A picture containing text, screenshot, plot, diagram

Description automatically generated

Figure : Visual showing relationship between Recency and Frequency for all the clusters

**References:**

Datamink. (2022, March 30). Automatically generate a Customer Decision Tree for a category of products in Retail. *Medium*. <https://datamink.medium.com/generate-a-customer-decision-tree-for-a-category-of-products-in-retail-53ee11bdce49>

*Finding associations with Yule’s Q and hierarchical clustering in R*. (n.d.). Step by Step DSci. <https://www.stepbystepdatascience.com/product-associations-with-yules-q>

Mahesh Huddar. (2023, February 22). *How to Compute Silhouette Coefficient – K Means Clustering in Machine Learning by Mahesh Huddar* [Video]. YouTube. <https://www.youtube.com/watch?v=FGXkbawTHRQ>

Krish Naik. (2020, August 25). *Silhouette (clustering)- Validating Clustering Models- Unsupervised Machine Learning* [Video]. YouTube. <https://www.youtube.com/watch?v=DpRPd274-0E>

*Selecting the number of clusters with silhouette analysis on KMeans clustering*. (n.d.). Scikit-learn. <https://scikit-learn.org/stable/auto_examples/cluster/plot_kmeans_silhouette_analysis.html>

Admin. (2023). Pandas Convert Date (datetime) to String Format. *Spark by {Examples}*. <https://sparkbyexamples.com/pandas/pandas-convert-datetime-to-string-format/>

Smith, W. (2023). Set theory – Definition and Examples. *The Story of Mathematics*. <https://www.storyofmathematics.com/set-theory/>

**Appendix**

# -\*- coding: utf-8 -\*-

"""Group\_Project\_final\_27Jun23\_v3.ipynb

Automatically generated by Colaboratory.

Original file is located at

https://colab.research.google.com/drive/1Vs\_UdA40yNeZIUG1goG25llVbcSrRBHq

"""

!apt-get install openjdk-8-jdk-headless -qq > /dev/null

!wget -q https://dlcdn.apache.org/spark/spark-3.4.0/spark-3.4.0-bin-hadoop3.tgz

!tar xf spark-3.4.0-bin-hadoop3.tgz

import os

os.environ["JAVA\_HOME"] = "/usr/lib/jvm/java-8-openjdk-amd64"

os.environ["SPARK\_HOME"] = '/content/spark-3.4.0-bin-hadoop3'

# Install library for finding Spark

get\_ipython().system('pip install -q findspark')

# Import the libary

import findspark

# Initiate findspark

findspark.init()

# Check the location for Spark

findspark.find()

# Import SparkSession

from pyspark.sql import SparkSession

# Create a Spark Session

spark = SparkSession.builder.master("local[\*]").getOrCreate()

# Check Spark Session Information

spark

# Import a Spark function from library

from pyspark.sql.functions import col

from google.colab import files

upload = files.upload()

# Reading csv file

data = spark.read.csv("Online Retail.csv", inferSchema=True, header = True)

data = data.toPandas()

data.head()

len(data)

import pandas as pd

# Delete rows starting with "C"

data = data[~data['InvoiceNo'].str.startswith('C')]

# Delete rows starting with "A"

data = data[~data['InvoiceNo'].str.startswith('A')]

len(data)

# List of values to be removed

values\_to\_remove = ["AMAZONFEE", "B", "BANK CHARGES", "C2", "CRUK", "D", "DOT", "gift\_0001\_10",

"gift\_0001\_20", "gift\_0001\_30", "gift\_0001\_40", "gift\_0001\_50", "M", "PADS",

"POST", "S"]

# Remove rows matching the specified values

data = data[~data['StockCode'].isin(values\_to\_remove)]

import pandas as pd

# Create a new column as the key by combining 'InvoiceNo' and 'StockCode'

data['Key'] = data['InvoiceNo'].astype(str) + '\_' + data['StockCode'].astype(str)

# Remove duplicates based on the key

data = data.drop\_duplicates(subset='Key')

# Remove the 'Key' column

data = data.drop('Key', axis=1)

data.head()

len(data)

columns\_to\_drop = ['Description', 'Quantity', 'InvoiceDate', 'UnitPrice', 'CustomerID', 'Country']

deleted\_cols\_data = data

# Drop the specified columns

deleted\_cols\_data = deleted\_cols\_data.drop(columns\_to\_drop, axis=1)

deleted\_cols\_data.info()

deleted\_cols\_data[deleted\_cols\_data['StockCode']=='10002'].info()

# Perform the left join

df\_left\_join = pd.merge(deleted\_cols\_data, deleted\_cols\_data, on='InvoiceNo', how='left', suffixes=('', '\_y'))

#df\_left\_join = df\_left\_join[df\_left\_join['StockCode'] != df\_left\_join['StockCode\_y']]

df\_left\_join.head()

# df\_left\_join[df\_left\_join['InvoiceNo'] == '551429']['InvoiceNo'].nunique()

combinations = df\_left\_join.groupby(['StockCode', 'StockCode\_y']).agg(num\_invoices=('InvoiceNo', 'nunique')).reset\_index()

combinations.head()

df\_left\_join[df\_left\_join['StockCode']=='10002']['InvoiceNo'].unique()

# deleted\_cols\_data[deleted\_cols\_data['StockCode'] == '10002']

combinations\_2 = df\_left\_join.groupby(['StockCode']).agg(num\_invoices=('InvoiceNo', 'nunique')).reset\_index()

combinations\_2.head()

merged\_combinations = combinations.merge(combinations\_2, on='StockCode', how='left')

merged\_combinations.head()

combinations\_2.rename(columns={'StockCode': 'StockCode\_y'}, inplace=True)

merged\_combinations\_2 = merged\_combinations.merge(combinations\_2, on = "StockCode\_y", how='left')

merged\_combinations\_2.head()

merged\_combinations\_2['total\_transactions\_xy'] = merged\_combinations\_2['num\_invoices\_y'] + merged\_combinations\_2['num\_invoices'] - merged\_combinations\_2['num\_invoices\_x']

merged\_combinations\_2.head()

merged\_combinations\_2['num\_invoices\_y'] = merged\_combinations\_2['num\_invoices\_y'] - merged\_combinations\_2['num\_invoices\_x']

merged\_combinations\_2['num\_invoices'] = merged\_combinations\_2['num\_invoices'] - merged\_combinations\_2['num\_invoices\_x']

merged\_combinations\_2.head()

txns = deleted\_cols\_data['InvoiceNo'].nunique()

txns

merged\_combinations\_2['total\_transactions\_xy'] = txns- merged\_combinations\_2['total\_transactions\_xy']

merged\_combinations\_2.head()

merged\_combinations\_2.rename(columns={'num\_invoices\_x': 'A', 'num\_invoices\_y': 'B', 'num\_invoices': 'C', 'total\_transactions\_xy': 'D'}, inplace=True)

# merged\_combinations\_2.head(30)

# Define the Yule's coefficient function

def yules\_coefficient(a, b, c, d):

numerator = (a \* d) - (b \* c)

denominator = (a \* d) + (b \* c)

if denominator != 0:

return numerator / denominator

else:

return 0

a = merged\_combinations\_2['A']\*merged\_combinations\_2['D']

b = merged\_combinations\_2['B']\*merged\_combinations\_2['C']

merged\_combinations\_2['YuleCoefficient'] = (a - b) / (a + b)

merged\_combinations\_2.head()

clustering\_dataset\_stage\_1 = merged\_combinations\_2[['StockCode', 'StockCode\_y', 'YuleCoefficient']]

clustering\_dataset\_stage\_1.head()

clustering\_dataset\_stage\_1['YuleCoefficient'].max()

clustering\_dataset\_stage\_1['YuleCoefficient'].min()

## Rescaling Yule's Coefficient

## Yule's Q is rescaled to be between 0-2 instead with 0 becoming the perfect association and 2 being a negative association.

# The reason for doing this is that clustering algorithms work off a notion of 'distance' between elements and won't work with negative values for distance.

clustering\_dataset\_stage\_1['YuleCoefficient'] = 1- clustering\_dataset\_stage\_1['YuleCoefficient']

clustering\_dataset\_stage\_1['YuleCoefficient'].min()

clustering\_dataset\_stage\_1['YuleCoefficient'].max()

# Transpose the DataFrame by 'StockCode\_y'

transposed\_df = clustering\_dataset\_stage\_1.pivot(index='StockCode', columns='StockCode\_y', values='YuleCoefficient')

transposed\_df.head()

transposed\_df = transposed\_df.fillna(2)

## 2 indicates negative association

transposed\_df.head()

transposed\_df = transposed\_df.rename(columns=lambda x: x.strip(), index=lambda x: x.strip())

transposed\_df = transposed\_df.rename\_axis(None, axis='columns')

transposed\_df.head()

transposed\_df\_copy = transposed\_df

transposed\_df\_copy.reset\_index(inplace = True)

transposed\_df\_copy.head()

from sklearn.cluster import AgglomerativeClustering

cluster\_df = pd.DataFrame()

for i in range(2, 11):

# Perform hierarchical clustering

cluster = AgglomerativeClustering(n\_clusters=i, affinity='euclidean', linkage='ward')

cluster\_labels = cluster.fit\_predict(transposed\_df.iloc[:, 1:])

# Add the cluster labels to the DataFrame

cluster\_df[f'cluster\_{i}'] = cluster\_labels + 1

cluster\_df = cluster\_df.add\_prefix('cluster\_')

cluster\_df.head()

from scipy.cluster.hierarchy import dendrogram, linkage

import matplotlib.pyplot as plt

# Perform hierarchical clustering using ward linkage

Z = linkage(transposed\_df.loc[:, ~transposed\_df.columns.isin(['StockCode'])], 'ward','euclidean')

# Plot the dendrogram

plt.figure(figsize=(12, 12))

plt.title('Hierarchical Clustering Dendrogram')

plt.xlabel('StockCode')

plt.ylabel('')

dendrogram(Z, labels=[''] \* len(transposed\_df.loc[:, ~transposed\_df.columns.isin(['StockCode'])]))

plt.tight\_layout()

plt.show()

# Concatenate horizontally (merge columns)

result = pd.concat([transposed\_df\_copy['StockCode'], cluster\_df], axis=1)

result.head()

result.to\_excel('Hierarchical\_CLustering\_Output.xlsx', index=False)

### In the next steps, we will run K-means clustering and identify optimum K using Silhouette coefficient

## Subsequently, the results from k-means will be compared against hierarchichal clustering output.

# Perform silhouette analysis for different k-values

from sklearn.cluster import KMeans

from sklearn.metrics import silhouette\_score

k\_values = range(2, 11)

silhouette\_scores = []

inertias = []

for k in k\_values:

# Perform k-means clustering

kmeans = KMeans(n\_clusters=k, random\_state=0)

labels = kmeans.fit\_predict(transposed\_df.iloc[:, 1:])

# Calculate silhouette score

silhouette\_avg = silhouette\_score(transposed\_df.iloc[:, 1:], labels)

silhouette\_scores.append(silhouette\_avg)

# Calculate inertia (within-cluster sum of squares)

inertia = kmeans.inertia\_

inertias.append(inertia)

# Plot the silhouette scores and elbow curve

fig, ax = plt.subplots(nrows=1, ncols=2, figsize=(12, 4))

# Plot silhouette scores

ax[0].plot(k\_values, silhouette\_scores)

ax[0].set\_xlabel('Number of Clusters (k)')

ax[0].set\_ylabel('Silhouette Score')

ax[0].set\_title('Silhouette Analysis')

# Plot elbow curve

ax[1].plot(k\_values, inertias)

ax[1].set\_xlabel('Number of Clusters (k)')

ax[1].set\_ylabel('Inertia')

ax[1].set\_title('Elbow Curve')

plt.tight\_layout()

plt.show()

silhouette\_scores

# When there is a discrepancy between the optimal value of k suggested by the elbow method and the decreasing silhouette scores with an increasing k value, it can be challenging to determine the appropriate number of clusters. In such cases, it is important to consider additional factors and interpret the results cautiously. Here are a few possible interpretations:

#

# 1. Complex Data Structure: The dataset may have a complex structure that does not conform well to the assumptions of the k-means algorithm. The decreasing silhouette scores could indicate that the data points are not well-separated into distinct clusters or that the clusters become more overlapping as the number of clusters increases. In such cases, k-means might not be the most suitable algorithm, and alternative clustering methods or dimensionality reduction techniques could be explored.

#

# 2. Overfitting with Elbow Method: The elbow method examines the rate of decrease in inertia as the number of clusters increases. It is possible that the elbow method suggests a higher k value (e.g., 5) due to the decrease in inertia, but the silhouette scores indicate that the resulting clusters are not well-separated. This could be an indication of overfitting, where the clusters become too specific and do not capture the underlying patterns or true structure of the data.

#

# 3. Domain Knowledge and Interpretation: It is important to consider domain knowledge and interpretability of the clusters. The silhouette score is a measure of how well each sample fits its assigned cluster compared to other clusters. Even if the silhouette scores decrease, it does not necessarily mean that the clustering is meaningless or incorrect. It could be that the resulting clusters have meaningful interpretations in the context of the problem domain, despite the decrease in silhouette scores.

#

# 4. Other Evaluation Metrics: The silhouette score is one evaluation metric for clustering, but it is not the only metric available. Consider exploring other evaluation metrics, such as the Davies-Bouldin Index or the Calinski-Harabasz Index, to gain a more comprehensive understanding of the clustering performance and validate the optimal k-value.

#

# In summary, when there is a discrepancy between the elbow method and decreasing silhouette scores, it is crucial to consider the complexity of the data, assess the interpretability of the clusters, and explore alternative evaluation metrics. Additionally, it may be worth experimenting with different clustering algorithms or techniques to identify the most appropriate way to cluster the data.

# In[11]:

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

sns.set()

import datetime as dt

import warnings

warnings.filterwarnings('ignore')

# https://medium.com/@ugursavci/customer-segmentation-using-rfm-analysis-in-python-218a3255f714

df = spark.read.csv("Online Retail.csv", inferSchema=True, header = True)

df = df.toPandas()

def summary(df):

display(df.head())

print('-'\*100)

display(df.info())

print('-'\*100)

display(df.describe([0.01,0.25,0.50,0.75,0.99]))

summary(df)

df['InvoiceDate1'] = pd.to\_datetime(df['InvoiceDate'], format='%d-%m-%Y %H:%M')

df['Date'] = df['InvoiceDate1'].dt.date

df['Date'].max()

df[df['InvoiceDate']=='2011-12-10 17:19:00']

df.info()

df = df[(df.Quantity>0) & (df.UnitPrice> 0)]

df = df[~df['InvoiceNo'].str.startswith('C')]

df = df.drop\_duplicates()

df.isnull().sum()

df.dropna(inplace=True)

df.info()

df['Total\_Price'] = df['UnitPrice']\*df['Quantity']

df['Date'].max()

now = dt.date(2011,12,10)

now

rfm = df.groupby('CustomerID').agg({'Date' : lambda day : (now - day.max()).days,

'InvoiceNo': lambda x: len(pd.unique(x)),

'Total\_Price': lambda price : price.sum()})

col\_list = ['Recency','Frequency','Monetary']

rfm.columns = col\_list

rfm.head()

rfm['Frequency'].nunique()

rfm.info()

## Trying K-means using raw continuous variables after normalizing

## if outliers influence the clusters then we will use pentiles rather than normalized form of continuous variables

from sklearn.cluster import AgglomerativeClustering

from sklearn.preprocessing import StandardScaler

# Select the relevant columns for clustering

columns\_for\_clustering = ['Recency','Frequency','Monetary']

# Perform feature scaling on the selected columns

scaler = StandardScaler()

scaled\_data = scaler.fit\_transform(rfm[columns\_for\_clustering])

scaled\_data

from sklearn.cluster import KMeans

from sklearn.metrics import silhouette\_score

k\_values = range(2, 11)

silhouette\_scores = []

inertias = []

for k in k\_values:

# Perform k-means clustering

kmeans = KMeans(n\_clusters=k, random\_state=0)

labels = kmeans.fit\_predict(scaled\_data)

# Calculate silhouette score

silhouette\_avg = silhouette\_score(scaled\_data, labels)

silhouette\_scores.append(silhouette\_avg)

# Calculate inertia (within-cluster sum of squares)

inertia = kmeans.inertia\_

inertias.append(inertia)

# Plot the silhouette scores and elbow curve

fig, ax = plt.subplots(nrows=1, ncols=2, figsize=(12, 4))

# Plot silhouette scores

ax[0].plot(k\_values, silhouette\_scores)

ax[0].set\_xlabel('Number of Clusters (k)')

ax[0].set\_ylabel('Silhouette Score')

ax[0].set\_title('Silhouette Analysis')

# Plot elbow curve

ax[1].plot(k\_values, inertias)

ax[1].set\_xlabel('Number of Clusters (k)')

ax[1].set\_ylabel('Inertia')

ax[1].set\_title('Elbow Curve')

plt.tight\_layout()

plt.show()

## The silhouette analysis shows that 2 is the ideal no. of clusters whereas k of 3 seems ideal from elbow curve.

## Let us first examine the clustering with K= 2 and then with K= 3

kmeans = KMeans(n\_clusters=2, random\_state=0)

labels = kmeans.fit\_predict(scaled\_data)

rfm['labels']=labels

rfm.head()

rfm.reset\_index(inplace = True)

rfm.info()

result = rfm.groupby('labels').size().reset\_index(name='count')

print(result)

## Here we can see that the using normalized continuous variables for clustering is not yielding proper results with k= 2

## as most of the extreme points are getting clubbed into one single cluster and the alogorithm is failing to identify patterns amongst remaining points.

## Let us try with K=3 and then decide the next steps

kmeans = KMeans(n\_clusters=3, random\_state=0)

labels = kmeans.fit\_predict(scaled\_data)

rfm['labels']=labels

rfm.head()

rfm.info()

result = rfm.groupby('labels').size().reset\_index(name='count')

print(result)

## The outliers are still influencing the algorithm. Hence, instead of using actual continuous variables (or their normalized version), we will use corresponding

## pentiles.

rfm["R"] = pd.qcut(rfm["Recency"],5,labels=[5,4,3,2,1])

rfm["F"] = pd.qcut(rfm["Frequency"],5,labels=[1,2,3,4], duplicates='drop')

rfm["M"] = pd.qcut(rfm["Monetary"],5,labels=[1,2,3,4,5])

rfm["RFM\_Score"] = rfm["R"].astype(str) +rfm["F"].astype(str) + rfm["M"].astype(str)

rfm['F'].unique()

rfm\_clustering = rfm[['R','F','M']]

rfm\_clustering

from sklearn.cluster import KMeans

from sklearn.metrics import silhouette\_score

k\_values = range(2, 11)

silhouette\_scores = []

inertias = []

for k in k\_values:

# Perform k-means clustering

kmeans = KMeans(n\_clusters=k, random\_state=0)

labels = kmeans.fit\_predict(rfm\_clustering)

# Calculate silhouette score

silhouette\_avg = silhouette\_score(rfm\_clustering, labels)

silhouette\_scores.append(silhouette\_avg)

# Calculate inertia (within-cluster sum of squares)

inertia = kmeans.inertia\_

inertias.append(inertia)

# Plot the silhouette scores and elbow curve

fig, ax = plt.subplots(nrows=1, ncols=2, figsize=(12, 4))

# Plot silhouette scores

ax[0].plot(k\_values, silhouette\_scores)

ax[0].set\_xlabel('Number of Clusters (k)')

ax[0].set\_ylabel('Silhouette Score')

ax[0].set\_title('Silhouette Analysis')

# Plot elbow curve

ax[1].plot(k\_values, inertias)

ax[1].set\_xlabel('Number of Clusters (k)')

ax[1].set\_ylabel('Inertia')

ax[1].set\_title('Elbow Curve')

plt.tight\_layout()

plt.show()

kmeans = KMeans(n\_clusters=4, random\_state=0)

labels = kmeans.fit\_predict(rfm\_clustering)

rfm['Cluster'] = labels

rfm.to\_excel('Customer\_Segmentation\_K\_means.xlsx', index=False)

from sklearn.cluster import DBSCAN

from sklearn.metrics import silhouette\_score

# Fixed min\_samples value

min\_samples = 300

# List of epsilon values to try

eps\_values = [0.02, 0.03, 0.04, 0.05, 0.06,0.07,0.08,0.09,0.1]

# Perform DBSCAN clustering with different epsilon values

silhouette\_scores = []

for eps in eps\_values:

dbscan = DBSCAN(eps=eps, min\_samples=min\_samples)

labels = dbscan.fit\_predict(rfm\_clustering)

silhouette\_avg = silhouette\_score(rfm\_clustering, labels)

silhouette\_scores.append(silhouette\_avg)

print("Epsilon =", eps)

print("Cluster Labels:", labels)

print("Silhouette Score:", silhouette\_avg)

print("-------------------")

# Plotting the silhouette scores

plt.plot(eps\_values, silhouette\_scores, marker='o')

plt.xlabel('Epsilon (eps)')

plt.ylabel('Silhouette Score')

plt.title('Silhouette Analysis')

plt.show()

np.unique(labels)

## Testing the cluster size with epsilon= 0.02 and min\_samples=300

dbscan = DBSCAN(eps=0.02, min\_samples=300)

labels = dbscan.fit\_predict(rfm\_clustering)

rfm\_clustering

rfm\_clustering['Customer\_ID']=rfm['CustomerID']

rfm\_clustering['dbscan\_labels']=labels

result = rfm\_clustering.groupby('dbscan\_labels').size().reset\_index(name='count')

print(result)

seg\_map = {

r'[1-2][1-2]': 'Hibernating',

r'[1-2][3-4]': 'At Risk',

r'[1-2]5': 'Can\'t Loose',

r'3[1-2]': 'About to Sleep',

r'33': 'Need Attention',

r'[3-4][4-5]': 'Loyal Customers',

r'41': 'Promising',

r'51': 'New Customers',

r'[4-5][2-3]': 'Potential Loyalists',

r'5[4-5]': 'Champions'

}

df.head()

len(df)

rfm['Segment'] = rfm['R'].astype(str) + rfm['F'].astype(str)

rfm['Segment'] = rfm['Segment'].replace(seg\_map, regex=True)

rfm.head()

rfm.reset\_index(inplace=True)

rfm.head()

len(rfm)

rfm.groupby('Segment').mean().sort\_values('Monetary')

merged\_df = pd.merge(df, rfm, on='CustomerID', how='right')

merged\_df = merged\_df.drop\_duplicates(subset='CustomerID')

merged\_df.head()

get\_ipython().system('pip install kmodes')

from kmodes.kprototypes import KPrototypes

merged\_df.head()

# Select the columns for clustering (both numerical and categorical)

# Select the columns for clustering

from sklearn.preprocessing import OneHotEncoder

columns\_for\_clustering = ['Quantity', 'UnitPrice', 'Total\_Price', 'Recency', 'Frequency', 'Monetary', 'R', 'F', 'M']

# Perform one-hot encoding on categorical variables

categorical\_cols = ['Country', 'Segment']

enc = OneHotEncoder(sparse=False)

encoded\_categorical = enc.fit\_transform(merged\_df[categorical\_cols])

# Concatenate numerical and encoded categorical variables

X = np.concatenate((merged\_df[columns\_for\_clustering].values, encoded\_categorical), axis=1)

# Specify the number of clusters

n\_clusters = 3

# Run k-prototype algorithm

kproto = KPrototypes(n\_clusters=n\_clusters, init='Cao', random\_state=42)

clusters = kproto.fit\_predict(X, categorical=[9, 10]) # Index 9 corresponds to the 'Segment' column

# Add the cluster labels to the original dataframe

merged\_df['Cluster'] = clusters

merged\_df.head()

merged\_df.to\_csv('k-prototype\_output.csv', index = False)

from google.colab import files

files.download("k-prototype\_output.csv")

plt.scatter(merged\_df['Recency'], merged\_df['Frequency'], c=merged\_df['Monetary'], cmap='viridis')

plt.xlabel('Recency')

plt.ylabel('Frequency')

plt.title('RFM Scatter Plot')

plt.colorbar(label='Monetary')

plt.show()

rfm\_avg = merged\_df.groupby('Segment')['Recency', 'Frequency', 'Monetary'].mean()

sns.heatmap(rfm\_avg, annot=True, cmap='YlGnBu')

plt.title('RFM Heatmap')

plt.show()

plt.figure(figsize=(10, 6))

plt.subplot(2, 2, 1)

sns.histplot(merged\_df['Recency'], kde=True)

plt.title('Recency Distribution')

plt.subplot(2, 2, 2)

sns.histplot(merged\_df['Frequency'], kde=True)

plt.title('Frequency Distribution')

plt.subplot(2, 2, 3)

sns.histplot(merged\_df['Monetary'], kde=True)

plt.title('Monetary Distribution')

plt.tight\_layout()

plt.show()

plt.figure(figsize=(10, 6))

plt.subplot(2, 2, 1)

sns.boxplot(x=merged\_df['Segment'], y=merged\_df['Recency'])

plt.title('Recency Box Plot')

plt.subplot(2, 2, 2)

sns.boxplot(x=merged\_df['Segment'], y=merged\_df['Frequency'])

plt.title('Frequency Box Plot')

plt.subplot(2, 2, 3)

sns.boxplot(x=merged\_df['Segment'], y=merged\_df['Monetary'])

plt.title('Monetary Box Plot')

plt.tight\_layout()

plt.show()