Data Mining- Summative

Online Retail Sales Performance

Name: Azmah Theba
School: Udgam School For Children
Email: 2305402@udgamschool.com

Introduction:

In today's world, businesses rely heavily on data mining to uncover valuable insights from large datasets. By using techniques like hierarchical clustering and K-means, companies can organize their data into useful groups. This helps them spot trends and patterns, which in turn guide their decision-making process. These algorithms play a crucial role in breaking down complex data, allowing businesses to develop targeted strategies based on important connections they discover.

Problem Statement:

Understanding consumer behavior and market trends gets harder as the number of online retail transactions rises. Businesses need to understand the underlying patterns in these transactions to improve customer satisfaction, effectively target audiences, and optimize operations. However, the volume and complexity of transactional data make traditional analytical techniques ineffective. Therefore, to extract meaningful insights from this enormous dataset, sophisticated data mining techniques are necessary. Effectively putting these strategies into practice will make it difficult to find useful patterns and relationships in the data and help organizations make well-informed decisions.

Goal/Objective:

Our goals are to deconstruct the aforementioned data mining project, comprehend its process, and analyze its findings. In doing so, we hope to acquire a practical understanding of data mining methods and their application in real-world scenarios, specifically K-means and hierarchical clustering. Additionally, we seek to enhance our analytical skills and ability to communicate complex concepts in a simple and accessible manner.

Approach:

Step 1: Reading and Understanding Data

In Figure 1.1, the code starts by importing the libraries required for data visualization and manipulation. After that, it reads the dataset of online retail transactions and offers descriptions of its components.

```
[ ] # mounting the drive
    from google.colab import drive
    drive.mount('/content/drive')

    Mounted at /content/drive
```

```
[ ] # Path to the CSV file in your Google Drive
    file_path = '/content/drive/MyDrive/OnlineRetail.csv'
```

```
# import required libraries for dataframe and visualization

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import datetime as dt

# import required libraries for clustering
import sklearn
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score
from scipy.cluster.hierarchy import linkage
from scipy.cluster.hierarchy import dendrogram
from scipy.cluster.hierarchy import cut_tree
```

```
[ ] # Reading the data on which analysis needs to be done

    retail = pd.read_csv('/content/drive/MyDrive/OnlineRetail.csv', sep=",", encoding="ISO-8859-1", header=0)
    retail.head()
```

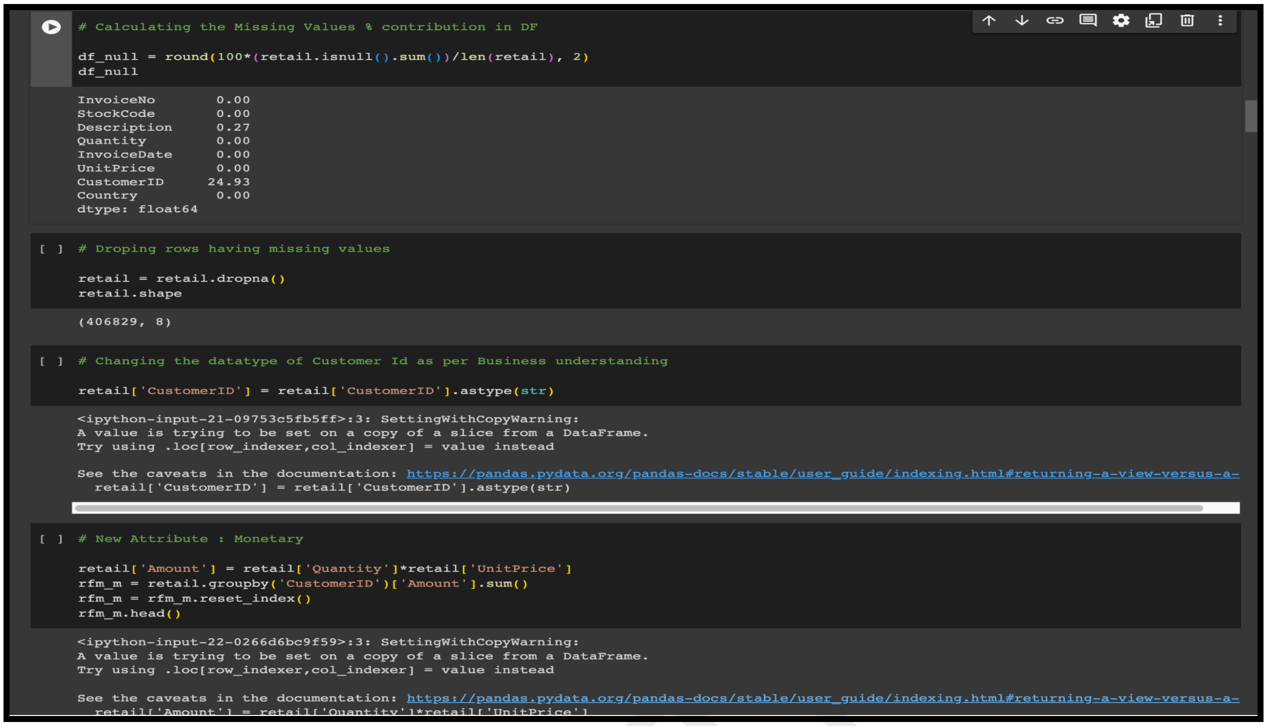
	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	01-12-2010 08:26	2.55	17850.0	United Kingdom
1	536365	71053	WHITE METAL LANTERN	6	01-12-2010 08:26	3.39	17850.0	United Kingdom
2	536365	84406B	CREAM CUPID HEARTS COAT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

```
[ ] retail.tail(10)
```

Figure 1.1

Step 2: Data Cleansing

In Figure 1.2, the code reads the dataset, determines the percentage of missing values, and then eliminates rows that have missing data. Furthermore, for consistency and additional analysis, it transforms the 'CustomerID' column into a string data type.



```
# Calculating the Missing Values % contribution in DF

df_null = round(100*(retail.isnull().sum())/len(retail), 2)
df_null

InvoiceNo       0.00
StockCode       0.00
Description     0.27
Quantity        0.00
InvoiceDate     0.00
UnitPrice       0.00
CustomerID     24.93
Country         0.00
dtype: float64
```

```
[ ] # Droping rows having missing values

retail = retail.dropna()
retail.shape

(406829, 8)
```

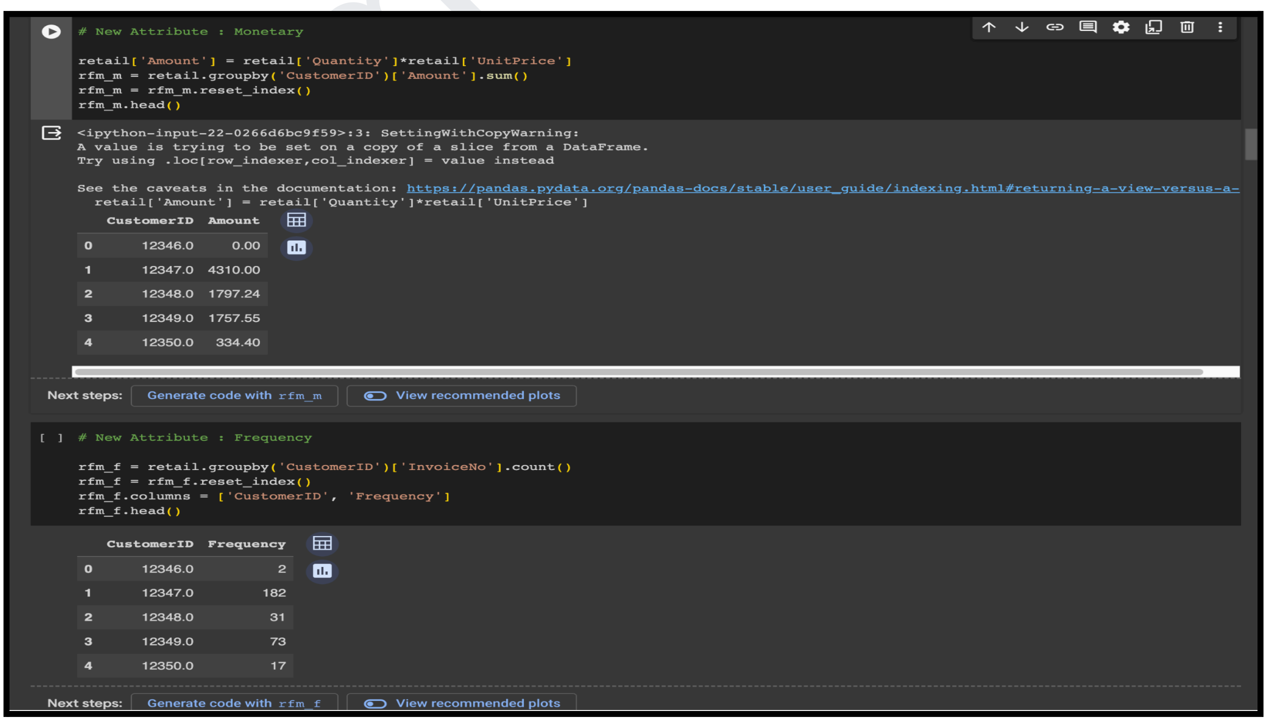
```
[ ] # Changing the datatype of Customer Id as per Business understanding

retail['CustomerID'] = retail['CustomerID'].astype(str)

<ipython-input-21-09753c5fb5ff>:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-
  retail['CustomerID'] = retail['CustomerID'].astype(str)
```

```
[ ] # New Attribute : Monetary

retail['Amount'] = retail['Quantity']*retail['UnitPrice']
rfm_m = retail.groupby('CustomerID')['Amount'].sum()
rfm_m = rfm_m.reset_index()
rfm_m.head()

<ipython-input-22-0266d6bc9f59>:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-
  retail['Amount'] = retail['Quantity']*retail['UnitPrice']
```

Figure 1.2

Step 3: Data Preparation

Figures 1.3 and 1.4 depict the creation of new attributes during data preparation, namely "Monetary," "Frequency," and "Recency," derived from customer transactions. Additionally, the code standardizes the data to ensure uniform scaling across all attributes.



```
# New Attribute : Monetary

retail['Amount'] = retail['Quantity']*retail['UnitPrice']
rfm_m = retail.groupby('CustomerID')['Amount'].sum()
rfm_m = rfm_m.reset_index()
rfm_m.head()

<ipython-input-22-0266d6bc9f59>:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-
  retail['Amount'] = retail['Quantity']*retail['UnitPrice']
```

	CustomerID	Amount
0	12346.0	0.00
1	12347.0	4310.00
2	12348.0	1797.24
3	12349.0	1757.55
4	12350.0	334.40

Next steps: Generate code with rfm_m View recommended plots

```
[ ] # New Attribute : Frequency

rfm_f = retail.groupby('CustomerID')['InvoiceNo'].count()
rfm_f = rfm_f.reset_index()
rfm_f.columns = ['CustomerID', 'Frequency']
rfm_f.head()
```

	CustomerID	Frequency
0	12346.0	2
1	12347.0	182
2	12348.0	31
3	12349.0	73
4	12350.0	17

Next steps: Generate code with rfm_f View recommended plots

Figure 1.3

```
# Merging the two dfs

rfm = pd.merge(rfm_m, rfm_f, on='CustomerID', how='inner')
rfm.head()
```

	CustomerID	Amount	Frequency
0	12346.0	0.00	2
1	12347.0	4310.00	182
2	12348.0	1797.24	31
3	12349.0	1757.55	73
4	12350.0	334.40	17

Next steps: Generate code with rfm View recommended plots

```
# New Attribute : Recency

# Convert to datetime to proper datatype

retail['InvoiceDate'] = pd.to_datetime(retail['InvoiceDate'],format='%d-%m-%Y %H:%M')
```

```
<ipython-input-25-6f1b941e83da>:5: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-
  retail['InvoiceDate'] = pd.to_datetime(retail['InvoiceDate'],format='%d-%m-%Y %H:%M')
```

```
# Compute the maximum date to know the last transaction date

max_date = max(retail['InvoiceDate'])
max_date
```

```
Timestamp('2011-12-09 12:50:00')
```

```
# Compute the difference between max date and transaction date

retail['Diff'] = max_date - retail['InvoiceDate']
retail.head(100)
```

InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country	Amount	Diff

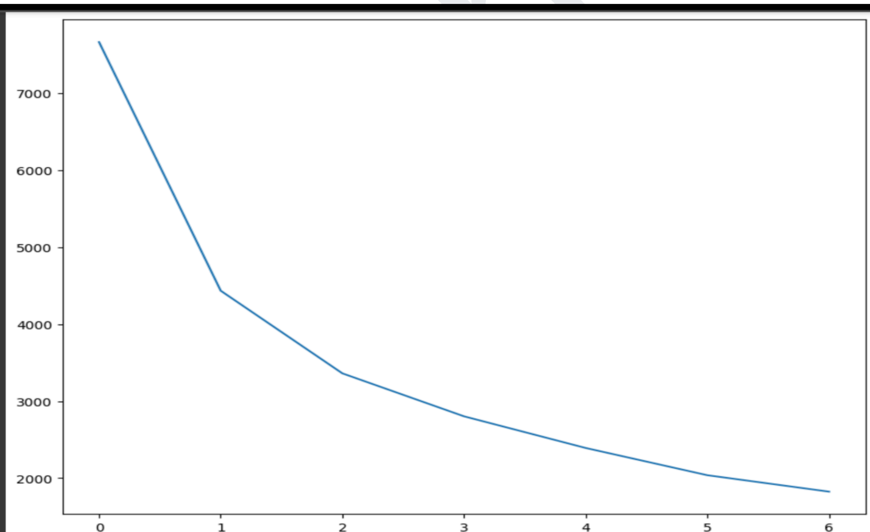
Figure 1.4

Step 4: Building the Model

Figures 1.5, 1.6, and 1.7 depict the application of K-means clustering for customer segmentation based on transactional patterns. Elbow Curve and Silhouette Analysis are employed to determine the optimal number of clusters, and cluster labels are assigned to customers.

```
[ ] # k-means with some arbitrary k

    kmeans = KMeans(n_clusters=4, max_iter=50)
    kmeans.fit(rfm_df_scaled)

    /usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change from
      warnings.warn(
    ▾          KMeans
    KMeans(max_iter=50, n_clusters=4)

[ ] kmeans.labels_

    array([3, 2, 0, ..., 3, 0, 0], dtype=int32)

    # Elbow-curve/SSD

    ssd = []
    range_n_clusters = [2, 3, 4, 5, 6, 7, 8]
    for num_clusters in range_n_clusters:
        kmeans = KMeans(n_clusters=num_clusters, max_iter=50)
        kmeans.fit(rfm_df_scaled)

        ssd.append(kmeans.inertia_)

    # plot the SSDs for each n_clusters
    plt.plot(ssd)

    /usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change from
      warnings.warn(
    /usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change from
      warnings.warn(
    /usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change from
      warnings.warn(
    /usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change from
      warnings.warn(
    /usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change from
      warnings.warn(
    /usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change from
      warnings.warn(
    /usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change from
      warnings.warn(
    [<matplotlib.lines.Line2D at 0x7e1adbef04c0>]
```

Figure 1.5




```
[ ] # Silhouette analysis
    range_n_clusters = [2, 3, 4, 5, 6, 7, 8]

    for num_clusters in range_n_clusters:

        # intialise kmeans
        kmeans = KMeans(n_clusters=num_clusters, max_iter=50)
        kmeans.fit(rfm_df_scaled)

        cluster_labels = kmeans.labels_
```

Figure 1.6

```python
# Silhouette analysis
range_n_clusters = [2, 3, 4, 5, 6, 7, 8]

for num_clusters in range_n_clusters:

    # intialise kmeans
    kmeans = KMeans(n_clusters=num_clusters, max_iter=50)
    kmeans.fit(rfm_df_scaled)

    cluster_labels = kmeans.labels_

    # silhouette score
    silhouette_avg = silhouette_score(rfm_df_scaled, cluster_labels)
    print("For n_clusters={0}, the silhouette score is {1}".format(num_clusters, silhouette_avg))
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change from
  warnings.warn(
For n_clusters=2, the silhouette score is 0.5411246404292333
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change from
  warnings.warn(
For n_clusters=3, the silhouette score is 0.5084896296141937
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change from
  warnings.warn(
For n_clusters=4, the silhouette score is 0.48132884121548086
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change from
  warnings.warn(
For n_clusters=5, the silhouette score is 0.46613075550600325
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change from
  warnings.warn(
For n_clusters=6, the silhouette score is 0.41712323985768335
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change from
  warnings.warn(
For n_clusters=7, the silhouette score is 0.41762636979384465
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change from
  warnings.warn(
For n_clusters=8, the silhouette score is 0.40192885592712724
```

```python
# Final model with k=3
kmeans = KMeans(n_clusters=3, max_iter=50)
kmeans.fit(rfm_df_scaled)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change from
  warnings.warn(
      KMeans
KMeans(max_iter=50, n_clusters=3)
```

Figure 1.7

Step 5: Final Analysis

In the K-Means clustering analysis with 3 cluster IDs, we conclude that customers with Cluster ID 2 have a higher number of transactions as compared to other customers. Customers with Cluster ID 2 are frequent buyers. Customers with Cluster ID 1 are not recent buyers and hence of the least importance from a business point of view.

Similarly, in the hierarchical clustering analysis with 3 cluster labels, customers with Cluster_Labels 1 are the ones with the highest number of transactions as compared to other customers. Customers with Cluster_Labels 1 are frequent buyers. Customers with Cluster_Labels 0 are not recent buyers and hence of the least importance from a business point of view.

Conclusion:

In summary, our analysis showed that by using K-Means and hierarchical clustering, we were able to group customers based on their shopping habits. We discovered different types of customers with unique buying patterns. We learned how clustering algorithms work and how to apply them to real-world data. With this knowledge, we can tackle similar tasks in data analysis and decision-making in the future.

References:

Hellbuoy. (2019, November 10). Online Retail K-Means & Hierarchical Clustering. https://www.kaggle.com/code/hellbuoy/online-retail-k-means-hierarchical-clustering/notebook#Step-4-:-Building-the-Model

Bothra, R. (n.d.). Home - Learn | HEvO. Learn | Hevo. https://hevodata.com/learn/

Das, V. K. (2020, October 8). K-Means Clustering vs Hierarchical Clustering. Global Tech Council. https://www.globaltechcouncil.org/clustering/k-means-clustering-vs-hierarchical-clustering/

Gustafsen, A. (2022, May 7). Hierarchical clustering and K-means clustering on country data. Medium. https://towardsdatascience.com/hierarchical-clustering-and-k-means-clustering-on-country-data-84b2bf54d282