National College of Ireland

Configuration Manual

MSc Research Project
MSc in Data Analytics

Nandheeswari Rajendran
Student ID: X22132210

School of Computing
National College of Ireland

Supervisor: Prof. Cristina Hava Muntean

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name: ……...Nandheeswari Rajendran…….

Student ID: ……X22132210…….

Programme: ……..MSc in Data Analytics…….. **Year:** 2023-2024

Module: ……MSc Research Project……

Supervisor: …….. Prof. Cristina Hava Muntean……

Submission Due Date: ……31/01/2024…….

Project Title: Enhancing Customer Segmentation and Behaviour Analysis with RFM Clustering: A Machine Learning Approach

Word Count: ……858…… **Page Count:** ……12……

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: ……Nandheeswari Rajendran……

Date: ……31/01/2024…..……

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Nandheeswari Rajendran
Student ID: X22132210

1 Introduction

This configuration manual contains the following information such as the system requirements, the hardware configuration, and the procedure of step-by-step implementation instructions of the research approach Clustering Models based on RFM analysis. Starting with setting up the execution environment for putting the research into practice, this handbook will be helpful in developing a reasonable understanding of the prerequisites.

2 Specifications

This section discusses the software specifications and the hardware configuration used for the research.

2.1 Hardware Configuration

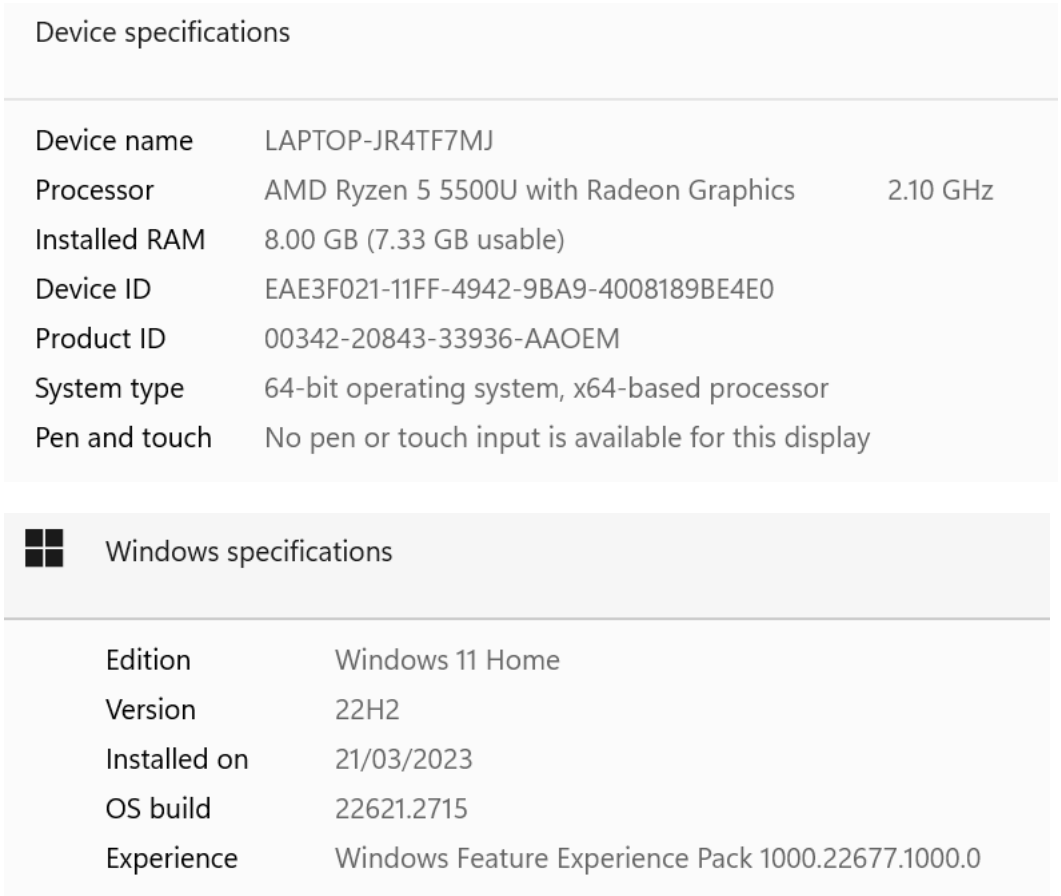

Figure 1: Hardware Configuration

2.2 Software Specifications

This section discusses the software specifications that are used in this research project.

Programming Language	Python 3.5 version
Application	Anaconda Navigator 2.4.0
Integrated Development Environment	Jupyter Notebook
Web Browsers	Microsoft Edge

Table 1: Software Specifications

3 Environment Setup

The first step is to activate the Anaconda Navigator application. This app provides an interface with the software of Juptyper Notebook to fulfil research needs. The code implementation has done in Jupyter Notebook by using the recent version of Python programming language.

3.1 Launching Jupyter Notebook Application

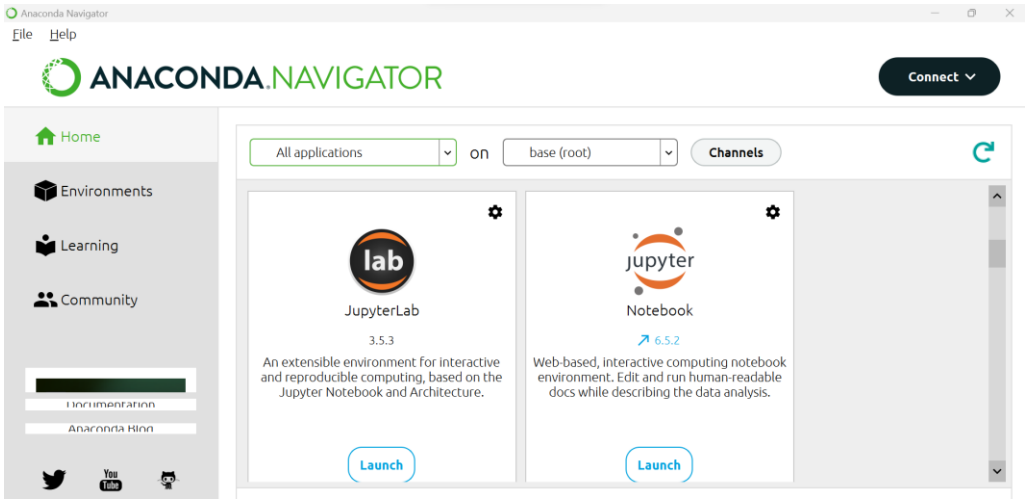

Figure 2: Anaconda Navigator Interface

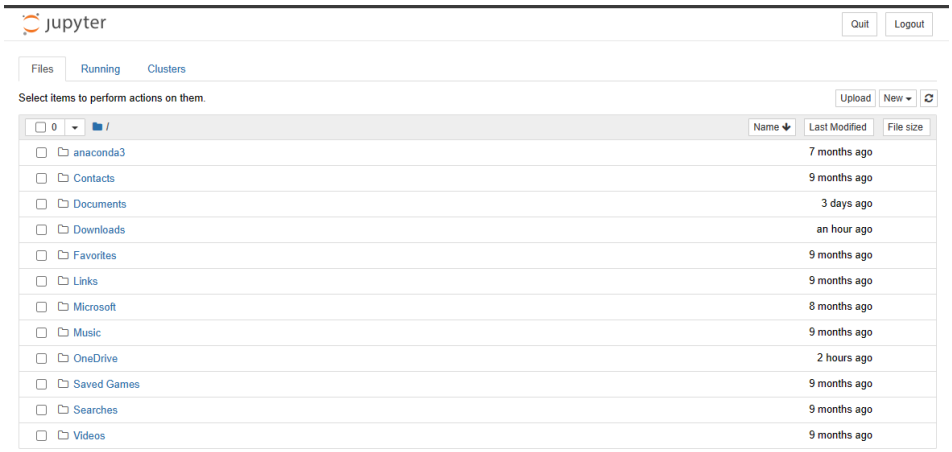

Figure 3: Jupyter Notebook Page

Before the execution of the code in Jupyter Notebook, the python code file and dataset to be loaded or placed in same folder in the local system.

Figure 4: Dataset in Local System Folder

3.2 Installing Python Packages

Finding out all the important libraries or importing packages pandas, numpy, seaborn, matplotlib and installing them is the essential process of starting the code implementation. In the Jupyter Notebook, these libraries can be installed using the pip command.

```
!pip install researchpy
import researchpy as rp
rp.summary_cont(data[['Quantity', 'UnitPrice','spend', 'Invoice_Month']])
```

```
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: researchpy in c:\users\nandh\appdata\roaming\python\python310\site-packages (0.3.5)
Requirement already satisfied: scipy in c:\programdata\anaconda3\lib\site-packages (from researchpy) (1.10.0)
Requirement already satisfied: patsy in c:\programdata\anaconda3\lib\site-packages (from researchpy) (0.5.3)
Requirement already satisfied: numpy in c:\programdata\anaconda3\lib\site-packages (from researchpy) (1.23.5)
Requirement already satisfied: statsmodels in c:\programdata\anaconda3\lib\site-packages (from researchpy) (0.13.5)
Requirement already satisfied: pandas in c:\programdata\anaconda3\lib\site-packages (from researchpy) (1.5.3)
Requirement already satisfied: pytz>=2020.1 in c:\programdata\anaconda3\lib\site-packages (from pandas->researchpy) (2022.7)
Requirement already satisfied: python-dateutil>=2.8.1 in c:\programdata\anaconda3\lib\site-packages (from pandas->researchpy)
(2.8.2)
Requirement already satisfied: six in c:\programdata\anaconda3\lib\site-packages (from patsy->researchpy) (1.16.0)
Requirement already satisfied: packaging>=21.3 in c:\programdata\anaconda3\lib\site-packages (from statsmodels->researchpy) (2
2.0)
```

Figure 5: Installing Python Necessary Library

3.3 Importing and Pre-Processing of Dataset

The Python pandas module was used to load the online retail transactional dataset as a dataFrame. The dataset was in excel format and it has 8 cloumns in which 5 are categorical and 3 of them are numerical variables.

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

```
data = pd.read_excel("C:\\Users\\Nandh\\Downloads\\Online Retail.xlsx")
```

Figure 6: Importing Online Retail Dataset

This section will provide an explanation of the implementation of the data collecting, pre-processing, and feature construction processes carried out on the dataset also it involves procedures like removing outliers and handling missing numbers. This procedure transforms data into a format suitable for modelling.

```
data.shape
```

```
(541909, 8)
```

```
data.columns
```

```
Index(['InvoiceNo', 'StockCode', 'Description', 'Quantity', 'InvoiceDate',
       'UnitPrice', 'CustomerID', 'Country'],
      dtype='object')
```

```
data.dtypes
```

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

Figure 7: Dataset Attributes

```
def missing(dff):
  print (round((dff.isnull().sum() * 100/ len(dff)),2).sort_values(ascending=False))

missing(data)
```

```
data.InvoiceDate = pd.to_datetime(data.InvoiceDate)

data.dtypes
```

```
data.groupby("CustomerID").agg("mean").sort_values(by="UnitPrice", ascending=False)
```

Figure 8: Data Pre-Processing Steps

```
data['Invoice_Year'] = data.InvoiceDate.dt.year
data['Invoice_Month'] = data.InvoiceDate.dt.month
data['Invoice_Day'] = data.InvoiceDate.dt.day
data['Invoice_WeekDay'] = data.InvoiceDate.dt.weekday
```

```
data["spend"] = data["UnitPrice"] * data["Quantity"]
```

```
import re

pattern = "^C\d+"  # Pattern to match invoice numbers starting with 'C'

cleanedf = data[data['InvoiceNo'].str.contains(pattern, na=False)]
print(cleanedf)
```

```
rfm= cleanedf.groupby('CustomerID').agg({'InvoiceDate': lambda date: (current_time - date.max()).days, #Recency
                                         'InvoiceNo': lambda num: len(num), #Frequency
                                         'spend': lambda price: price.sum()}) #Monetary
```

Figure 9: Features Extraction Process

```
def remove_outlier_with_outliers(df_in, col_name):
    q1 = df_in[col_name].quantile(0.25)
    q3 = df_in[col_name].quantile(0.75)
    iqr = q3 - q1  # Interquartile range
    fence_low = q1 - 1.5 * iqr
    fence_high = q3 + 1.5 * iqr
    outliers = df_in.loc[(df_in[col_name] < fence_low) | (df_in[col_name] > fence_high)]
    df_out = df_in.loc[(df_in[col_name] >= fence_low) & (df_in[col_name] <= fence_high)]
    return df_out, outliers
```

```
noutlier = cleanedf[["Quantity", "UnitPrice", "spend", "Invoice_Month"]].copy()

for col in noutlier.columns:
    noutlier, outliers = remove_outlier_with_outliers(noutlier, col)
    print(f"Outliers in column '{col}':")
    print(outliers)
```

Figure 11: Handling Outliers – Interquartile Range Method

4 Data Representation

4.1 Exploratory Data Analysis

```
t1=cleanedf["Country"].value_counts(sort=True).rename("Country #")
t2=cleanedf["Country"].value_counts(normalize=True,sort=True).rename("Country %")
tx= pd.concat([t1, t2], axis = 1)

print(tx)
```

Figure 10: Country Wise Percentage of Orders

```
import seaborn as sns
import matplotlib.pyplot as plt

# Boxplot
plt.figure(figsize=(8, 6))
sns.boxplot(data=noutlier[['Quantity', 'UnitPrice', 'spend', 'Invoice_Month']])
plt.title('Boxplot of Quantity, UnitPrice, Spend, Invoice_Month')
plt.show()
```

Figure 11: Distribution of Variables

```
import plotly.graph_objs as go

# Grouping the data by CustomerID and summing up their total spend
customer_spending = cleanedf.groupby('CustomerID')['spend'].sum().reset_index()

top_5_customers = customer_spending.sort_values(by='spend', ascending=False).head(5)

# Plotting the top 5 customers based on total spending
fig = go.Figure(data=[
    go.Bar(
        name='Customers With Maximum Total Purchase Amount',
        x=top_5_customers['CustomerID'].astype(str),
        y=top_5_customers['spend'],
        marker_opacity=1,
        hovertext=top_5_customers['CustomerID'].astype(str),
        marker={'color': top_5_customers['spend'], 'colorscale': 'YlGnBu'}
    )
])

fig.update_traces(texttemplate='£%{y:.3s}', textposition='inside')

fig.update_layout(
    title='Customers With Maximum Total Purchase Amount',
    xaxis_title="Customer ID",
    yaxis_title="Total Amount, £",
    plot_bgcolor='white'
)

fig.show()
```

Figure 12: Analysis of Customers Total Purchases

```
from os import path
from PIL import Image
from wordcloud import WordCloud, STOPWORDS, ImageColorGenerator
import matplotlib.pyplot as plt
```

```
months = cleanedf.groupby('Invoice_Month')['Description'].apply(','.join).reset_index()
months.set_index("Invoice_Month", inplace = True)
months
```

```
novem = months.Description[10]

stopwords = set(STOPWORDS)
stopwords.update(["abc"])

# Generating a word cloud image
wordcloud = WordCloud(max_font_size=50, max_words=300,stopwords=stopwords, background_color="white",width=800, height=400).genera

# Displaying the generated image:
plt.figure( figsize=(20,10))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.title('November Sold Items')
plt.show()
```

Figure 13: Word Cloud Analysis on Sold Products

4.2 Time Series Analysis

```
mon=cleanedf.groupby('Invoice_Month')['UnitPrice'].mean()
size=list(mon)
labels=['jan','feb','march','april','may','june','july','august','sep','oct','nov','dec']
fig1, ax1 = plt.subplots(figsize=(10,10))
plt.title('High Sales Proportion of Each Month')
ax1.pie(size, labels=labels,shadow=True,autopct='%1.1f%%')
ax1.axis('equal')
plt.show()
```

Figure 14: High Sales Proportion on Monthly Analysis

```
# Adding columns with year, month, and weekday name
timely['Year'] = timely.index.year
timely['Month'] = timely.index.month
timely['Weekday'] = timely.index.day_name()
```

```
f, axes = plt.subplots(2, figsize = (10,10))
sns.boxplot(x='Weekday', y='Quantity',  data=timely, ax=axes[0])
sns.boxplot(x='Weekday', y='spend',  data=timely, ax=axes[1])
```

```
def f(row):
    if 5 <= row['datehour'] <= 7:
        val = 'night'
    elif 8 <= row['datehour'] <= 13:
        val = 'morning'
    elif 14 <= row['datehour'] <= 17:
        val = 'afternoon'
    elif 18 <= row['datehour'] <= 22:
        val = 'evening'
    else:
        val = 'notime'
    return val

clean['time'] = clean.apply(f, axis=1)
```

Figure 15: Weekly and Daily Transaction Analysis

```
# Monetary,Frequency, and Recency Analysis

attributes = ['monetary','frequency','recency']
plt.rcParams['figure.figsize'] = [10,8]
sns.boxplot(data = rfmv2[attributes], orient="v", palette="Set2" ,whis=1.5,saturation=1, width=0.7)
plt.title("Outliers Variable Distribution", fontsize = 14, fontweight = 'bold')
plt.ylabel("Range", fontweight = 'bold')
plt.xlabel("Attributes", fontweight = 'bold')
```

Figure 16: RFM Outlier Distribution

```
fig, axs = plt.subplots(2, 2, figsize=(10, 10))

sns.histplot(data=rfm_df_scaled, x="monetary", bins= 10, kde=True, color="skyblue", ax=axs[0, 0])
sns.histplot(data=rfm_df_scaled, x="frequency", bins= 10, kde=True, color="olive", ax=axs[0, 1])
sns.histplot(data=rfm_df_scaled, x="recency", bins= 10, kde=True, color="gold", ax=axs[1, 0])
plt.show()
```

```
sns.heatmap(rfm_df_scaled.iloc[:, 0:3].corr(), annot=True)
```

Figure 17: Visualization of RFM Features

5 Project Implementation

This section focuses on the implementation details that explores three different clustering models and their performance evaluation metrics.

```
from matplotlib import colors
import seaborn as sns
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from yellowbrick.cluster import KElbowVisualizer
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt, numpy as np
from mpl_toolkits.mplot3d import Axes3D
from sklearn.cluster import AgglomerativeClustering
from matplotlib.colors import ListedColormap
from sklearn import metrics
import warnings
import sys
if not sys.warnoptions:
    warnings.simplefilter("ignore")
```

Figure 18: Importing Python Libraries for Modelling and Metrics Evaluation

5.1 Model 1 – Agglomerative Clustering

```
print('Elbow Method to determine the number of clusters to be formed:')
Elbow_M = KElbowVisualizer(KMeans(), k=4)
Elbow_M.fit(knndf)
Elbow_M.show()
```

```
from scipy.cluster.hierarchy import dendrogram, linkage
import matplotlib.pyplot as plt

# Perform Agglomerative Clustering and obtain linkage matrix

Z = linkage(knndf, 'ward')  # 'ward' linkage method used here

plt.figure(figsize=(12, 8))
dendrogram(Z)
plt.title('Dendrogram for Optimal Cluster Count')
plt.xlabel('Data Points')
plt.ylabel('Distance')
plt.show()
```

Figure 19: Elbow Method and Dendrogram Method

```
#Initiating the Agglomerative Clustering model

AC = AgglomerativeClustering(n_clusters=4)

# fit model and predict clusters

yhat_AC = AC.fit_predict(knndf)

knndf["Clusters"] = yhat_AC

#Adding the Clusters feature to the orignal dataframe

knndf["Clusters"] = yhat_AC
```

Figure 20: Agglomertive Clustering

```
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
from matplotlib import colors

cmap = colors.ListedColormap(["#682F2F", "#9E726F", "#D6B2B1", "#B9C0C9", "#9F8A78", "#F3AB60"])

x = knndf["recency"]
y = knndf["frequency"]
z = knndf["monetary"]

# Plotting the clusters
fig = plt.figure(figsize=(10, 8))
ax = plt.subplot(111, projection='3d')

# Scatter plot
scatter = ax.scatter(x, y, z, s=40, c=knndf["Clusters"], marker='o', cmap=cmap)

# Labels for the axes
ax.set_xlabel('Recency')
ax.set_ylabel('Frequency')
ax.set_zlabel('Monetary')

ax.set_title("The Plot Of The Clusters")
plt.show()
```

Figure 21: Plotting the Dimension of Clusters

5.2 Model 2 – K-Means Algorithm

```
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans

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(range_n_clusters, ssd, marker='o')
plt.xlabel('Number of Clusters')
plt.ylabel('Sum of Squared Distances')
plt.title('Elbow Method for Optimal k')
plt.show()
```

Figure 22: Elbow Method 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 = metrics.silhouette_score(rfm_df_scaled, cluster_labels)
    print("For n_clusters={0}, the silhouette score is {1}".format(num_clusters, silhouette_avg))
```

Figure 23: Parameter Tuning - Silhouette Analysis

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

```
KMeans(max_iter=50, n_clusters=3)
```

```
# assign the label
rfm_df_scaled['Cluster_Id'] = kmeans.labels_
rfm_df_scaled.head()
```

```
import seaborn as sns
import matplotlib.pyplot as plt

data_for_scatterplot = rfm_df_scaled[['monetary', 'frequency', 'Cluster_Id']]

plt.figure(figsize=(12, 8))
sns.scatterplot(data=data_for_scatterplot, x='monetary', y='frequency', hue='Cluster_Id', palette=sns.color_palette('hls', 3))
plt.title('KMeans with 3 Clusters')
plt.xlabel('monetary')
plt.ylabel('frequency')
plt.show()
```

Figure 24 : Distribution of K-Means Clustering

5.3 Model 3 – Mean-Shift Algorithm

```
from sklearn.cluster import MeanShift

# Creating and fitting MeanShift clustering model
ms = MeanShift()
ms.fit(rfm_df_scaled[['monetary', 'frequency']])  # Considering 'monetary' and 'frequency' for clustering

# Assigning cluster labels to the original DataFrame
rfm_df_scaled['Cluster_Id'] = ms.labels_

# Printing the unique cluster labels
print("Unique cluster labels:", rfm_df_scaled['Cluster_Id'].unique())
```

Figure 24 : Mean-Shift Clustering Modelling

```
from sklearn.cluster import MeanShift
import seaborn as sns
import matplotlib.pyplot as plt

ms = MeanShift()
ms.fit(rfm_df_scaled[['monetary', 'frequency']])

rfm_df_scaled['Cluster_Id'] = ms.labels_

plt.figure(figsize=(12, 8))

# Custom color palette
custom_palette = sns.color_palette('husl', n_colors=len(rfm_df_scaled['Cluster_Id'].unique()))

sns.scatterplot(data=rfm_df_scaled, x='monetary', y='frequency', hue='Cluster_Id', palette=custom_palette)
plt.title('MeanShift Clustering')

# Adjust legend placement outside the plot area
plt.legend(title='Cluster ID', bbox_to_anchor=(1.05, 1), loc='upper left')
plt.show()
```

Figure 25 : Distribution of Mean-shift Clustering

```
import seaborn as sns
import matplotlib.pyplot as plt

plt.figure(figsize=(8, 6))

# Countplot to visualize cluster distribution

sns.countplot(data=rfm_df_scaled, x='Cluster_Id', palette='viridis')
plt.title('Distribution of Data Points in Clusters (Mean-Shift)')
plt.xlabel('Cluster ID')
plt.ylabel('Count of Data Points')

plt.show()
```

Figure 26 : Visualization of Cluster Distribution

5.4 Model Evaluations

This section of research evaluates the implemented clustering models by comprehensive analysis of the performance evaluation metrics.

Metric 1 - Silhouette Score

```
# Agglemorative Model

metrics.silhouette_score(knndf, yhat_AC, metric='euclidean')
```

```
# K-Means Model

metrics.silhouette_score(rfm_df_scaled, kmeans.labels_, metric='euclidean')
```

```
# Mean-shift Model

metrics.silhouette_score(rfm_df_scaled, ms.labels_, metric='euclidean')
```

Metric 2 - Calinski-Harabasz index

```
# Agglemorative Model

metrics.calinski_harabasz_score(knndf, yhat_AC)
```

```
# Kmeans Model

metrics.calinski_harabasz_score(rfm_df_scaled, kmeans.labels_)
```

```
# Mean-shift Model

metrics.calinski_harabasz_score(rfm_df_scaled, ms.labels_)
```

Metric 3 - Davies-Bouldin Index

```
# Agglemorative Model

from sklearn.metrics import davies_bouldin_score
davies_bouldin_score(knndf, yhat_AC)
```

```
# Kmeans Model

davies_bouldin_score(rfm_df_scaled, kmeans.labels_)
```

```
# Mean-shift Model

davies_bouldin_score(rfm_df_scaled, ms.labels_)
```

Figure 27: Clustering Performance Evaluation