CustSegProj_CGrubb

March 11, 2025

0.1 Note: This is a sample solution for the project. Projects will NOT be graded on the basis of how well the submission matches this sample solution. Projects will be graded on the basis of the rubric only.

1 Problem Statement

Business Context Understanding customer personality and behavior is pivotal for businesses to enhance customer satisfaction and increase revenue. Segmentation based on a customer's personality, demographics, and purchasing behavior allows companies to create tailored marketing campaigns, improve customer retention, and optimize product offerings.

A leading retail company with a rapidly growing customer base seeks to gain deeper insights into their customers' profiles. The company recognizes that understanding customer personalities, lifestyles, and purchasing habits can unlock significant opportunities for personalizing marketing strategies and creating loyalty programs. These insights can help address critical business challenges, such as improving the effectiveness of marketing campaigns, identifying high-value customer groups, and fostering long-term relationships with customers.

With the competition intensifying in the retail space, moving away from generic strategies to more targeted and personalized approaches is essential for sustaining a competitive edge.

Objective In an effort to optimize marketing efficiency and enhance customer experience, the company has embarked on a mission to identify distinct customer segments. By understanding the characteristics, preferences, and behaviors of each group, the company aims to:
1. Develop personalized marketing campaigns to increase conversion rates.
2. Create effective retention strategies for high-value customers.
3. Optimize resource allocation, such as inventory management, pricing strategies, and store layouts.

As a data scientist tasked with this project, your responsibility is to analyze the given customer data, apply machine learning techniques to segment the customer base, and provide actionable insights into the characteristics of each segment.

Data Dictionary The dataset includes historical data on customer demographics, personality traits, and purchasing behaviors. Key attributes are:

1. **Customer Information**
    - **ID:** Unique identifier for each customer.
    - **Year_Birth:** Customer's year of birth.
    - **Education:** Education level of the customer.
    - **Marital_Status:** Marital status of the customer.
    - **Income:** Yearly household income (in dollars).
    - **Kidhome:** Number of children in the household.
    - **Teenhome:** Number of teenagers in the household.
    - **Dt_Customer:** Date when the customer enrolled with the company.
    - **Recency:** Number of days since the customer's last purchase.
    - **Complain:** Whether the customer complained in the last 2 years (1 for yes, 0 for no).
2. **Spending Information (Last 2 Years)**
    - **MntWines:** Amount spent on wine.
    - **MntFruits:** Amount spent on fruits.
    - **MntMeatProducts:** Amount spent on meat.
    - **MntFishProducts:** Amount spent on fish.
    - **MntSweetProducts:** Amount spent on sweets.
    - **MntGoldProds:** Amount spent on gold products.
3. **Purchase and Campaign Interaction**
    - **NumDealsPurchases:** Number of purchases made using a discount.
    - **AcceptedCmp1:** Response to the 1st campaign (1 for yes, 0 for no).
    - **AcceptedCmp2:** Response to the 2nd campaign (1 for yes, 0 for no).
    - **AcceptedCmp3:** Response to the 3rd campaign (1 for yes, 0 for no).
    - **AcceptedCmp4:** Response to the 4th campaign (1 for yes, 0 for no).
    - **AcceptedCmp5:** Response to the 5th campaign (1 for yes, 0 for no).
    - **Response:** Response to the last campaign (1 for yes, 0 for no).
4. **Shopping Behavior**
    - **NumWebPurchases:** Number of purchases made through the company's website.

- **NumCatalogPurchases:** Number of purchases made using catalogs.
- **NumStorePurchases:** Number of purchases made directly in stores.
- **NumWebVisitsMonth:** Number of visits to the company's website in the last month.

2 Let's start coding!

2.1 Importing necessary libraries

[28]:
```
# Libraries to help with reading and manipulating data
import pandas as pd
import numpy as np

# libaries to help with data visualization
import matplotlib.pyplot as plt
import seaborn as sns

# Removes the limit for the number of displayed columns
pd.set_option("display.max_columns", None)
# Sets the limit for the number of displayed rows
pd.set_option("display.max_rows", 200)

# to scale the data using z-score
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA

# to compute distances
from scipy.spatial.distance import cdist, pdist

# to perform k-means clustering and compute silhouette scores
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score

# to visualize the elbow curve and silhouette scores
from yellowbrick.cluster import KElbowVisualizer, SilhouetteVisualizer

# to perform hierarchical clustering, compute cophenetic correlation, and␣
↪create dendrograms
from sklearn.cluster import AgglomerativeClustering
from scipy.cluster.hierarchy import dendrogram, linkage, cophenet

# to suppress warnings
import warnings
from datetime import datetime as dt #For casting object type column to date
```

```
warnings.filterwarnings("ignore")
```

2.2 Loading the data

[2]:
```
# uncomment and run the following line if using Google Colab
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

[3]:
```
# loading data into a pandas dataframe
data = pd.read_csv("/content/drive/MyDrive/MIT IDSS/Projects/Making Sense of␣
↪Unstructured Data/marketing_campaign.csv", sep="\t")
```

2.3 Data Overview

Question 1: What are the data types of all the columns?

[5]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2240 entries, 0 to 2239
Data columns (total 29 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   ID                   2240 non-null   int64
 1   Year_Birth           2240 non-null   int64
 2   Education            2240 non-null   object
 3   Marital_Status       2240 non-null   object
 4   Income               2216 non-null   float64
 5   Kidhome              2240 non-null   int64
 6   Teenhome             2240 non-null   int64
 7   Dt_Customer          2240 non-null   object
 8   Recency              2240 non-null   int64
 9   MntWines             2240 non-null   int64
 10  MntFruits            2240 non-null   int64
 11  MntMeatProducts      2240 non-null   int64
 12  MntFishProducts      2240 non-null   int64
 13  MntSweetProducts     2240 non-null   int64
 14  MntGoldProds         2240 non-null   int64
 15  NumDealsPurchases    2240 non-null   int64
 16  NumWebPurchases      2240 non-null   int64
 17  NumCatalogPurchases  2240 non-null   int64
 18  NumStorePurchases    2240 non-null   int64
 19  NumWebVisitsMonth    2240 non-null   int64
 20  AcceptedCmp3         2240 non-null   int64
 21  AcceptedCmp4         2240 non-null   int64
 22  AcceptedCmp5         2240 non-null   int64
```

```
 23  AcceptedCmp1         2240 non-null   int64
 24  AcceptedCmp2         2240 non-null   int64
 25  Complain             2240 non-null   int64
 26  Z_CostContact        2240 non-null   int64
 27  Z_Revenue            2240 non-null   int64
 28  Response             2240 non-null   int64
dtypes: float64(1), int64(25), object(3)
memory usage: 507.6+ KB
```

Observations: Columns containing NULL Values: Income
DT_Customer is an object. Will need converted to date. (FIXED AT Q4) Most columns are integers/floats with the exception of Dt_customer (should be date), Marital_status, and Education.

Note: AcceptedCmp 1-5, Complain, and Response columns are all Binary. Binary data points can affect clusterings based on distances.

Question 2: Check the statistical summary of the data. What is the average household income?

[6]:
```python
print("AVG Household Income: ", round(data.Income.mean(),2))

data.describe()
```

```
AVG Household Income:  52247.25
```

[6]:

	ID	Year_Birth	Income	Kidhome	Teenhome	\
count	2240.000000	2240.000000	2216.000000	2240.000000	2240.000000	
mean	5592.159821	1968.805804	52247.251354	0.444196	0.506250	
std	3246.662198	11.984069	25173.076661	0.538398	0.544538	
min	0.000000	1893.000000	1730.000000	0.000000	0.000000	
25%	2828.250000	1959.000000	35303.000000	0.000000	0.000000	
50%	5458.500000	1970.000000	51381.500000	0.000000	0.000000	
75%	8427.750000	1977.000000	68522.000000	1.000000	1.000000	
max	11191.000000	1996.000000	666666.000000	2.000000	2.000000	

	Recency	MntWines	MntFruits	MntMeatProducts	\
count	2240.000000	2240.000000	2240.000000	2240.000000	
mean	49.109375	303.935714	26.302232	166.950000	
std	28.962453	336.597393	39.773434	225.715373	
min	0.000000	0.000000	0.000000	0.000000	
25%	24.000000	23.750000	1.000000	16.000000	
50%	49.000000	173.500000	8.000000	67.000000	
75%	74.000000	504.250000	33.000000	232.000000	
max	99.000000	1493.000000	199.000000	1725.000000	

	MntFishProducts	MntSweetProducts	MntGoldProds	NumDealsPurchases	\
count	2240.000000	2240.000000	2240.000000	2240.000000	
mean	37.525446	27.062946	44.021875	2.325000	
std	54.628979	41.280498	52.167439	1.932238	

```
min         0.000000           0.000000       0.000000             0.000000
25%         3.000000           1.000000       9.000000             1.000000
50%        12.000000           8.000000      24.000000             2.000000
75%        50.000000          33.000000      56.000000             3.000000
max       259.000000         263.000000     362.000000            15.000000

       NumWebPurchases  NumCatalogPurchases  NumStorePurchases  \
count      2240.000000          2240.000000        2240.000000
mean          4.084821             2.662054           5.790179
std           2.778714             2.923101           3.250958
min           0.000000             0.000000           0.000000
25%           2.000000             0.000000           3.000000
50%           4.000000             2.000000           5.000000
75%           6.000000             4.000000           8.000000
max          27.000000            28.000000          13.000000

       NumWebVisitsMonth  AcceptedCmp3  AcceptedCmp4  AcceptedCmp5  \
count        2240.000000   2240.000000   2240.000000   2240.000000
mean            5.316518      0.072768      0.074554      0.072768
std             2.426645      0.259813      0.262728      0.259813
min             0.000000      0.000000      0.000000      0.000000
25%             3.000000      0.000000      0.000000      0.000000
50%             6.000000      0.000000      0.000000      0.000000
75%             7.000000      0.000000      0.000000      0.000000
max            20.000000      1.000000      1.000000      1.000000

       AcceptedCmp1  AcceptedCmp2     Complain  Z_CostContact  Z_Revenue  \
count   2240.000000   2240.000000  2240.000000         2240.0     2240.0
mean       0.064286      0.013393     0.009375            3.0       11.0
std        0.245316      0.114976     0.096391            0.0        0.0
min        0.000000      0.000000     0.000000            3.0       11.0
25%        0.000000      0.000000     0.000000            3.0       11.0
50%        0.000000      0.000000     0.000000            3.0       11.0
75%        0.000000      0.000000     0.000000            3.0       11.0
max        1.000000      1.000000     1.000000            3.0       11.0

          Response
count  2240.000000
mean      0.149107
std       0.356274
min       0.000000
25%       0.000000
50%       0.000000
75%       0.000000
max       1.000000
```

Observations: Average Household Income (mean) is $52,245.25 rounded.

Question 3: Are there any missing values in the data? If yes, treat them using an appropriate method

[7]:
```
# Write your code here
print("Number of rows that are NULL:",len(data[data.Income.isnull()]))
print("Percentage of rows that are NULL: ", (len(data[data.Income.isnull()])/
 ↪len(data))*100)
"""~ 1% of rows are NULL for income. That is a low percentage, so deleting the␣
 ↪rows"""
data_cleaned = data.dropna()
```

```
Number of rows that are NULL: 24
Percentage of rows that are NULL:  1.0714285714285714
```

Observations: Income was the only column containing NULLs there were only 24 rows that were NULL, equating to ~ 1.07% of the data. This being such a small percentage of the data, I chose to drop the rows for analysis. this way they do not affect the averages and we still have plenty of data to work with.

Question 4: Are there any duplicates in the data?

[8]:
```
#Check for duplicate data
data_cleaned[data_cleaned.duplicated()]
```

[8]:
```
Empty DataFrame
Columns: [ID, Year_Birth, Education, Marital_Status, Income, Kidhome, Teenhome,
Dt_Customer, Recency, MntWines, MntFruits, MntMeatProducts, MntFishProducts,
MntSweetProducts, MntGoldProds, NumDealsPurchases, NumWebPurchases,
NumCatalogPurchases, NumStorePurchases, NumWebVisitsMonth, AcceptedCmp3,
AcceptedCmp4, AcceptedCmp5, AcceptedCmp1, AcceptedCmp2, Complain, Z_CostContact,
Z_Revenue, Response]
Index: []
```

Observations: NO duplicates in the data showing.

Additional Data Clean-up / Prep

[9]:
```
#Fixing the data type for Dt_Customer
data_cleaned['Dt_Customer'] = pd.to_datetime(data_cleaned['Dt_Customer'],␣
 ↪format="%d-%m-%Y")
reference_date = data_cleaned['Dt_Customer'].max()
#Add column to use in place of date in clustering
data_cleaned['Days_Since_Enrollment'] =  (reference_date -␣
 ↪data_cleaned['Dt_Customer']).dt.days

#Add column to do a percentage of the binary columns(except complaint). This␣
 ↪way the data can still be included without affecting the clustering.
```

```
data_cleaned['perc_cmps'] = (data_cleaned['AcceptedCmp1'] +␣
↪data_cleaned['AcceptedCmp2'] + data_cleaned['AcceptedCmp3'] +␣
↪data_cleaned['AcceptedCmp4'] + data_cleaned['AcceptedCmp5'] +␣
↪data_cleaned['Response'])/6.0
data_cleaned.head()
```

[9]:

	ID	Year_Birth	Education	Marital_Status	Income	Kidhome	Teenhome
0	5524	1957	Graduation	Single	58138.0	0	0
1	2174	1954	Graduation	Single	46344.0	1	1
2	4141	1965	Graduation	Together	71613.0	0	0
3	6182	1984	Graduation	Together	26646.0	1	0
4	5324	1981	PhD	Married	58293.0	1	0

	Dt_Customer	Recency	MntWines	MntFruits	MntMeatProducts	MntFishProducts
0	2012-09-04	58	635	88	546	172
1	2014-03-08	38	11	1	6	2
2	2013-08-21	26	426	49	127	111
3	2014-02-10	26	11	4	20	10
4	2014-01-19	94	173	43	118	46

	MntSweetProducts	MntGoldProds	NumDealsPurchases	NumWebPurchases
0	88	88	3	8
1	1	6	2	1
2	21	42	1	8
3	3	5	2	2
4	27	15	5	5

	NumCatalogPurchases	NumStorePurchases	NumWebVisitsMonth	AcceptedCmp3
0	10	4	7	0
1	1	2	5	0
2	2	10	4	0
3	0	4	6	0
4	3	6	5	0

	AcceptedCmp4	AcceptedCmp5	AcceptedCmp1	AcceptedCmp2	Complain
0	0	0	0	0	0
1	0	0	0	0	0
2	0	0	0	0	0
3	0	0	0	0	0
4	0	0	0	0	0

	Z_CostContact	Z_Revenue	Response	Days_Since_Enrollment	perc_cmps
0	3	11	1	663	0.166667
1	3	11	0	113	0.000000
2	3	11	0	312	0.000000
3	3	11	0	139	0.000000
4	3	11	0	161	0.000000

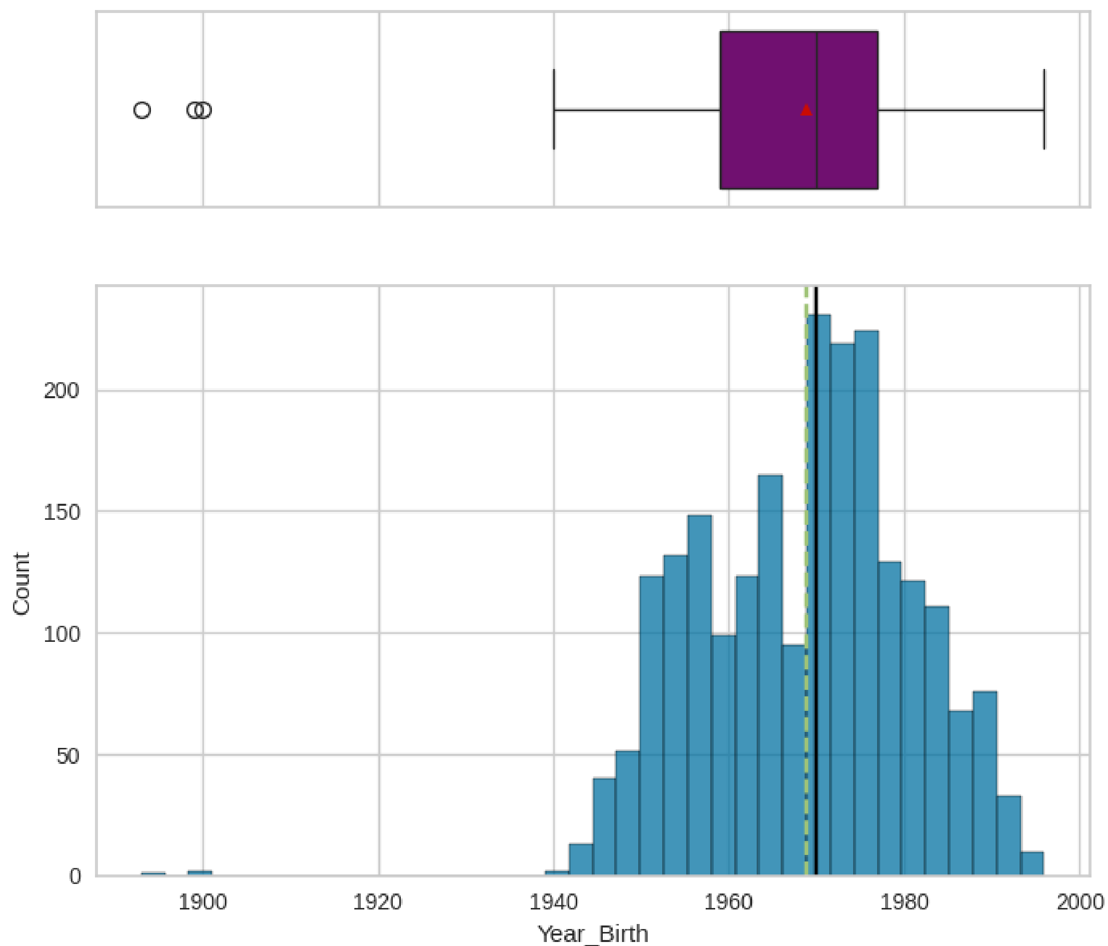
2.4 Exploratory Data Analysis

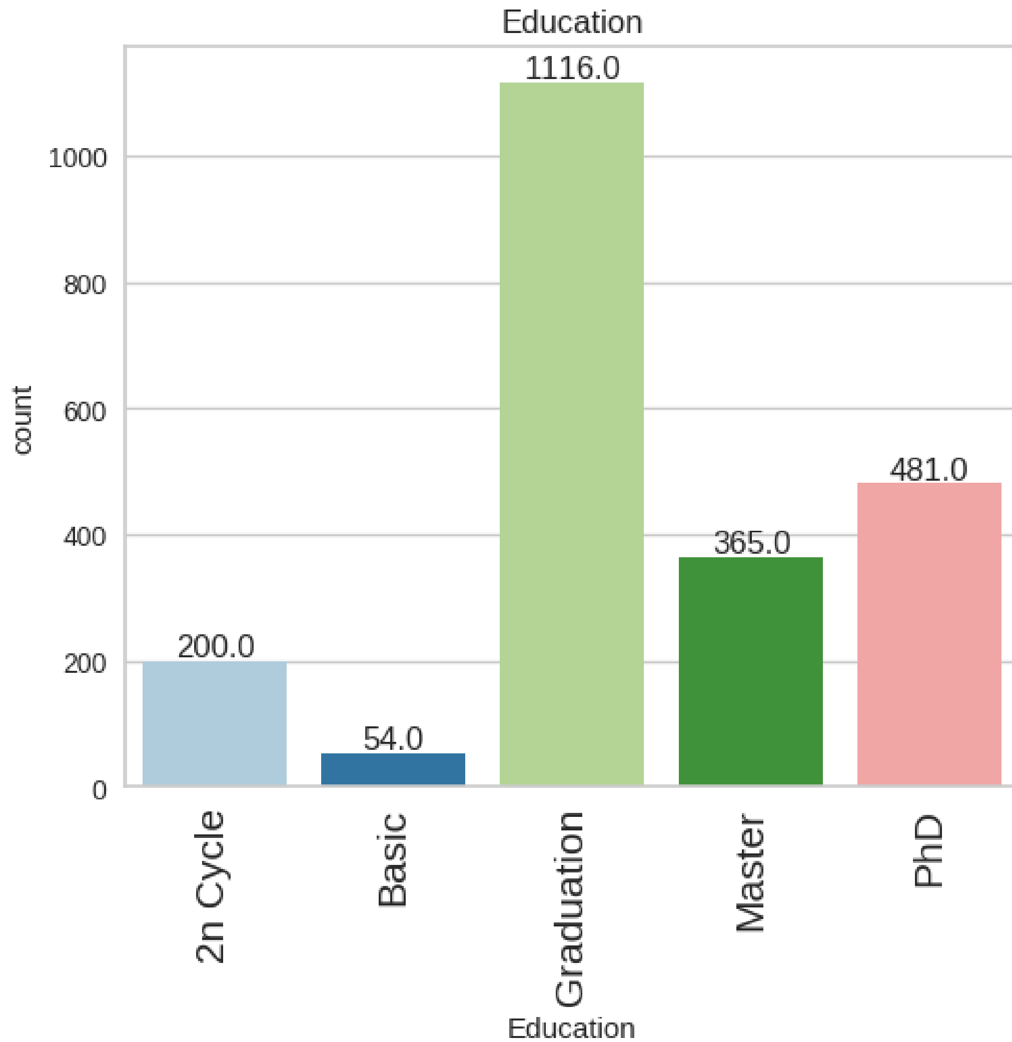
2.4.1 Univariate Analysis

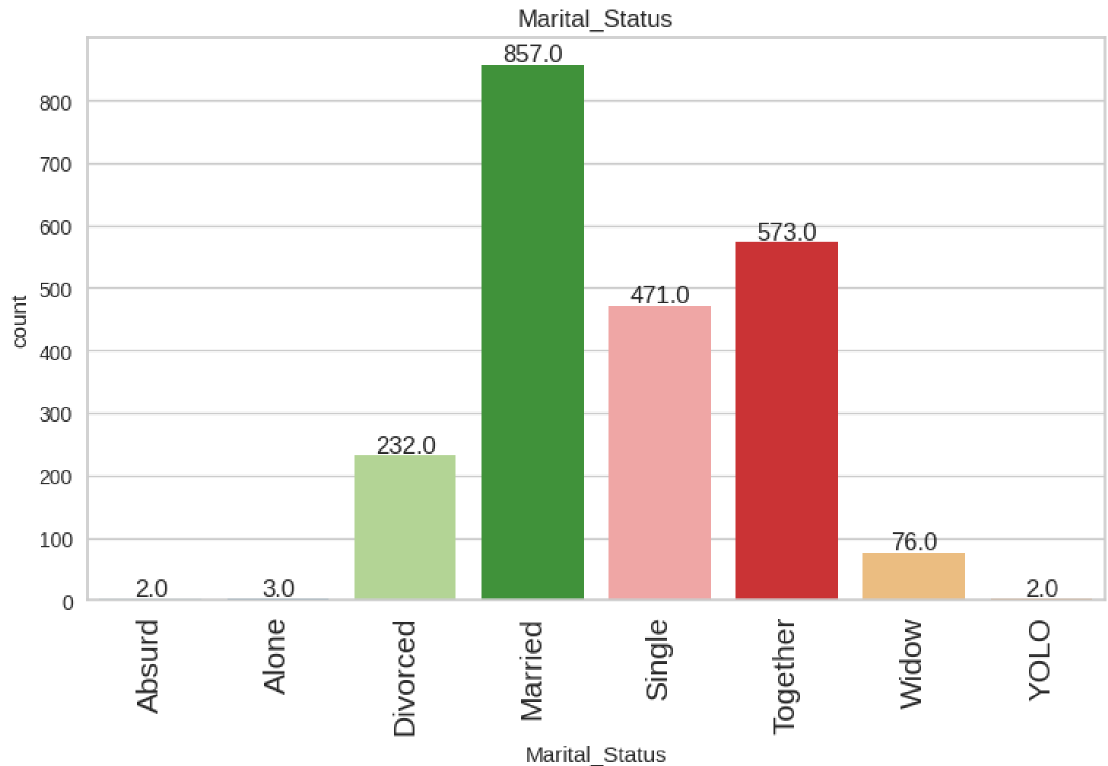
Question 5: Explore all the variables and provide observations on their distributions. (histograms and boxplots)

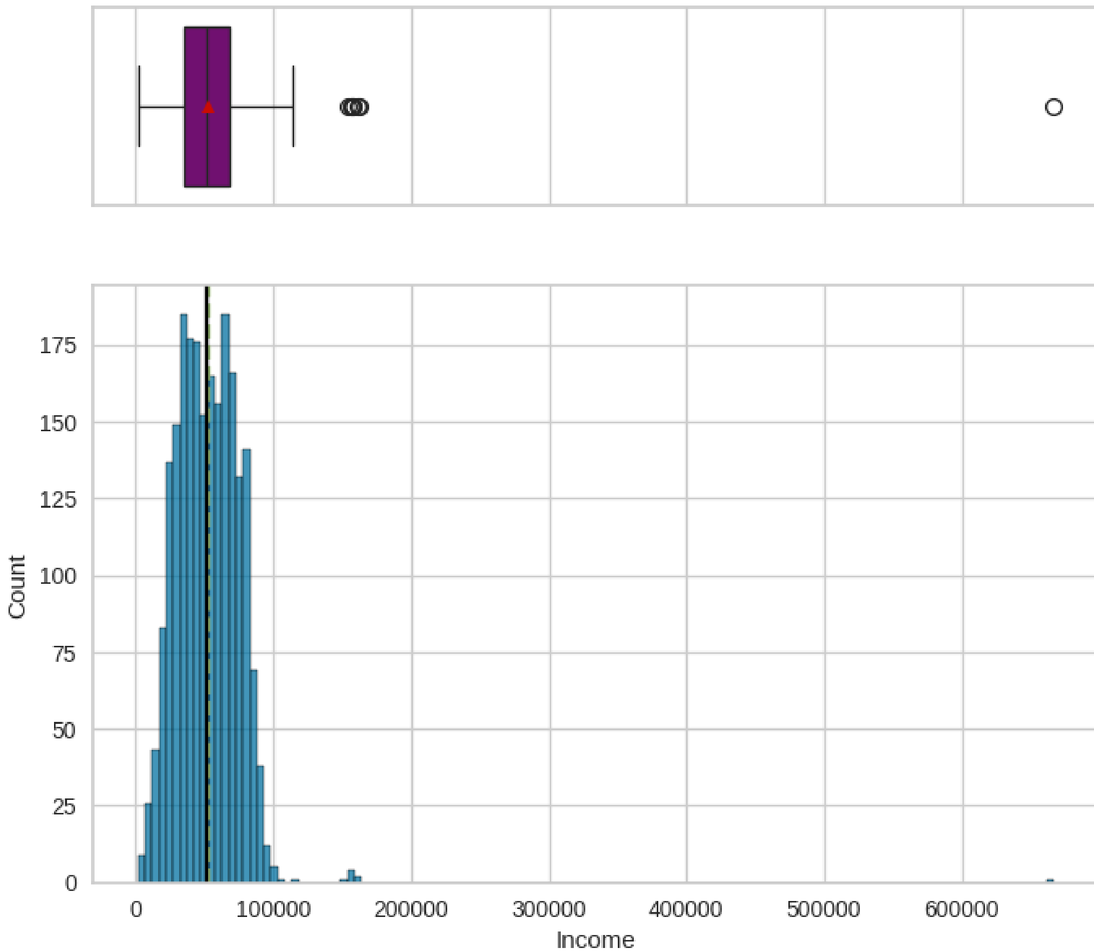
[10]:
```
#Define reusable subplot function
def sub_plots(nrows = 2,  gridspec_kw = {"height_ratios": (.25, .75)}
                  , figsize = (8,7)):

        f, (ax_box, ax_hist) = plt.subplots(
        nrows = nrows,  # Number of rows of the subplot grid
        sharex = True,  # The X-axis will be shared among all the subplots
        gridspec_kw = gridspec_kw,
        figsize = figsize)

        return f, ax_box, ax_hist


#Define histogram boxplot combo:
def box_hist_plot( x):

    f, ax_box, ax_hist = sub_plots()

    sns.boxplot(x=x, ax=ax_box, showmeans=True, color='purple')
    sns.histplot(x=x, kde=False, ax=ax_hist, bins="auto")
    ax_hist.axvline(x.mean(), color='g', linestyle='--')      # Add mean to the␣
 ↪histogram
    ax_hist.axvline(x.median(), color='black', linestyle='-') # Add median to␣
 ↪the histogram

    plt.show()


#BAR CHART FOR STRING DATA

def barplot(data, feature, perc=False, top_n = None):
  total = len(data[feature])
  count = data[feature].nunique()
  f = (count+1,5) if top_n is None else (top_n+1,5)
  plt.figure(figsize=f)
  plt.title(feature)
  plt.xticks(rotation=90, fontsize=15)
  ax = sns.countplot(
                    data = data,
                    x = feature,
                    palette = "Paired",
```

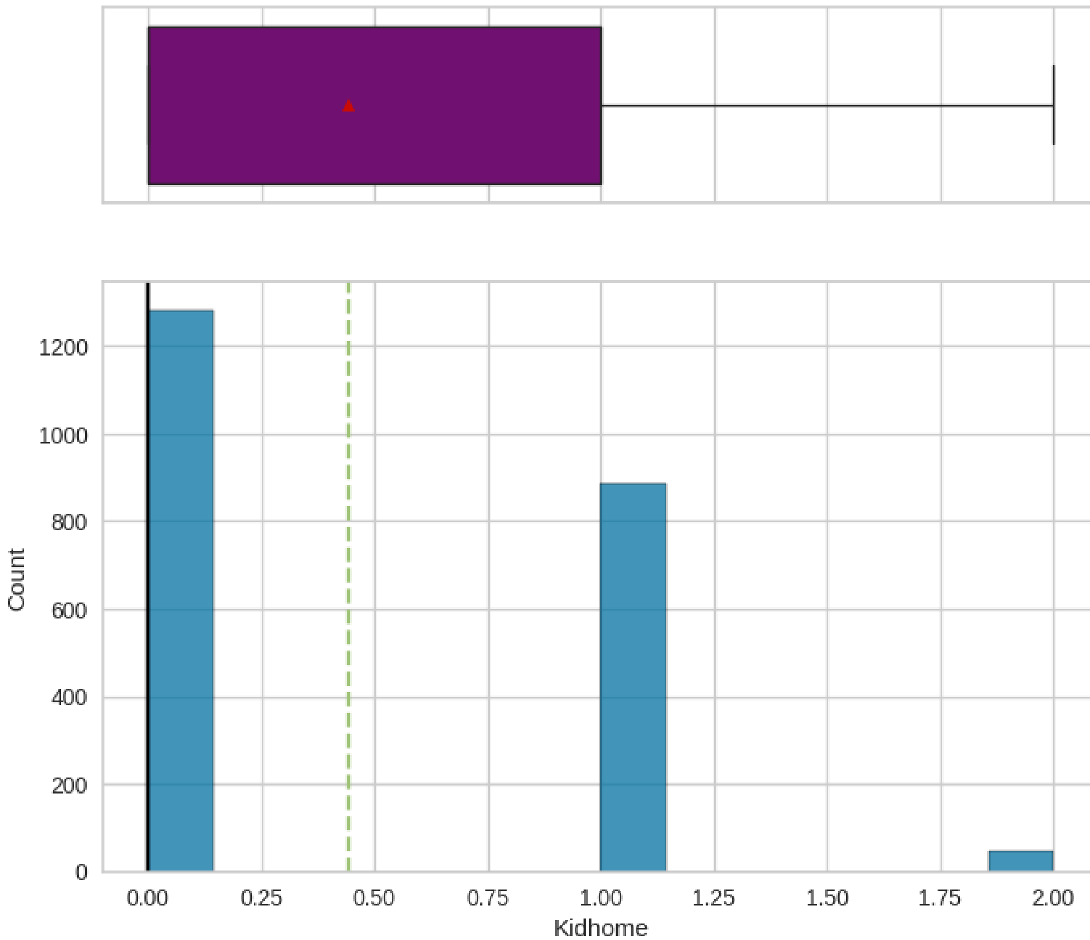
```
                      order = data[feature].value_counts().index[:top_n].
 ↪sort_values(),
                      )
  for p in ax.patches:

      label = "{:.1f}%".format(100 * p.get_height() / total)  if perc else p.
 ↪get_height()

      x = p.get_x() + p.get_width() / 2  # Width of the plot
      y = p.get_height()                 # Height of the plot

      ax.annotate(
          label,
          (x, y),
          ha = "center",
          va = "center",
          size = 12,
          xytext = (0, 5),
          textcoords = "offset points",
      )  # Annotate the percentage
  plt.show()

#avoiding unique identifier ID and date/binary columns that don't represent␣
 ↪well in box/hist plots or bar charts
avoid_columns␣
 ↪=["ID","Dt_Customer",'AcceptedCmp1','AcceptedCmp2','AcceptedCmp3','AcceptedCmp4','AcceptedC
 ↪'Complain','Response',"Z_CostContact",'Z_Revenue']
for col in data_cleaned.columns:
  if col in avoid_columns:
    continue #Skip and go to next iteration
  elif data_cleaned[col].dtype in ['int64','float64']:
      box_hist_plot(data_cleaned[col])
  elif data_cleaned[col].dtype in ['str', 'object']:
    barplot(data_cleaned, col)
```


200
150
100
50
0
Count
1900
1920
1940
1960
1980
2000
Year_Birth

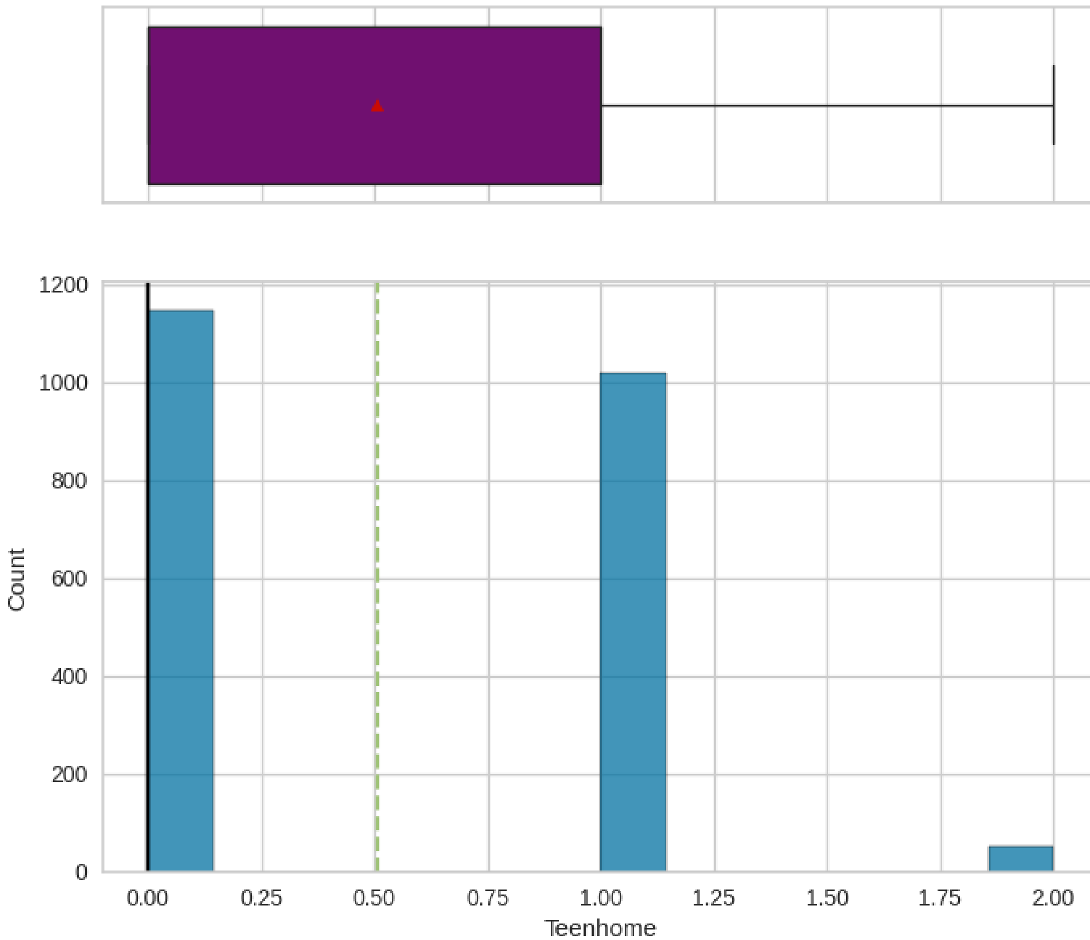

Education
1116.0
481.0
365.0
200.0
54.0
1000
800
600
400
200
0
count
2n Cycle
Basic
Graduation
Master
PhD
Education

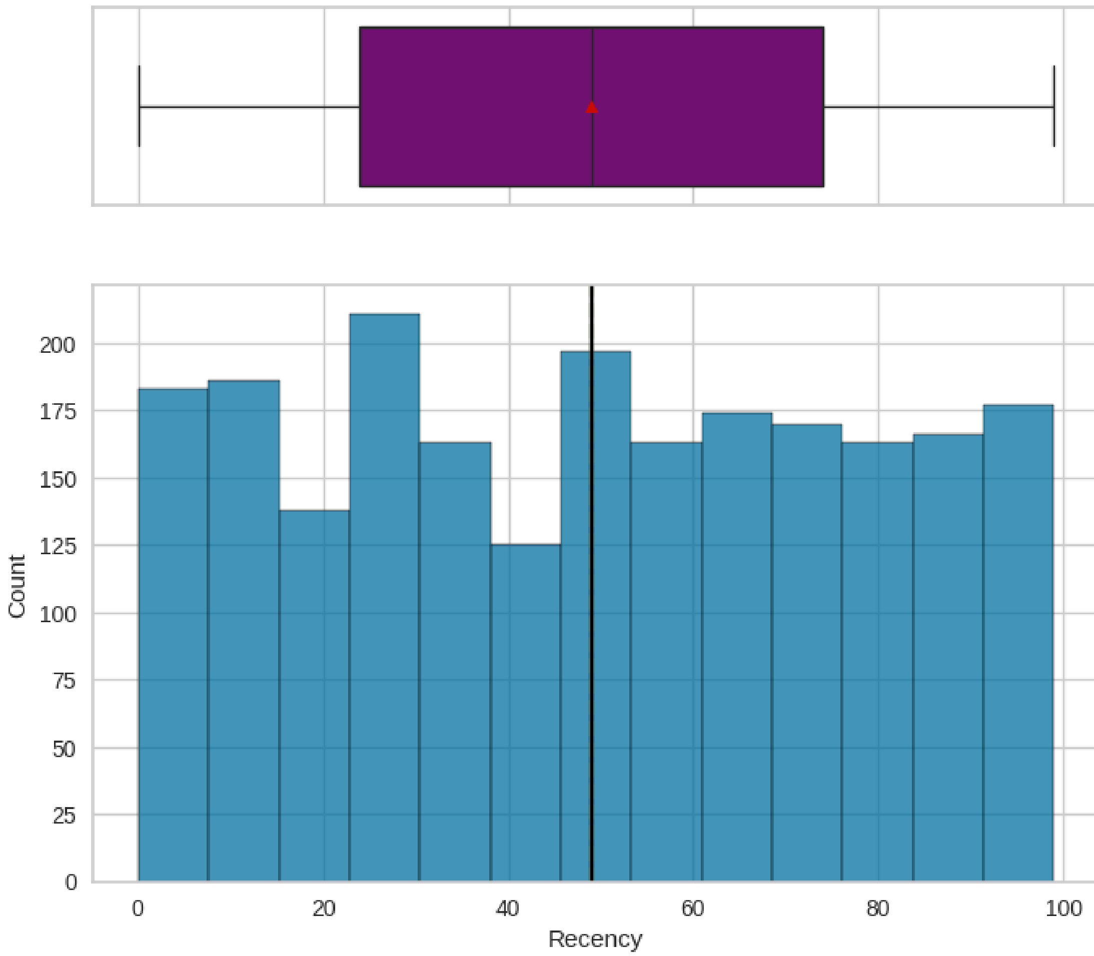

Marital_Status
count
0
100
200
300
400
500
600
700
800
2.0
3.0
232.0
857.0
471.0
573.0
76.0
2.0
Absurd
Alone
Divorced
Married
Single
Together
Widow
YOLO
Marital_Status

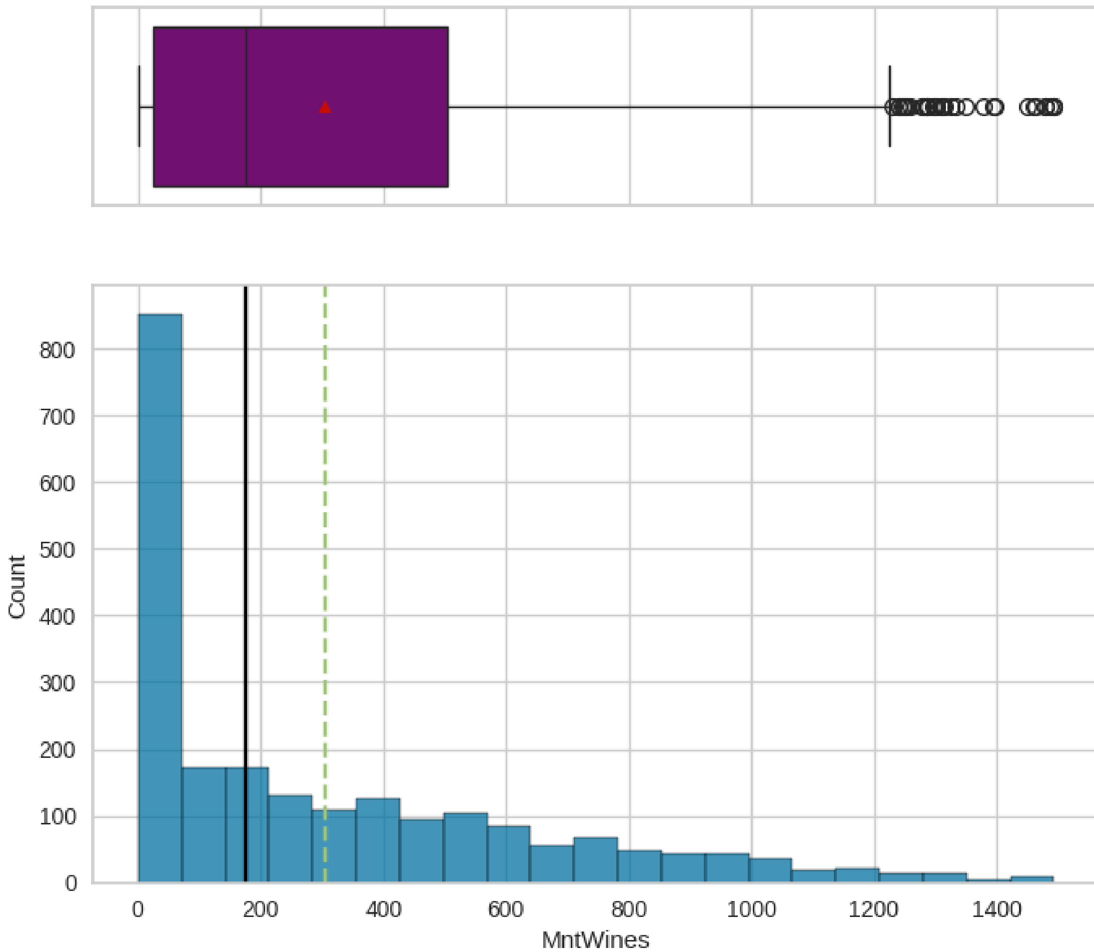

175
150
125
100
75
50
25
0
Count
0
100000
200000
300000
400000
500000
600000
Income



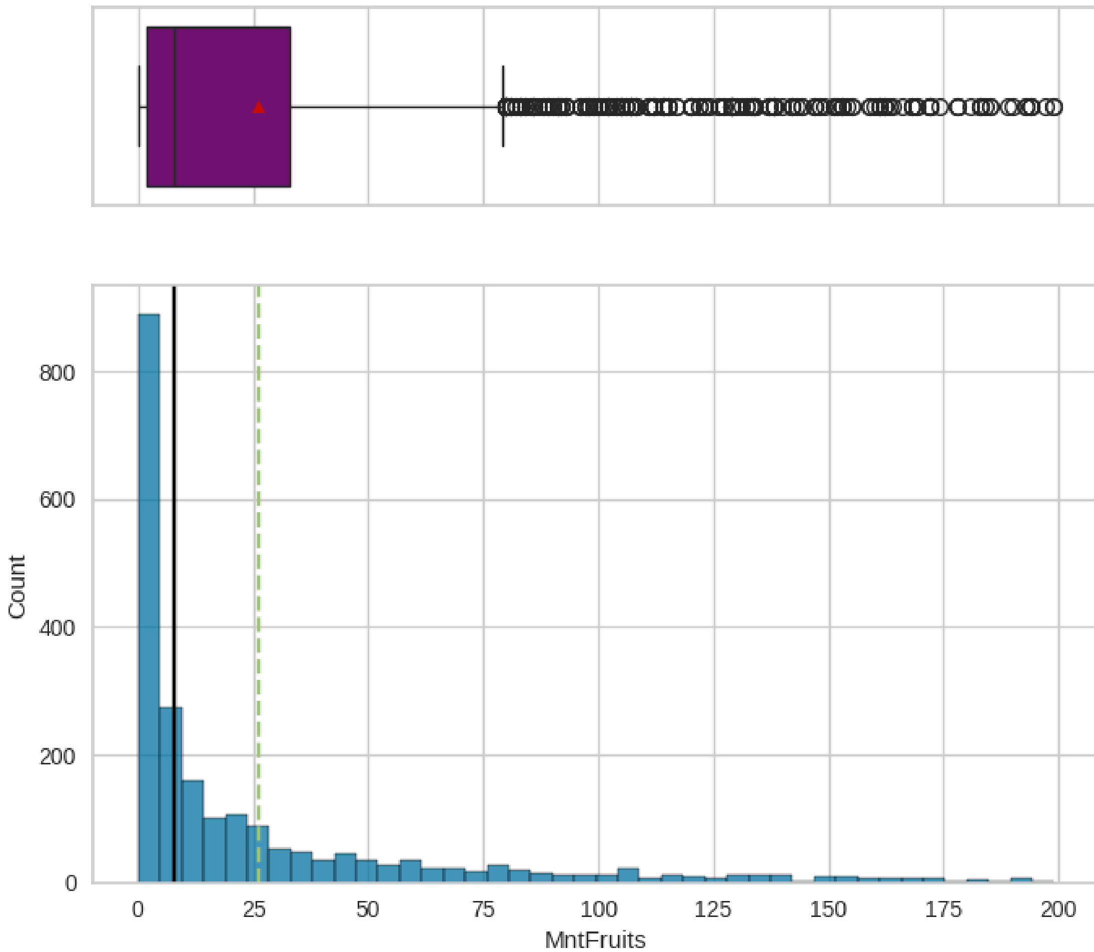
1200
1000
800
600
400
200
0
Count
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
Kidhome


1200
1000
800
600
400
200
0
Count
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
Teenhome

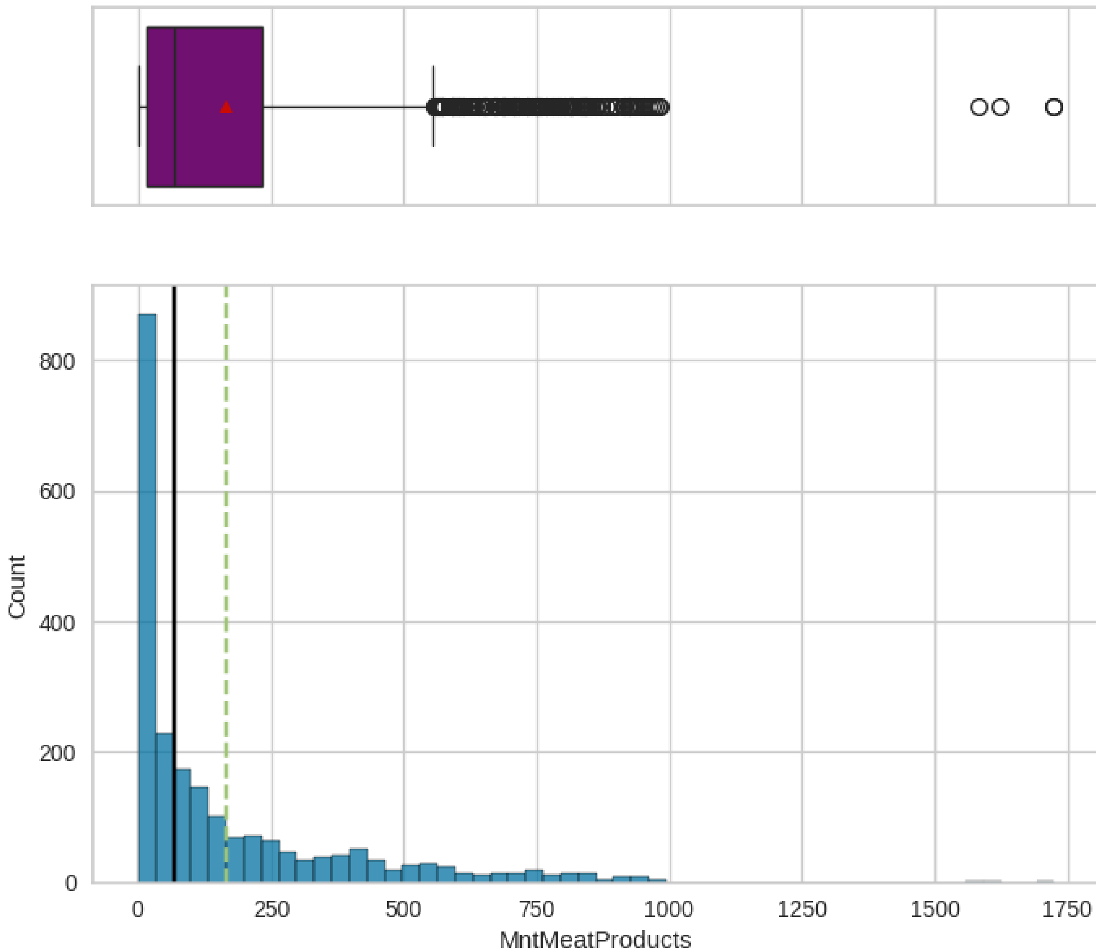

Count
200
175
150
125
100
75
50
25
0
0
20
40
60
80
100
Recency



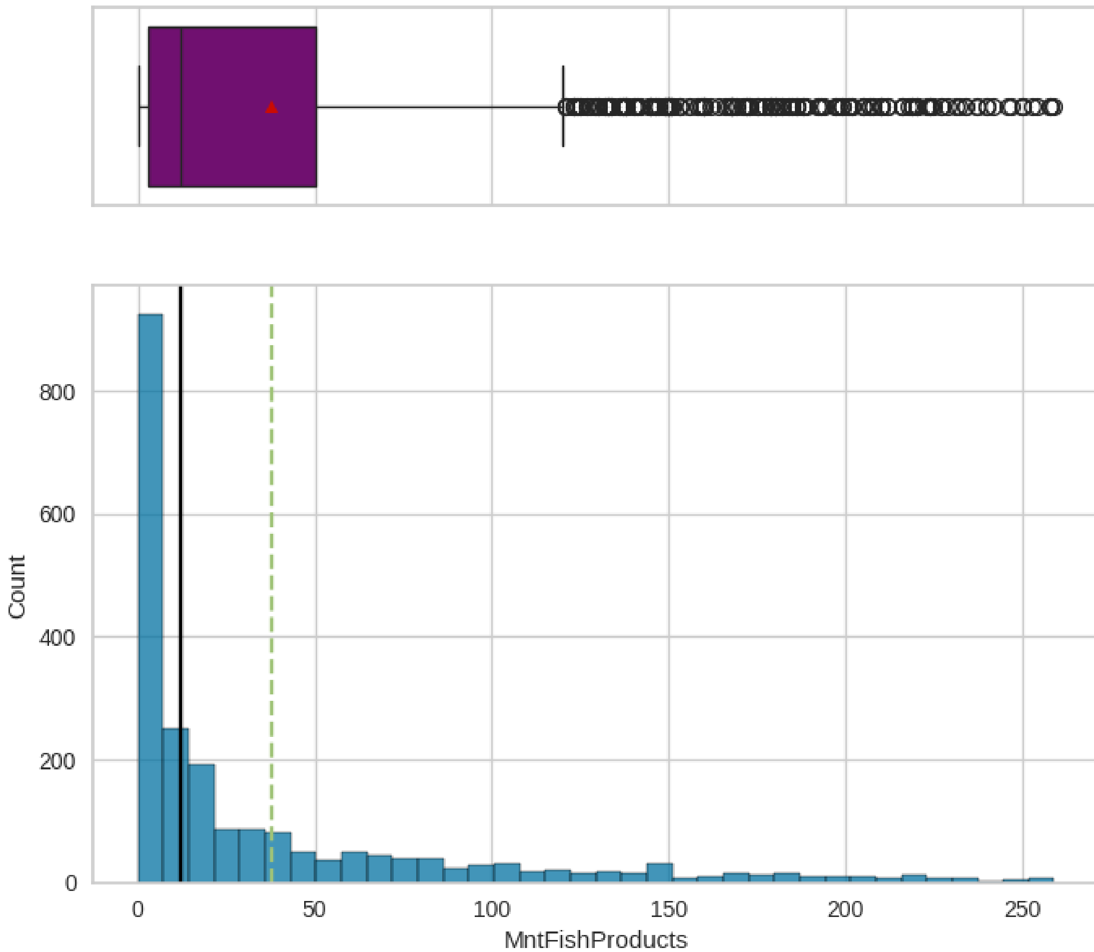
Count
800
700
600
500
400
300
200
100
0
0
200
400
600
800
1000
1200
1400
MntWines



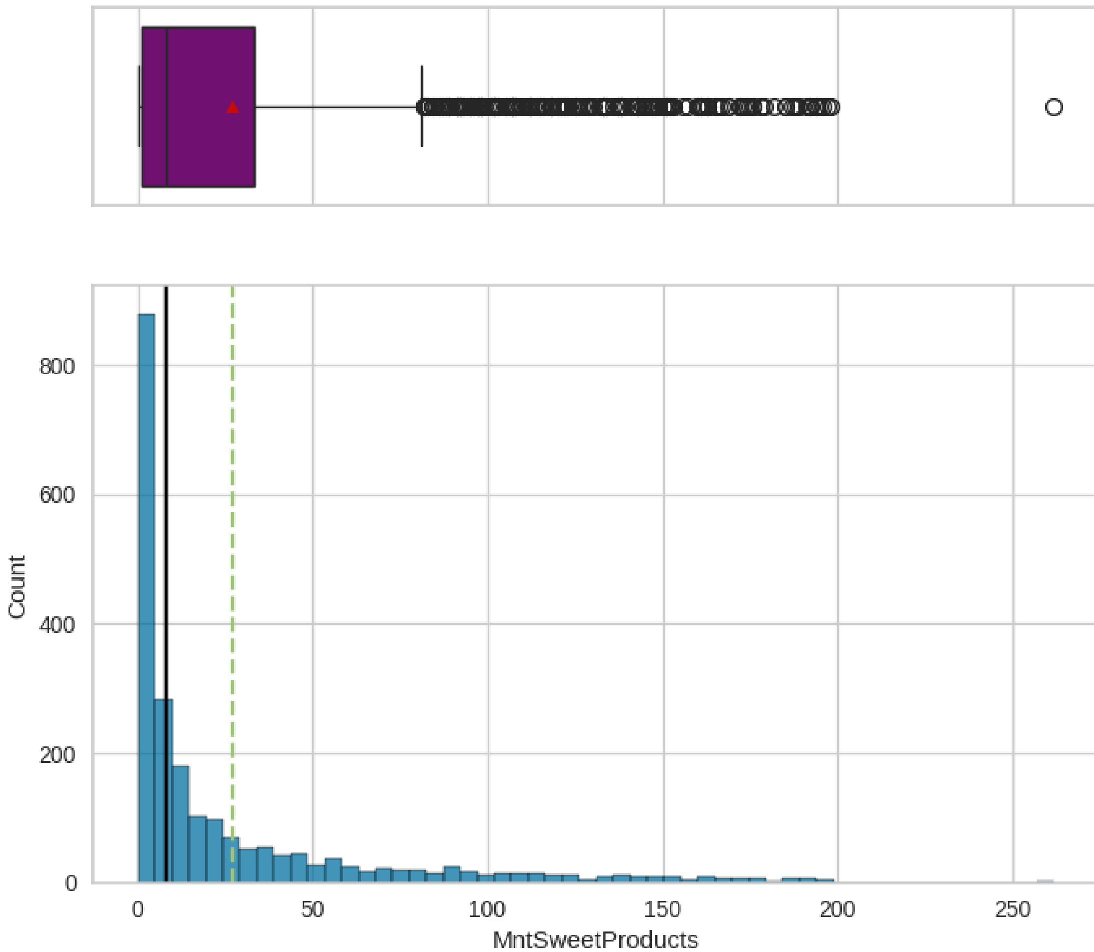
800
600
400
200
0
Count
0
25
50
75
100
125
150
175
200
MntFruits

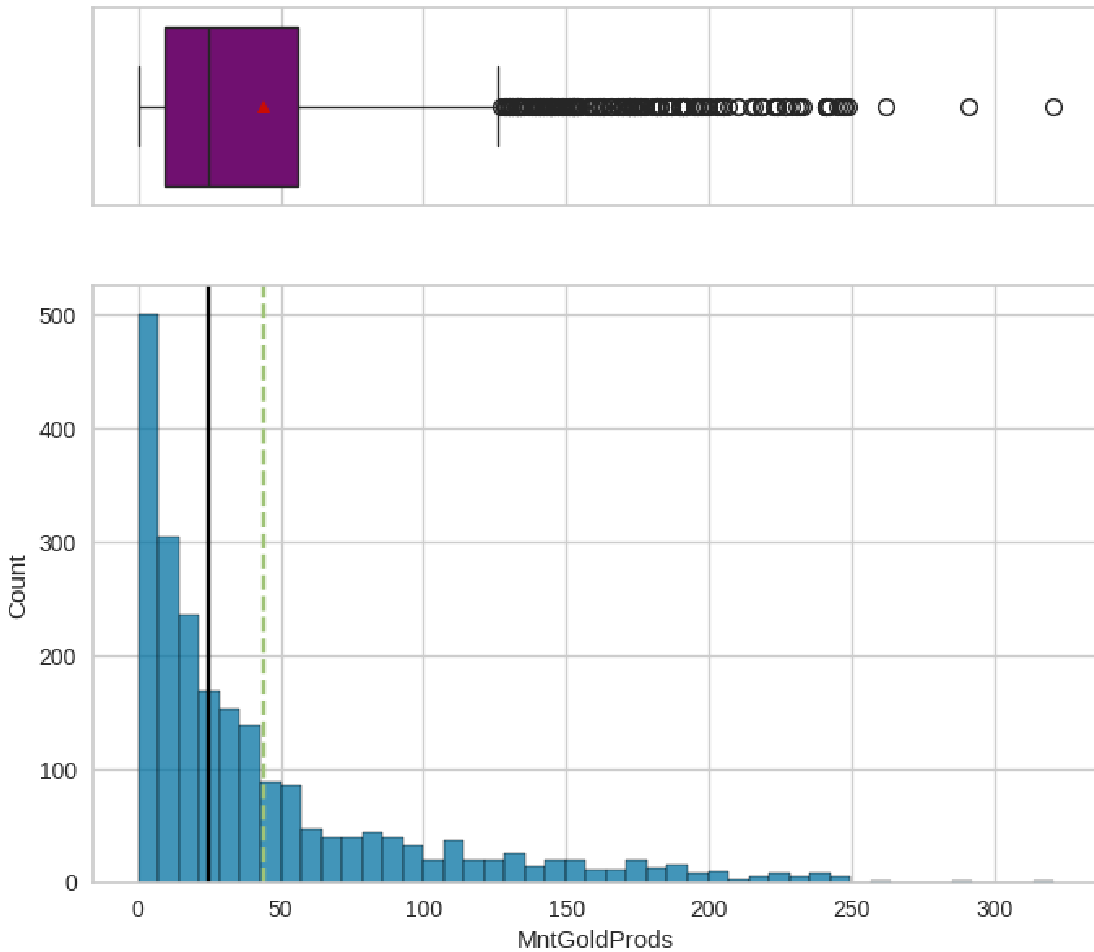


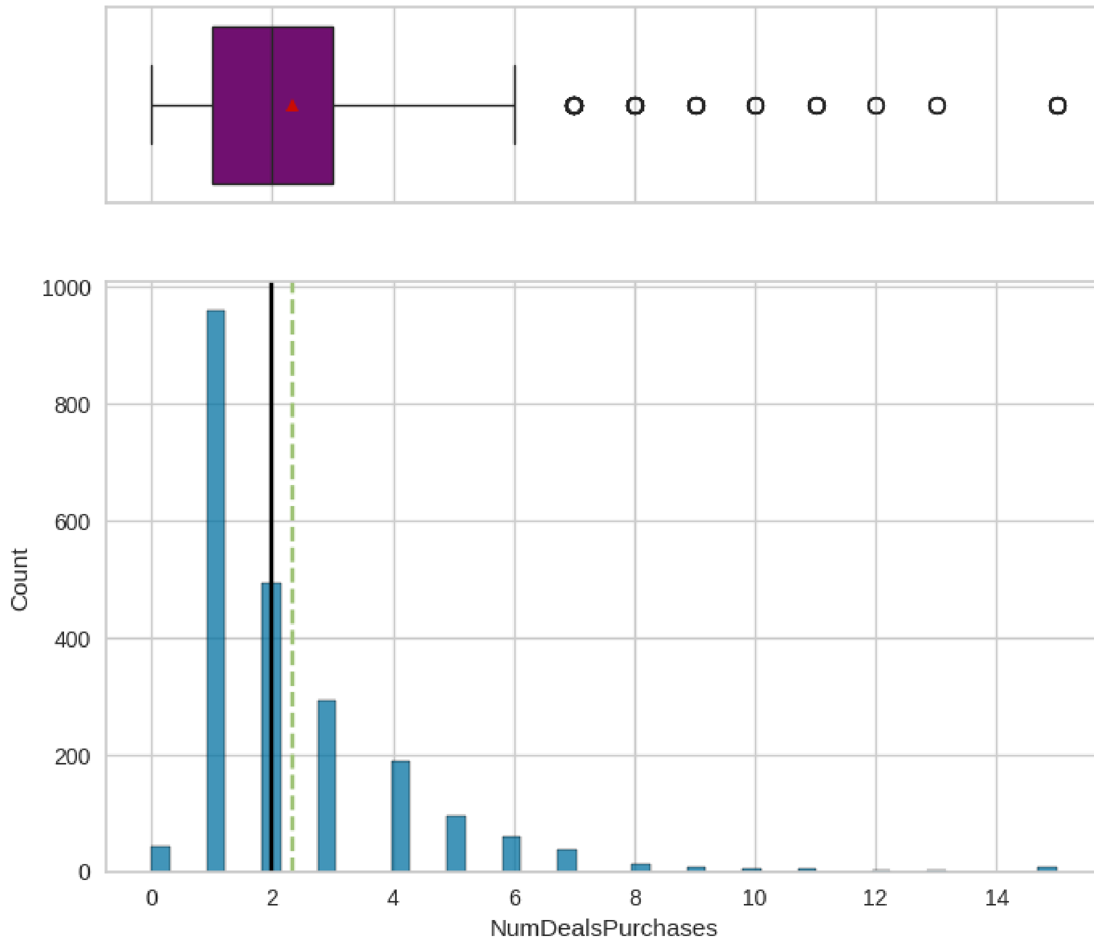
Count
800
600
400
200
0
0
250
500
750
1000
1250
1500
1750
MntMeatProducts

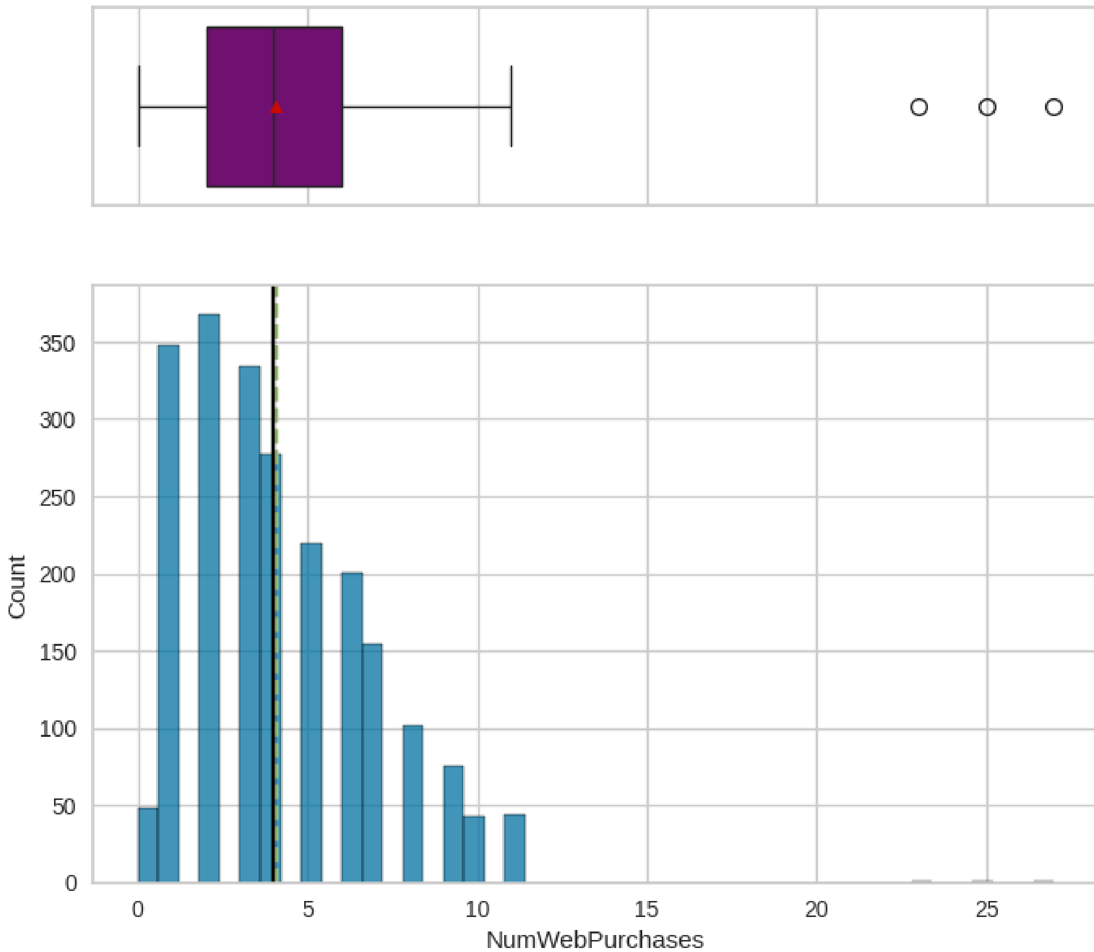


800
600
400
200
0
Count
0
50
100
150
200
250
MntFishProducts

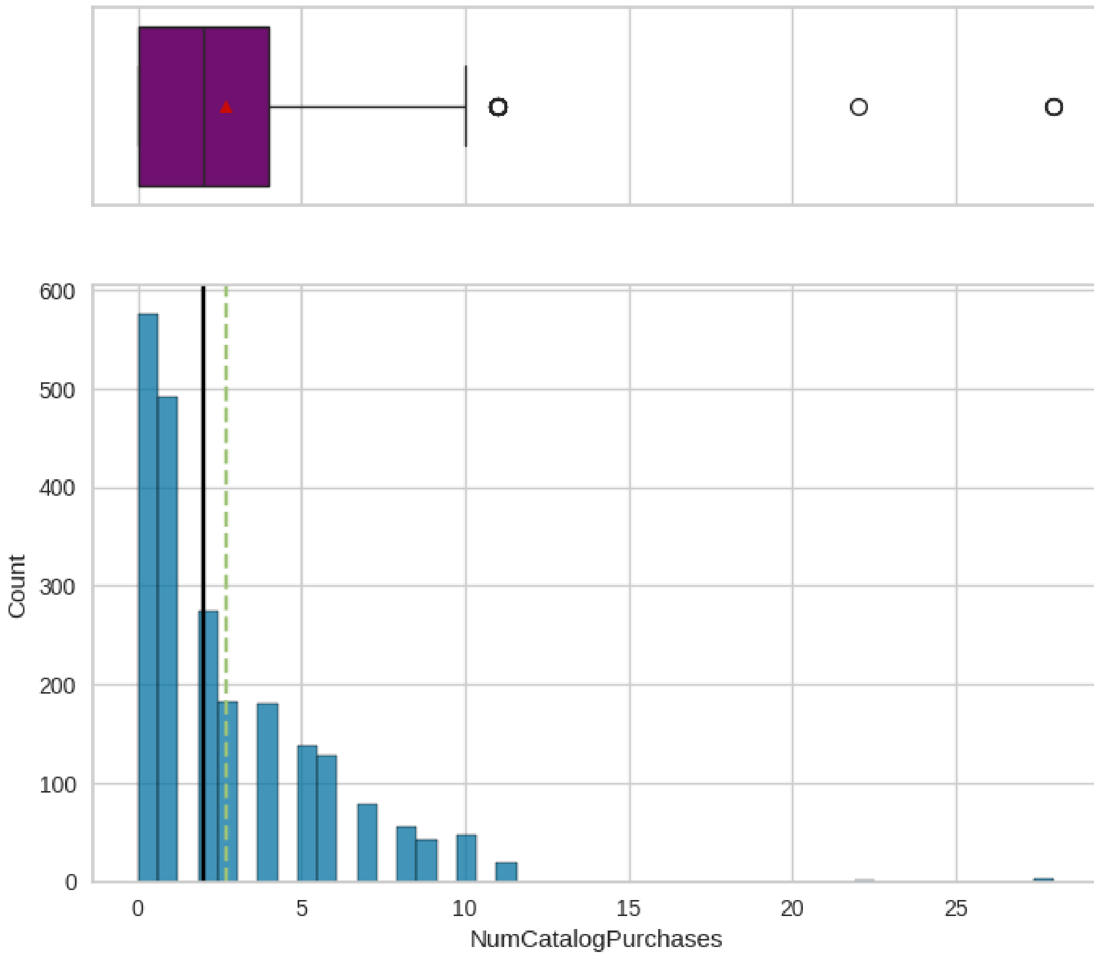

800
600
400
200
0
Count
0
50
100
150
200
250
MntSweetProducts


0
50
100
150
200
250
300
500
400
300
200
100
0
Count
MntGoldProds

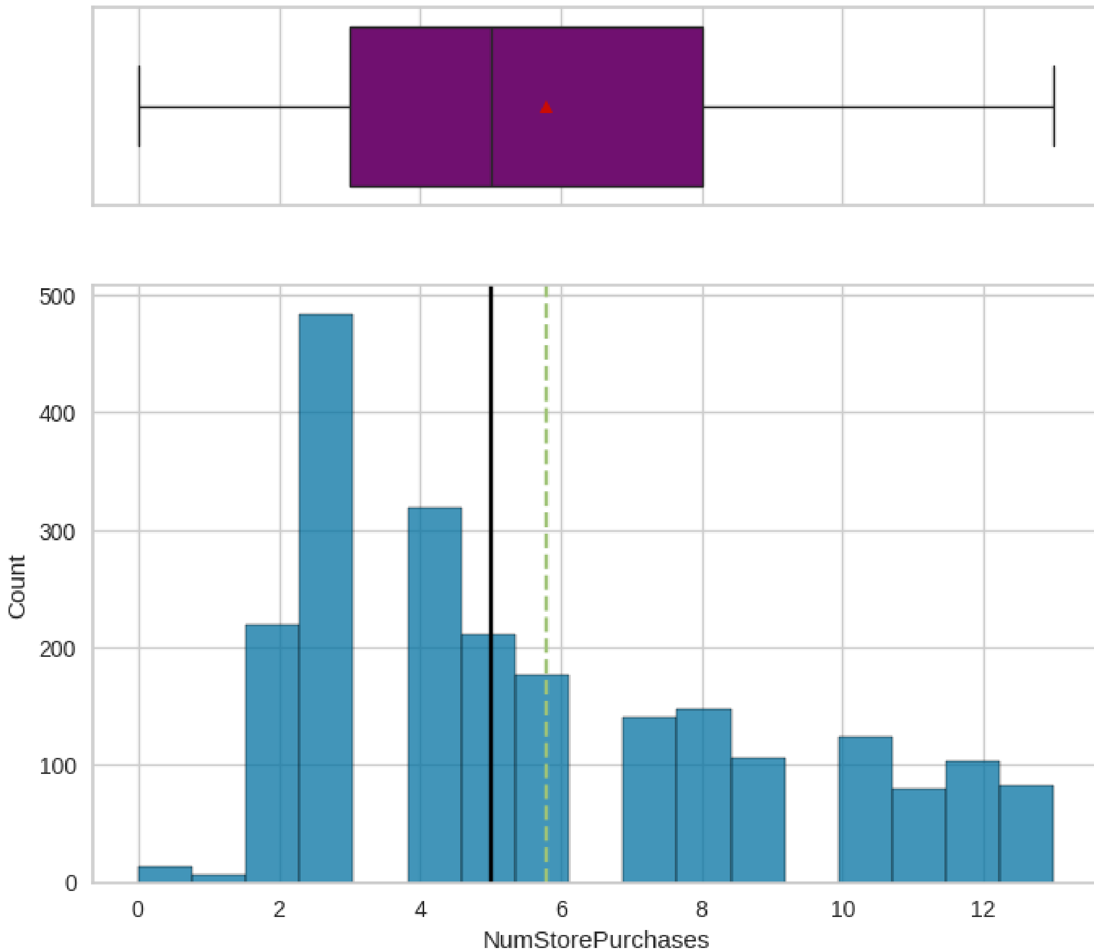

1000
800
600
400
200
0
Count
0
2
4
6
8
10
12
14
NumDealsPurchases



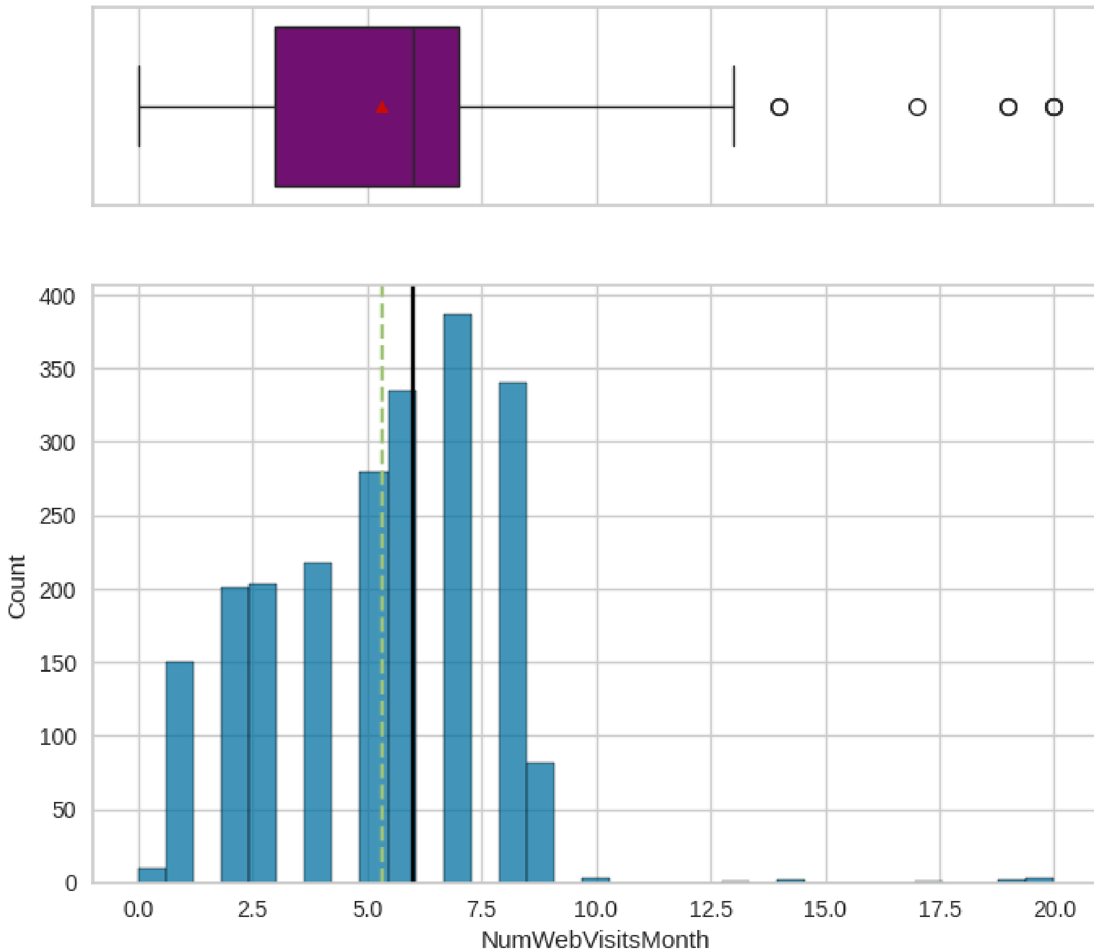
Count
350
300
250
200
150
100
50
0
0
5
10
15
20
25
NumWebPurchases

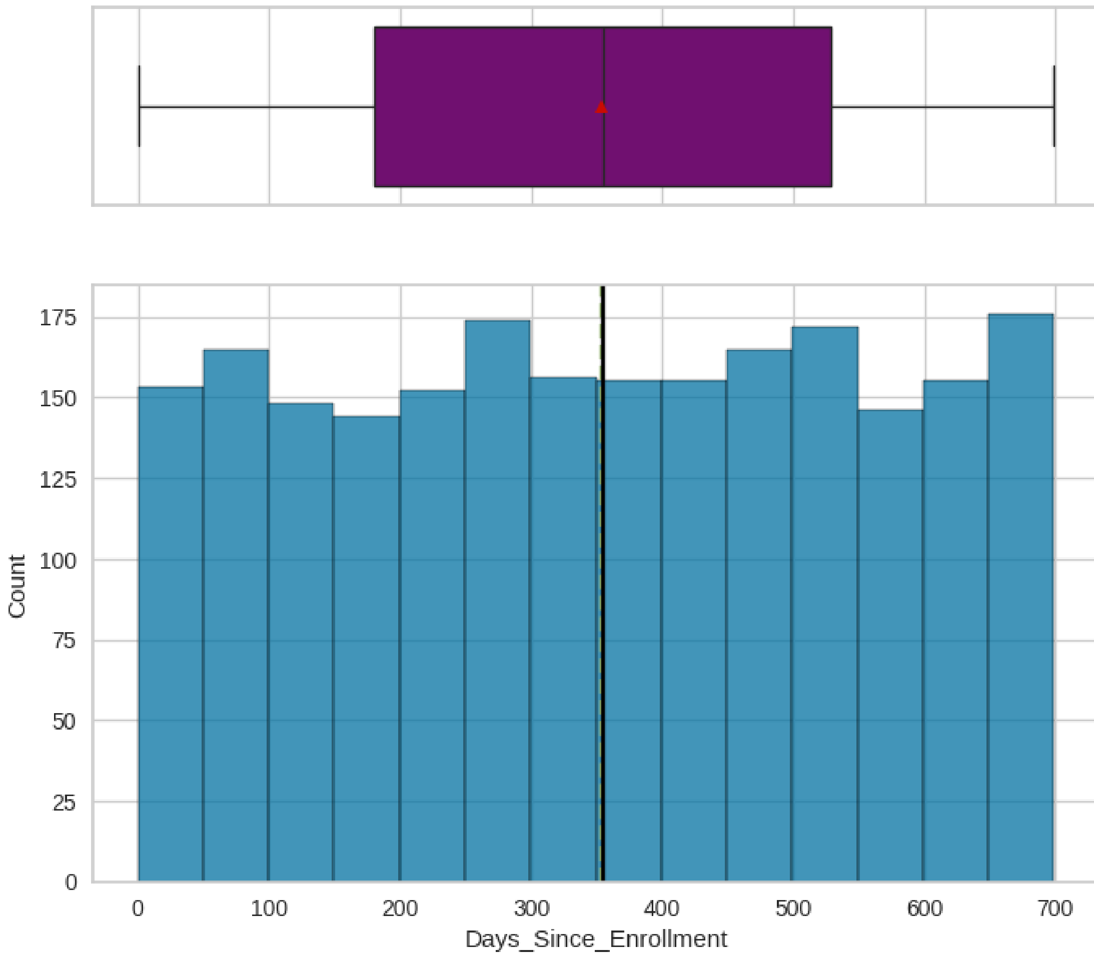


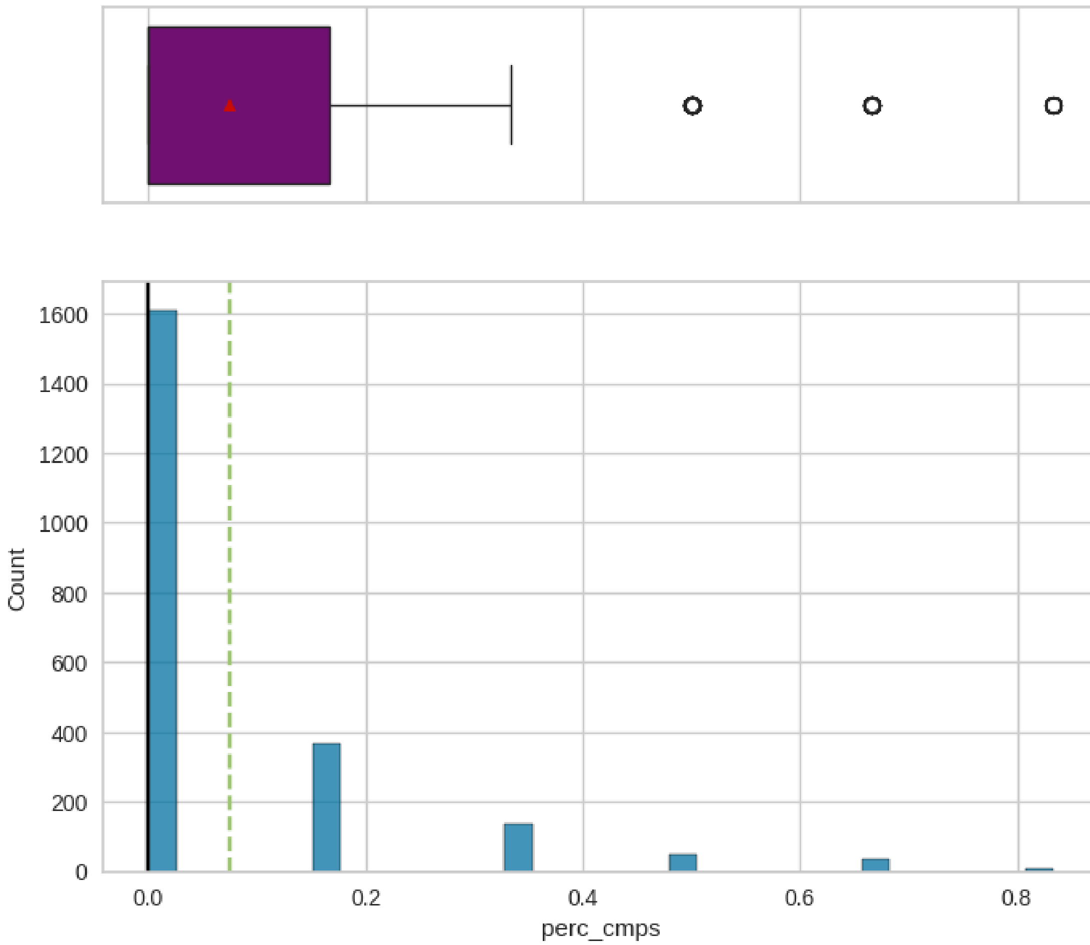
600
500
400
300
200
100
0
Count
0
5
10
15
20
25
NumCatalogPurchases



500
400
300
200
100
0
Count
0
2
4
6
8
10
12
NumStorePurchases


400
350
300
250
200
150
100
50
0
Count
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
NumWebVisitsMonth


175
150
125
100
75
50
25
0
Count
0
100
200
300
400
500
600
700
Days_Since_Enrollment



Observations: The Following Data Columns are:

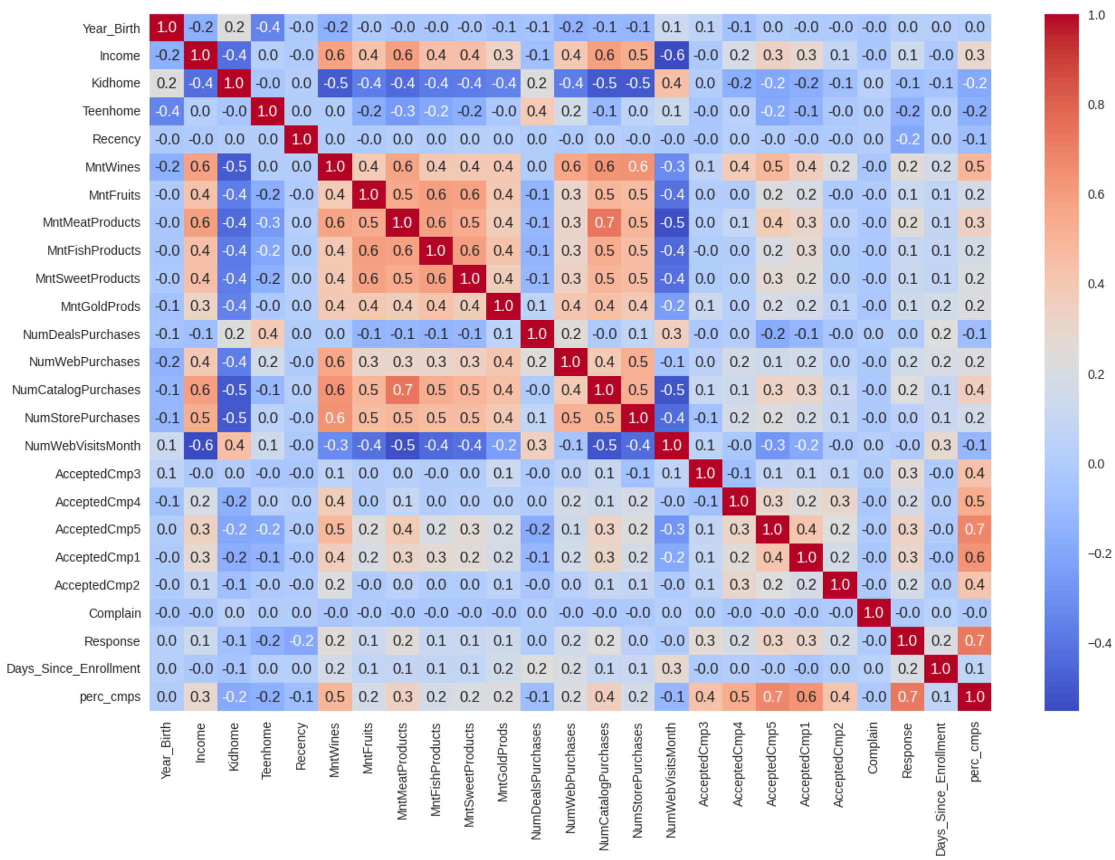
- Left Skewed:
  - numWebVisitsMonth has a slight left skewing but the outliers at 10 and above make it less left skewed.
- Right Skewed:
  - Kidhome
  - Teenhome
  - MntWines
  - MntFruits
  - MntMeatProducts
  - MntFishProducts
  - MntsweetProducts
  - MntGoldProds
  - NumDealsPurchases - with the mode being 1 but the median being 2
  - NumWebPurchases
  - NumCatalogPurchases
  - perc_cmps - most people did not act on the previous Campaigns.

- Normalized Distribution:
  - Income - Mostly normalized except a few outliers.
  - Recency - Relatively flat and even. This column is how many days since customers last purchase. Ideal would be to get this data to move more right skewed, so that many customers are returning to purchase often.
  - Year_Birth looks mostly Normalized, with a slight left skew
  - Days_Since_Enrollemnt is relatively flat
- Other:
  - The majority had an education level of Graduation, with PhD being the second highest, but significantly behind the Graduation level (1116 Grad to 481 PhD).
  - The majority were either married or together. There were a few alternative options that aren't standard (Absurd, Alone, YOLO). Will update these values to 'Single'

2.4.2 Bivariate Analysis

Question 6: Perform multivariate analysis to explore the relationsips between the variables.

[11]:
```python
#heat map to check the coorelation between variables
numeric_df = data_cleaned.select_dtypes(include=np.number)
#remove irrelivant columns so easier to see important data
numeric_df=numeric_df.drop('ID', axis=1)
numeric_df=numeric_df.drop('Z_Revenue', axis=1)
numeric_df=numeric_df.drop('Z_CostContact', axis=1)
#Get coorelation dataset
coor_df = numeric_df.select_dtypes(include=np.number).corr()

plt.figure(figsize = (15,10))

sns.heatmap(coor_df, annot = True, cmap = 'coolwarm',
fmt = ".1f",
xticklabels = coor_df.columns,
yticklabels = coor_df.columns
)

plt.show()
```



Observations:

- Columns with Highest Coorelation:
  - NumCatalogPurchases/NumStorePurchases and MntMeatProducts - Customers tend to buy their meat products from the catalog or in store
  - NumCatalogPurchases and Income
  - NumWebVisitsMonth and KidHome - People are more often ordering via web if they have a kid at home
  - MntWines has a high coorelation with all three types of purchase, as wel as participating in campaigns. Campaign 5 having the most coorelation with MtnWines
- Columns with Little to No Correlation to any other column:
  - Complain - No correlation showing with any other column
  - Recency - How recent customers have purchased doesn't appear to be influenced by the campaigns or other forms of purchase. Even the income appears to have no coorelation.
  - AcceptedCmp3
- Columns with Negative Correlation:
  - NumWebVisitsMonth is negatively coorelated with:
    * Income - While web purchases has a positive coorelation, the visits to the website are less when the customer has higher income.
    * MntMeatProducts

- MntFishProducts
- MntSweetProtducts
- MntFruits
- NumCatalogPurchases

Prep for Clustering: Scale, PCA, Outliers

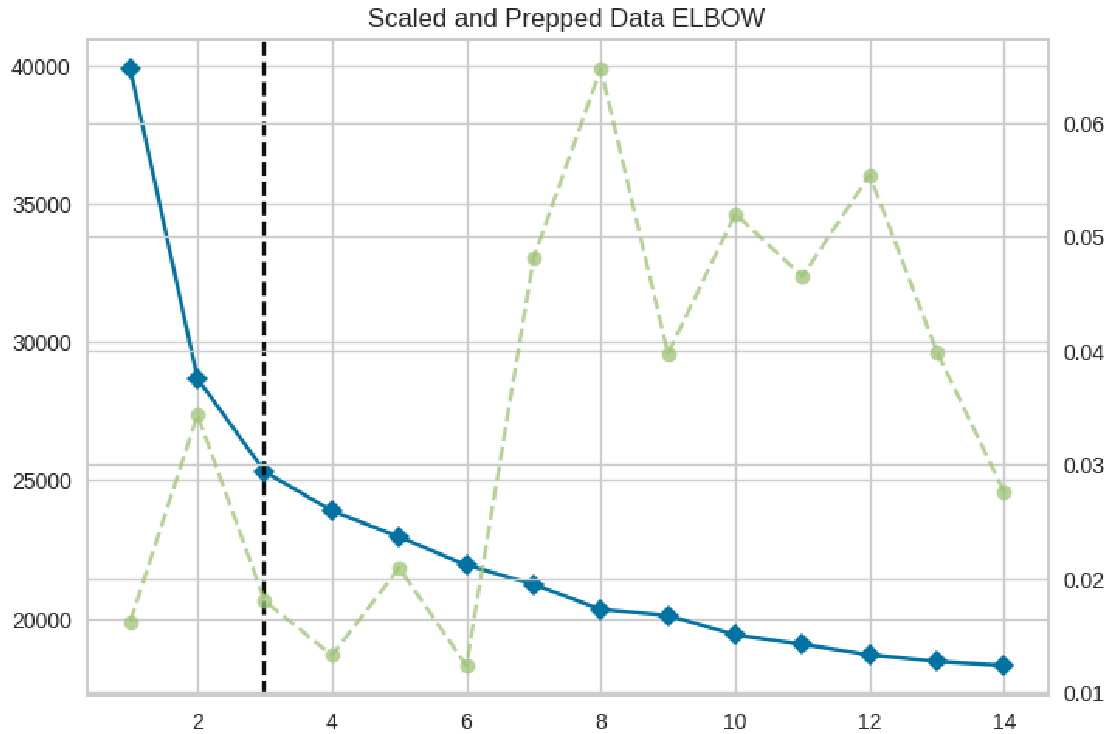
[29]:
```
df_cluster = data_cleaned.copy()

########## FIX OUTLIER

#The max income is $666,666 while the 75% is $68,522. There is only one person␣
↪who makes more than 200,000 for income - the one making 666,666
df_cluster.loc[df_cluster['Income'] >200000, 'Income'] = 200000
#update the small obscur answers to the proper label.
df_cluster.loc[df_cluster['Marital_Status'].isin(['Absurd','Alone','YOLO']),␣
↪'Marital_Status'] = 'Single'

########### REMOVE BINARY COLUMNS (As they affect clustering analysis, we will␣
↪keep the perc_cmp logic and use it)
columns_to_drop =␣
↪['ID','Dt_Customer','AcceptedCmp1','AcceptedCmp2','AcceptedCmp3','AcceptedCmp4','AcceptedCm
↪'Complain', 'Z_CostContact', 'Z_Revenue', 'Response']
df_cluster = df_cluster.drop(columns=columns_to_drop, axis=1)

# SCALE THE DATA
#Scale the Data
scaler = StandardScaler()
subset = df_cluster.select_dtypes(include=np.number)
subset_scaled = scaler.fit_transform(subset)
subset_scaled_df = pd.DataFrame(subset_scaled, columns = subset.columns)

# PCA

n = subset_scaled_df.shape[1]
pca = PCA(n_components = n, random_state = 1)
data_pca = pd.DataFrame(pca.fit_transform(subset_scaled_df ))
exp_var = (pca.explained_variance_ratio_)
print(exp_var)
```

```
[0.35598145 0.11307053 0.08085631 0.06312549 0.05438169 0.04693204
 0.0403548  0.0384733  0.03435759 0.02919104 0.02487121 0.02345447
 0.02187156 0.02161087 0.01624845 0.01331163 0.01219214 0.00971544]
```

2.5 K-means Clustering

Question 7 : Select the appropriate number of clusters using the elbow Plot. What do you think is the appropriate number of clusters?

[33]:
```
k_means_df = subset_scaled_df.copy()

model = KMeans(random_state=1)
visualizer = KElbowVisualizer(model, k=(1,15), timings = True)
visualizer.fit(k_means_df)
plt.title("Scaled and Prepped Data ELBOW")
plt.show()
```

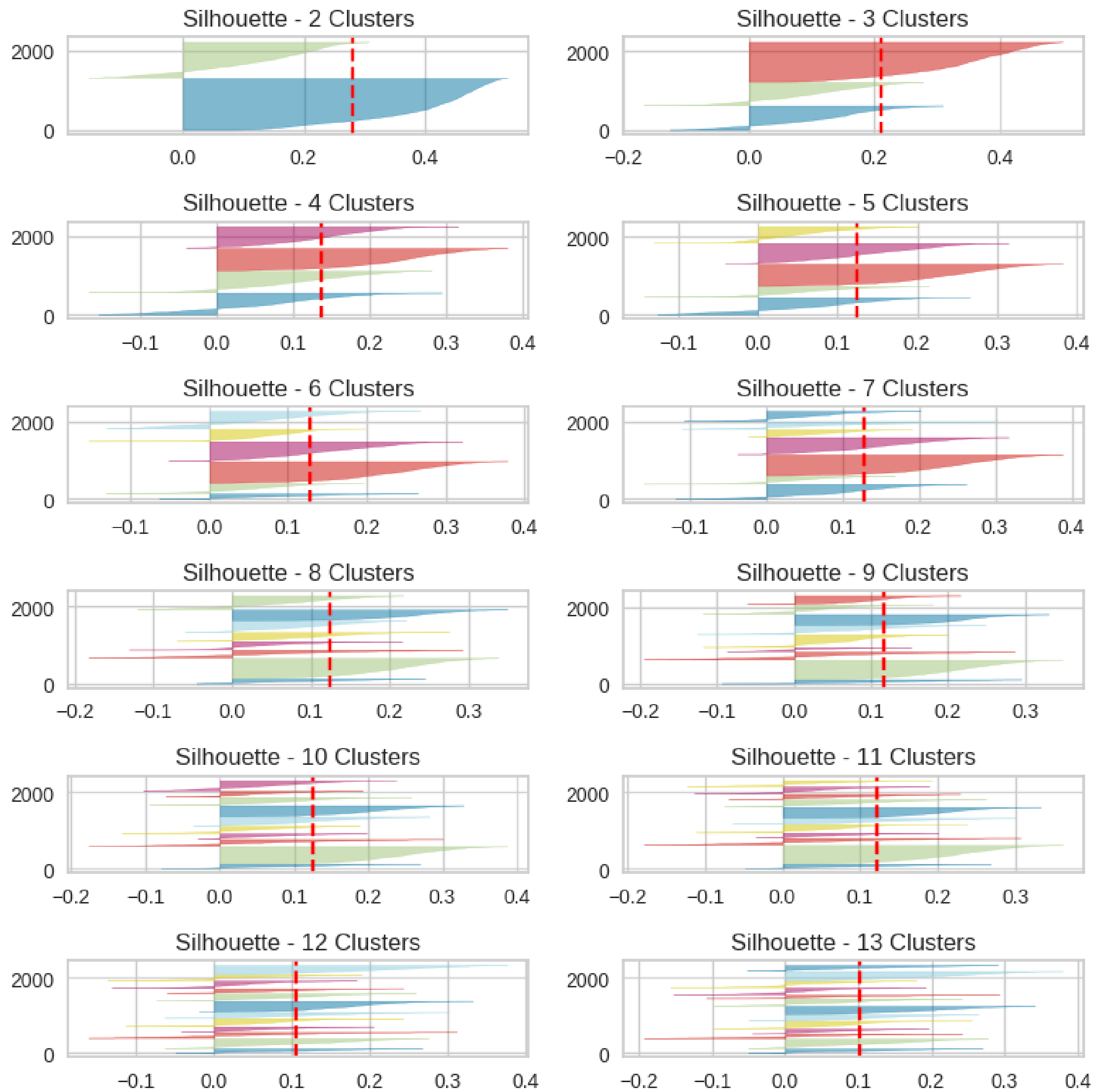


Observations: According to the ELBOW, the optimal k is 3.

Question 8 : finalize appropriate number of clusters by checking the silhoutte score as well. Is the answer different from the elbow plot?

[34]:
```
# Create subplots for each silhouette plot
fig, axes = plt.subplots(6, 2, figsize=(8, 8))
axes = axes.flatten()

# Loop through different numbers of clusters
for k in range(2,14):
    # Create KMeans instance with the current number of clusters
    model = KMeans(n_clusters=k, random_state=1)

    # Create SilhouetteVisualizer instance
```

```
    visualizer = SilhouetteVisualizer(model, colors='yellowbrick', ax=axes[k-2])

    # Fit the data to the visualizer
    visualizer.fit(k_means_df)

    # Set the title for the subplot
    axes[k-2].set_title(f'Silhouette - {k} Clusters')

# Adjust layout and show the plot
plt.tight_layout()
plt.show()
```



Observations: According to the Silhoutte scores, 2-3 clusters appears to be the most optimal. Comparing the data with the ELBOW, we will go with 3.

Question 9: Do a final fit with the appropriate number of clusters. How much total time does it take for the model to fit the data?

[35]:
```
import time
k_means_df = data_pca.copy()
k = 3 #Given the ELBOW and Sillhoutte score, 3 clusters appears to be the most␣
 ↪optimal.
kmeans = KMeans(n_clusters = k, random_state = 1, n_init = "auto")

start_time = time.time()#Begin Timer
kmeans.fit(k_means_df)
end_time = time.time()#End Timer

fit_time = end_time - start_time #Take difference and display
print(f"Total fit time: {fit_time:.4f} seconds.")
```

```
Total fit time: 0.0147 seconds.
```

Observations: Total time taken to fit the model using 3 clusters: 0.0147 seconds.

2.6 Hierarchical Clustering

Question 10: Calculate the cophnetic correlation for every combination of distance metrics and linkage. Which combination has the highest cophnetic correlation?

[36]:
```
hc_df = subset_scaled_df.copy()

# List of distance metrics
distance_metrics = ["euclidean", "chebyshev", "mahalanobis", "cityblock"]

# List of linkage methods
linkage_methods = ["single", "complete", "average", "weighted"]

high_cophenet_corr = 0
high_dm_lm = [0, 0]

for dm in distance_metrics:
    for lm in linkage_methods:
        Z = linkage(hc_df, metric = dm, method = lm)
        c, coph_dists = cophenet(Z, pdist(hc_df))
        print(
            "Cophenetic correlation for {} distance and {} linkage is {}.".
 ↪format(
                dm.capitalize(), lm, c
            )
        )
```

```
        if high_cophenet_corr < c:
            high_cophenet_corr = c
            high_dm_lm[0] = dm
            high_dm_lm[1] = lm

# Printing the combination of distance metric and linkage method with the␣
↪highest cophenetic correlation
print('*'*100)
print(
    "Highest cophenetic correlation is {}, which is obtained with {} distance␣
↪and {} linkage.".format(
        high_cophenet_corr, high_dm_lm[0].capitalize(), high_dm_lm[1]
    )
)
```

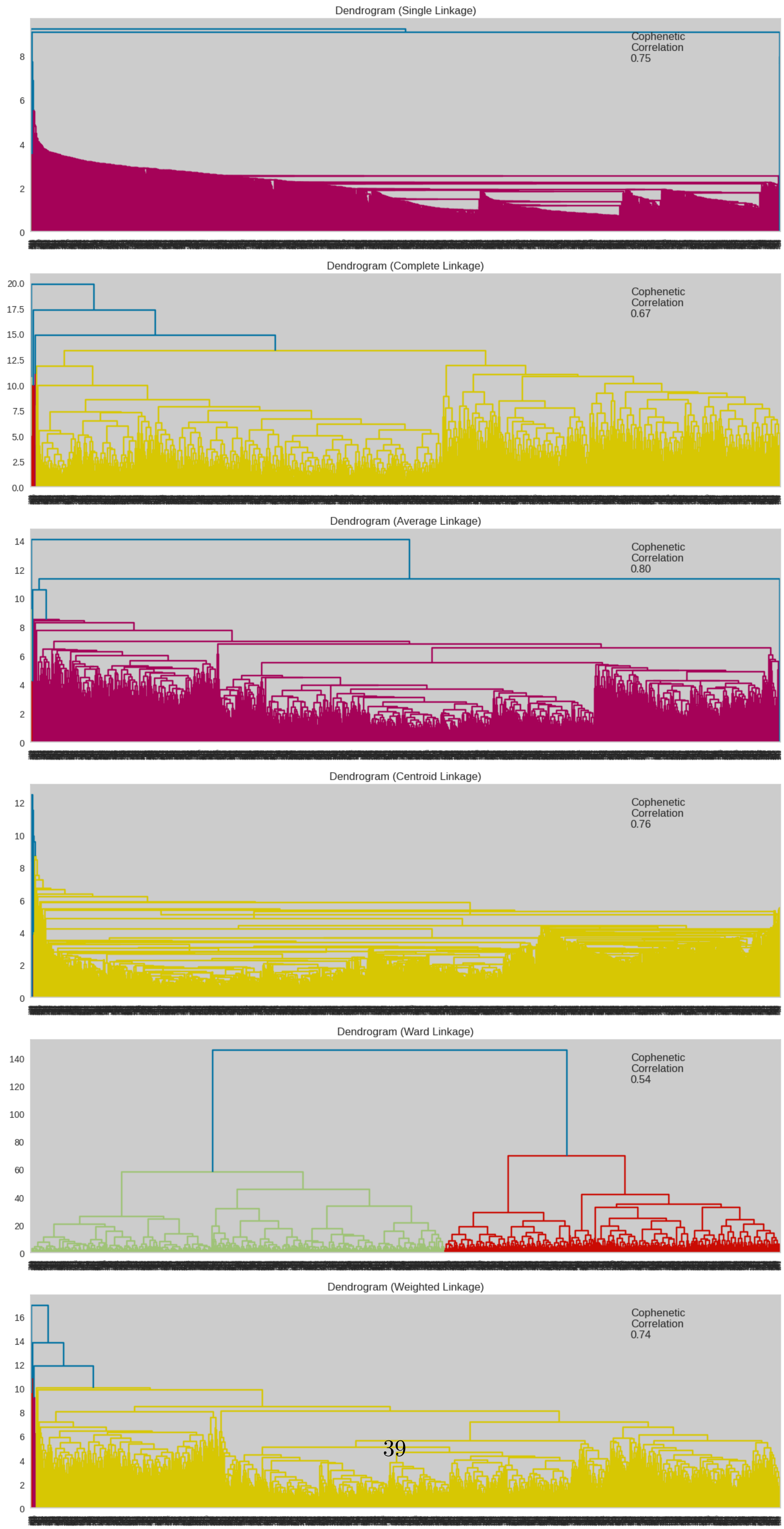
```
Cophenetic correlation for Euclidean distance and single linkage is
0.7486520152679234.
Cophenetic correlation for Euclidean distance and complete linkage is
0.6733473590391738.
Cophenetic correlation for Euclidean distance and average linkage is
0.8037736960257093.
Cophenetic correlation for Euclidean distance and weighted linkage is
0.7355257727477994.
Cophenetic correlation for Chebyshev distance and single linkage is
0.5483907030407056.
Cophenetic correlation for Chebyshev distance and complete linkage is
0.5383635546415438.
Cophenetic correlation for Chebyshev distance and average linkage is
0.751378468471372.
Cophenetic correlation for Chebyshev distance and weighted linkage is
0.598246301870843.
Cophenetic correlation for Mahalanobis distance and single linkage is
0.7262777027220269.
Cophenetic correlation for Mahalanobis distance and complete linkage is
0.4219695739486921.
Cophenetic correlation for Mahalanobis distance and average linkage is
0.7495314109195897.
Cophenetic correlation for Mahalanobis distance and weighted linkage is
0.6405539837202854.
Cophenetic correlation for Cityblock distance and single linkage is
0.7847856542533527.
Cophenetic correlation for Cityblock distance and complete linkage is
0.5012247691561919.
Cophenetic correlation for Cityblock distance and average linkage is
0.7714796440003644.
Cophenetic correlation for Cityblock distance and weighted linkage is
0.603354336769852.
```

```
********************************************************************************
********************
Highest cophenetic correlation is 0.8037736960257093, which is obtained with
Euclidean distance and average linkage.
```

Observations: Highest cophenetic correlation is 0.8037736960257093, which is obtained with Euclidean distance and average linkage.

Question 11: plot the dendogram for every linkage method with "Euclidean" distance only. What should be the appropriate linkage according to the plot?

[37]:
```
# List of linkage methods
linkage_methods = ["single", "complete", "average", "centroid", "ward",␣
 ↪"weighted"]

# Lists to save results of cophenetic correlation calculation
compare_cols = ["Linkage", "Cophenetic Coefficient"]
compare = []

# To create a subplot image
fig, axs = plt.subplots(len(linkage_methods), 1, figsize = (15, 30))

# We will enumerate through the list of linkage methods above
# For each linkage method, we will plot the dendrogram and calculate the␣
 ↪cophenetic correlation
for i, method in enumerate(linkage_methods):
    Z = linkage(hc_df, metric = "euclidean", method = method)

    dendrogram(Z, ax = axs[i])
    axs[i].set_title(f"Dendrogram ({method.capitalize()} Linkage)")

    coph_corr, coph_dist = cophenet(Z, pdist(hc_df))
    axs[i].annotate(
        f"Cophenetic\nCorrelation\n{coph_corr:0.2f}",
        (0.80, 0.80),
        xycoords="axes fraction",
    )

    compare.append([method, coph_corr])
```


Dendrogram (Single Linkage)
Cophenetic Correlation 0.75
Dendrogram (Complete Linkage)
Cophenetic Correlation 0.67
Dendrogram (Average Linkage)
Cophenetic Correlation 0.80
Dendrogram (Centroid Linkage)
Cophenetic Correlation 0.76
Dendrogram (Ward Linkage)
Cophenetic Correlation 0.54
Dendrogram (Weighted Linkage)
Cophenetic Correlation 0.74

[38]:
```python
df_cc = pd.DataFrame(compare, columns = compare_cols)

df_cc = df_cc.sort_values(by = "Cophenetic Coefficient")
df_cc
```

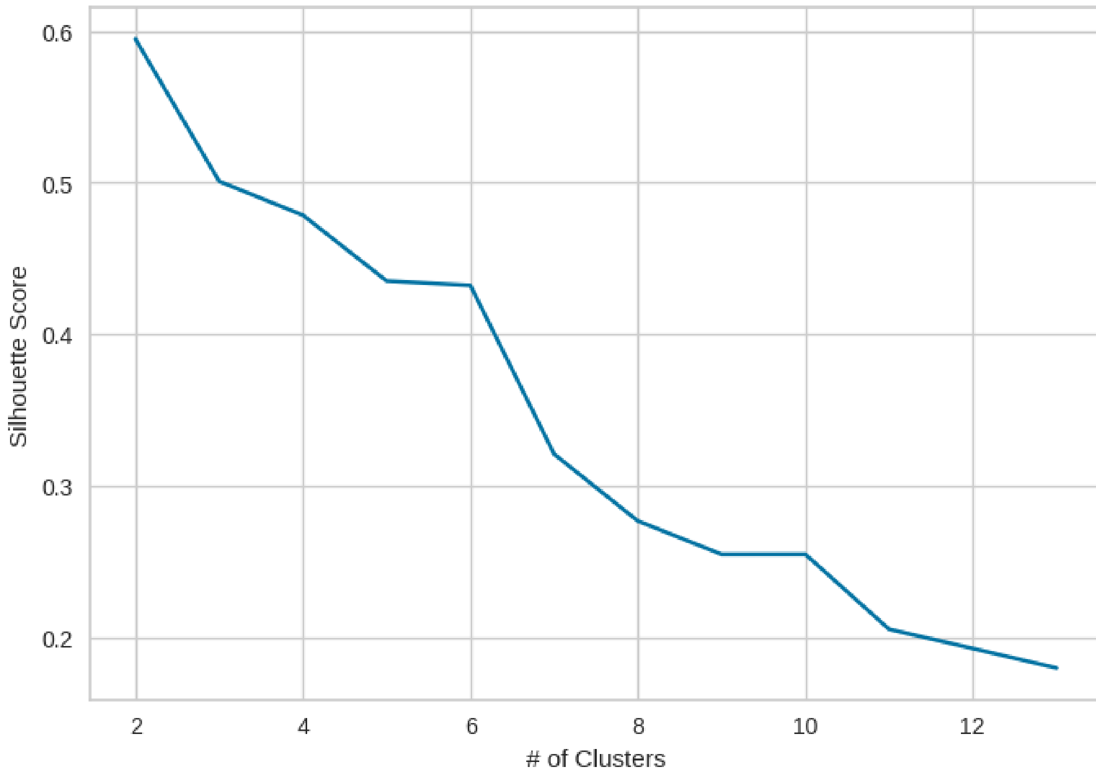
[38]:

	Linkage	Cophenetic Coefficient
4	ward	0.536756
1	complete	0.673347
5	weighted	0.735526
0	single	0.748652
3	centroid	0.764129
2	average	0.803774

Observations: Use 3 clusters, Euclidean Distance, and average linkage.

Question 12: Check the silhoutte score for the hierchial clustering. What should be the appropriate number of clusters according to this plot?

[39]:
```python
### SCALED AND REDUCED DATA
X = hc_df.copy()
# Create subplots for each silhouette plot

sc={}
# Loop through different numbers of clusters
for k in range(2,14):
    # Create Agglomerative hierchial instance with the current number of␣
 ↪clusters
    model = AgglomerativeClustering(n_clusters = k, metric = "euclidean",␣
 ↪linkage = "average")
    lbls = model.fit_predict(X)
    sc[k] = silhouette_score(X, lbls)
    # Set the title for the subplot

plt.figure()
plt.plot(list(sc.keys()), list(sc.values()), 'bx-')
plt.xlabel("# of Clusters")
plt.ylabel("Silhouette Score")
plt.show()
```



Observations: After 3 clusters, the silhoute score goes flattens before dropping more after 6. 3 appears to still be the best choice.

Question 13: Fit the Hierarchial clustering model with the appropriate parameters finalized above. How much time does it take to fit the model?

[40]:
```
# Create model
HCmodel = AgglomerativeClustering(n_clusters = 3, metric = "euclidean", linkage␣
↪= "average")

start_time = time.time() #start timer
HCmodel.fit(hc_df)
end_time = time.time() #end timer

#Calculate difference and print how long it took to fit.
fit_time = end_time - start_time
print(f"Total fit time: {fit_time:.4f} seconds.")
```

```
Total fit time: 0.2164 seconds.
```

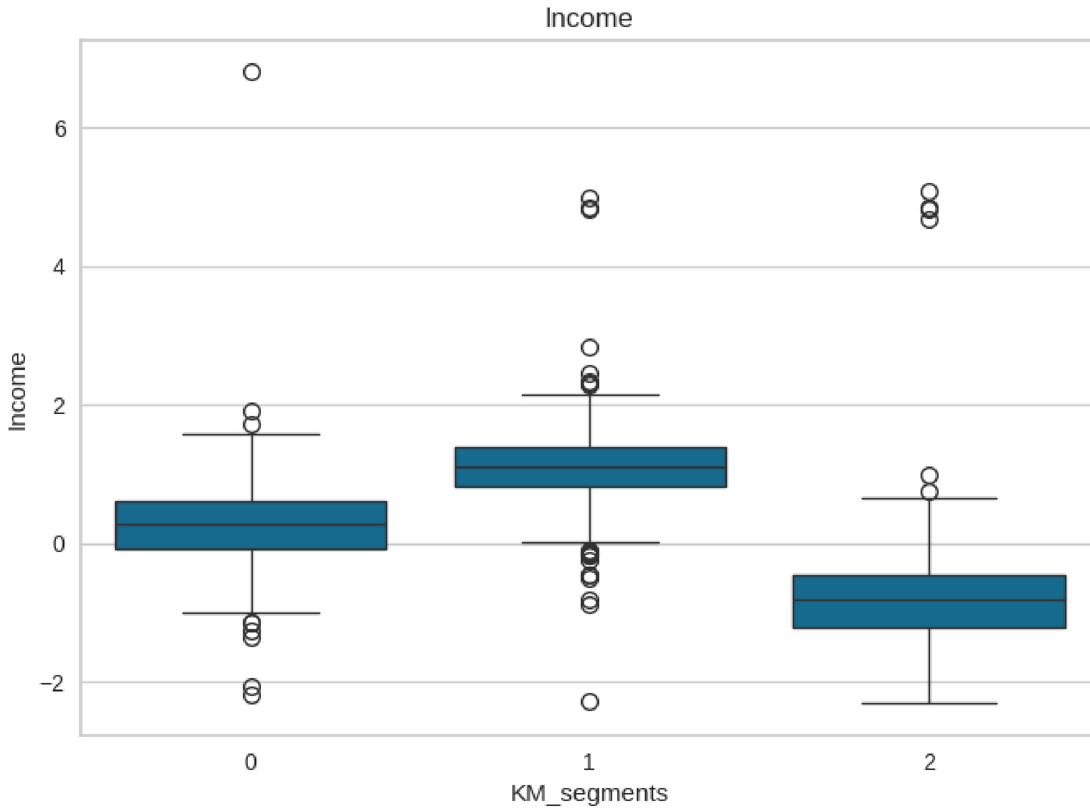
Observations: The final fit took 0.2164 seconds.

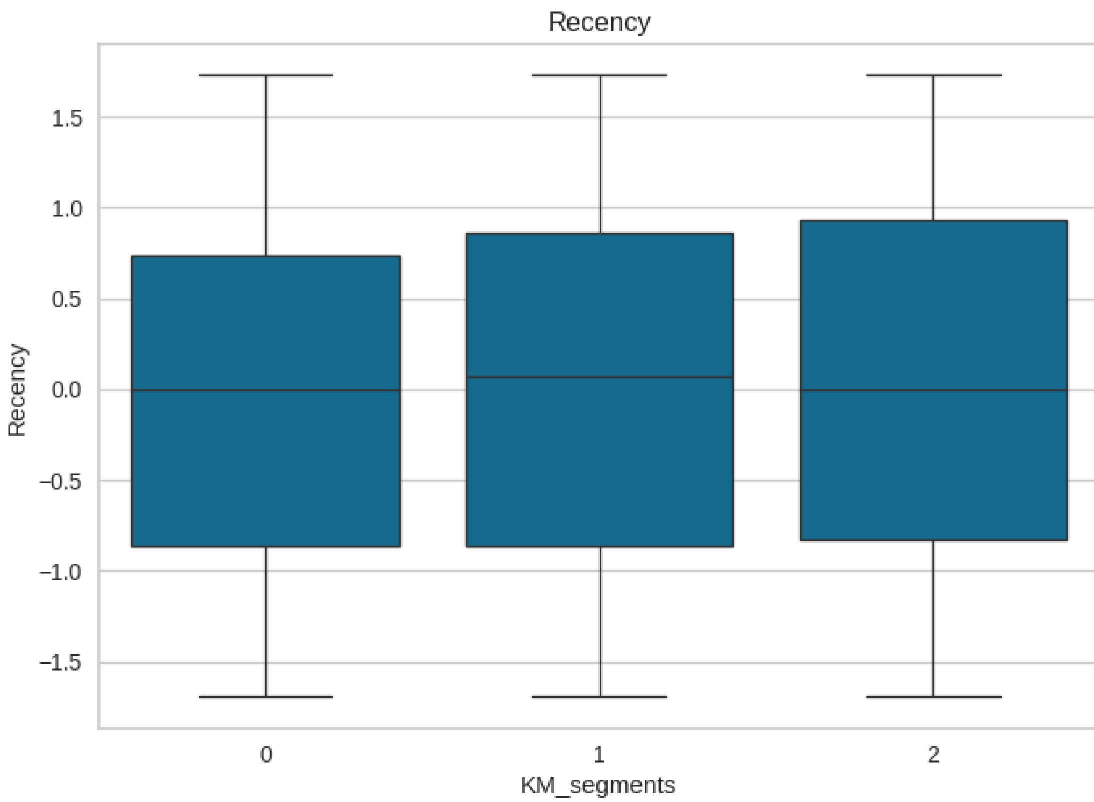
2.7 Cluster Profiling and Comparison

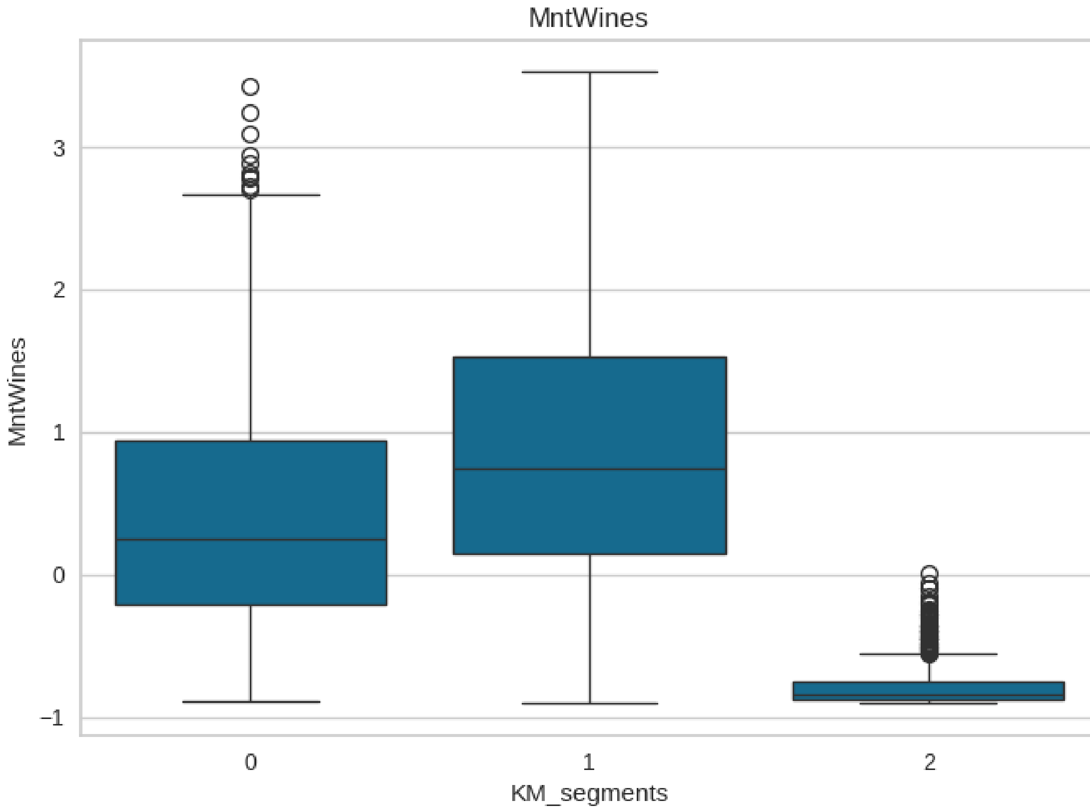
2.7.1 K-Means Clustering vs Hierarchical Clustering Comparison

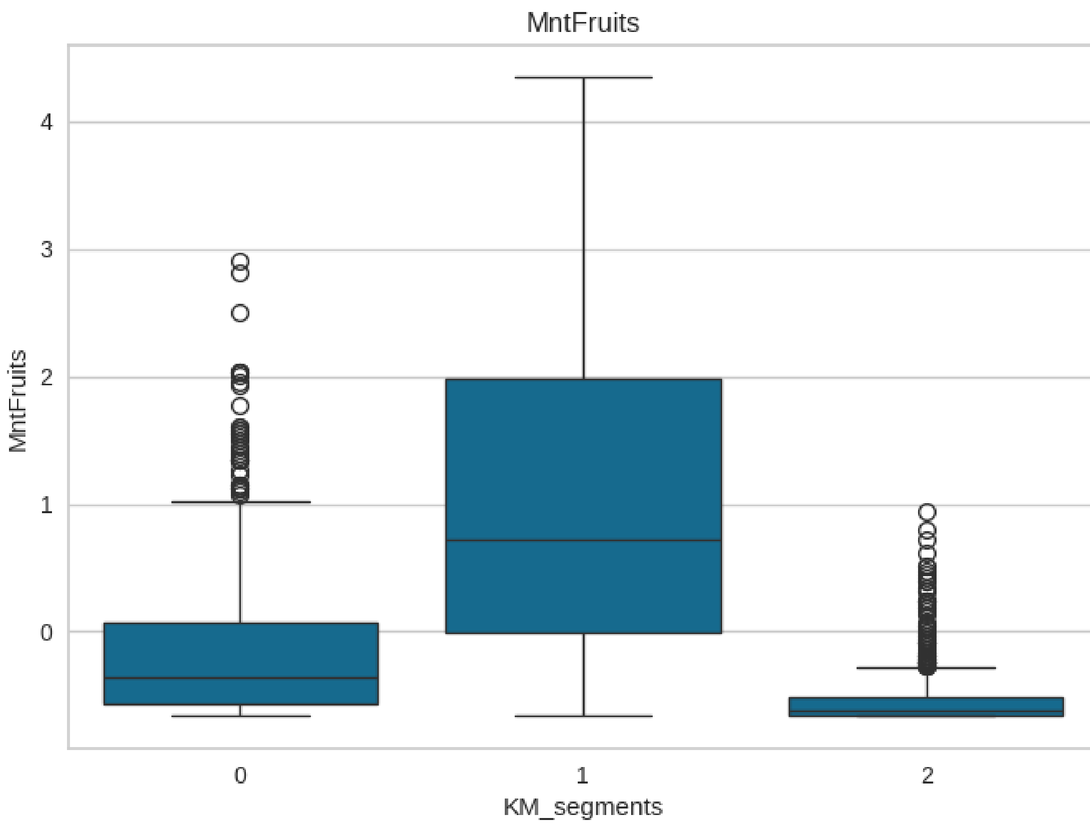
Question 14: Perform and compare Cluster profiling on both algorithms using boxplots. Based on the all the observaions Which one of them provides better clustering?

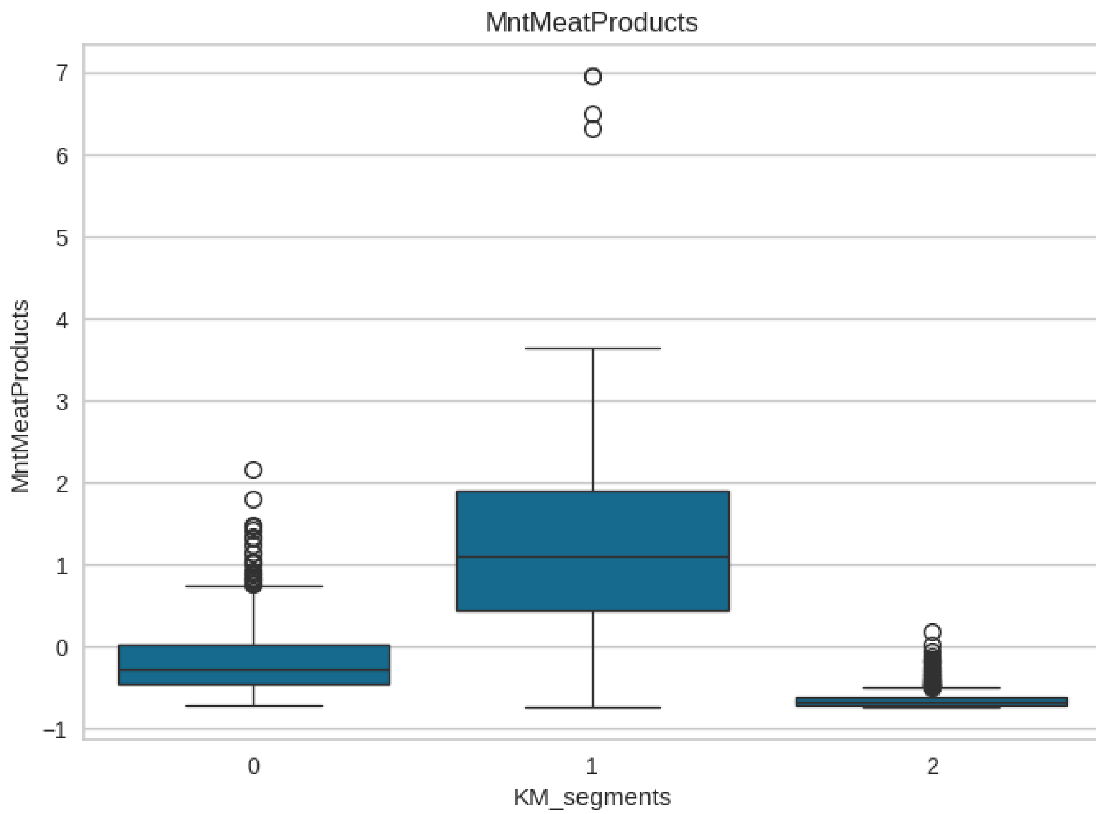
[74]:
```
#KMEANS CLUSTER ANALYSIS

df1 = subset_scaled_df.copy()
df1["KM_segments"] = kmeans.labels_
df_cluster['KM_segments']=kmeans.labels_
#Group by the clusters from kmeans
km_cluster_profile = df1.groupby("KM_segments").mean(numeric_only = True)

#Look at Income
km_cluster_profile["count_in_each_segment"] = (
    df1.groupby("KM_segments")["Income"].count().values
)
cols_visualise=['Income', 'Recency', 'MntWines','MntFruits','MntMeatProducts'
 ␣
 ↪,'MntFishProducts','MntSweetProducts','MntGoldProds','NumDealsPurchases','NumWebPurchases',
              ,'NumStorePurchases','NumWebVisitsMonth']


num_cols = cols_visualise + ['KM_segments']

mean = df1[num_cols].groupby('KM_segments').mean()
median = df1[num_cols].groupby('KM_segments').median()

df_kmeans = pd.concat([mean, median], axis = 0)
df_kmeans.index = ['group_0 Mean', 'group_1 Mean', 'group_2 Mean'
              , 'group_0 Median', 'group_1 Median' , 'group_2 Median']


for col in cols_visualise:
    sns.boxplot(x = 'KM_segments', y = col, data = df1)
    plt.title(col)
    plt.show()



km_cluster_profile.style.highlight_max(color = "green", axis = 0)
```

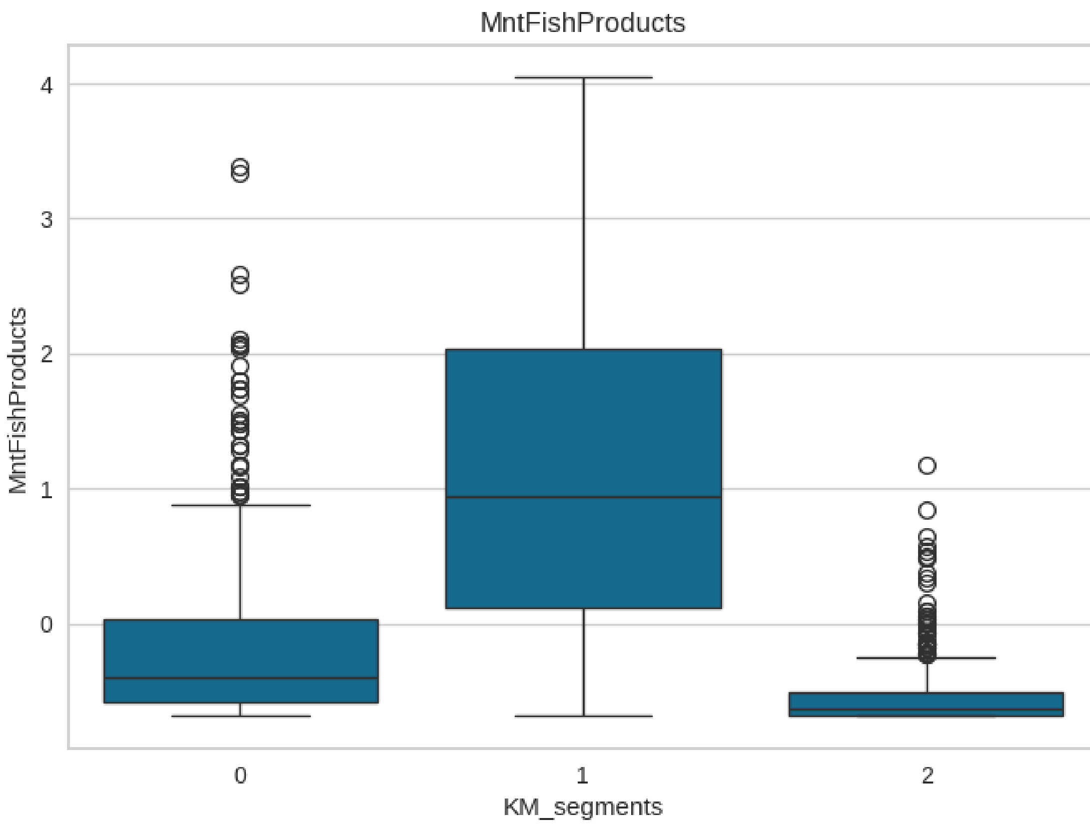

Income
6
4
2
0
−2
Income
0
1
2
KM_segments

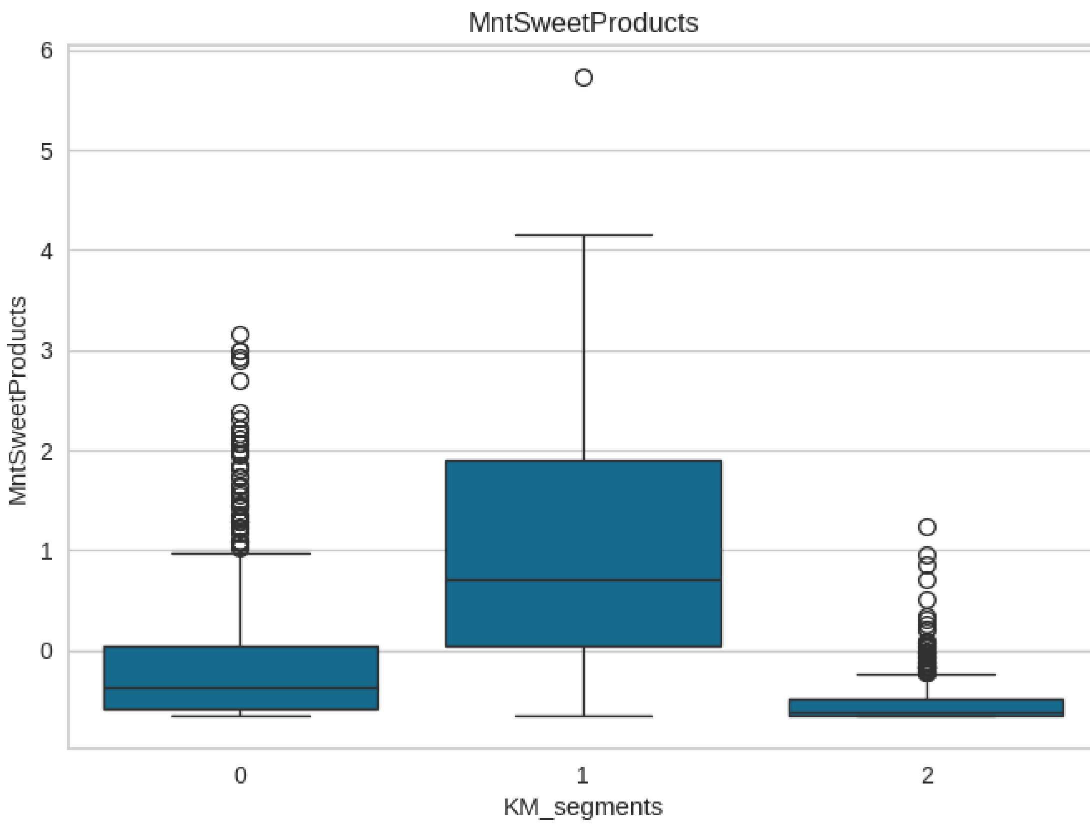

Recency
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
Recency
0
1
2
KM_segments

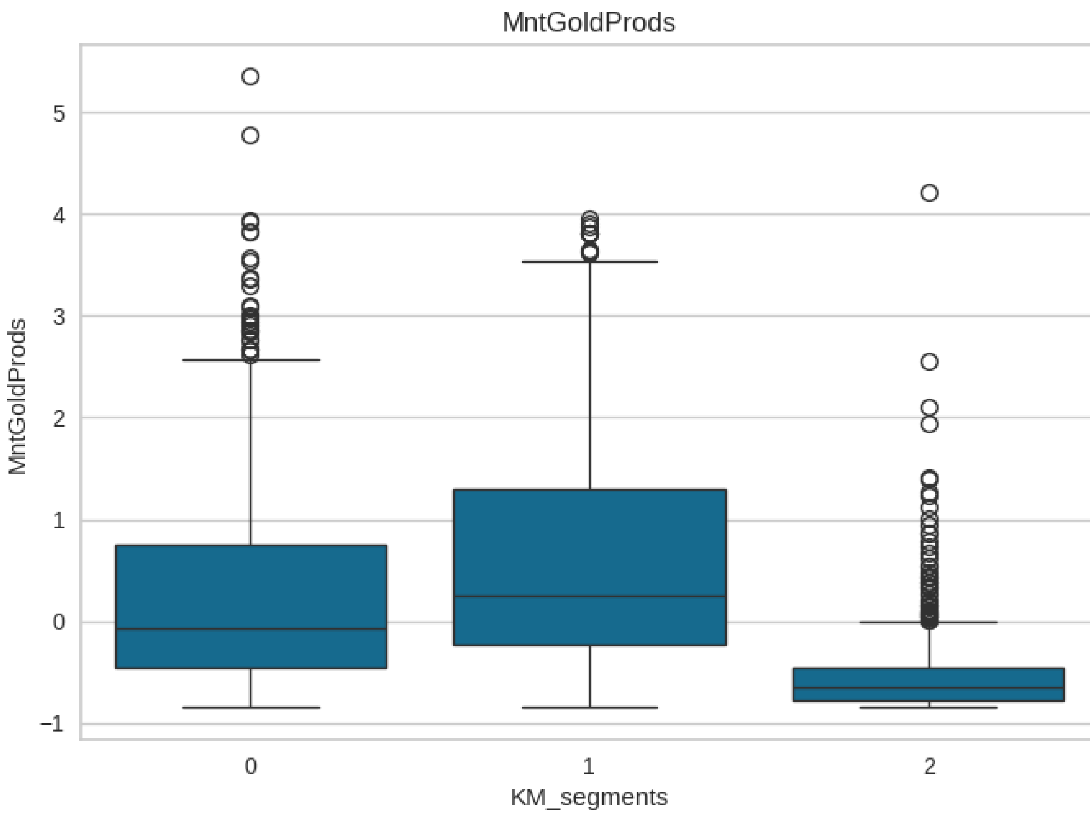

MntWines
MntWines
3
2
1
0
−1
0
1
2
KM_segments

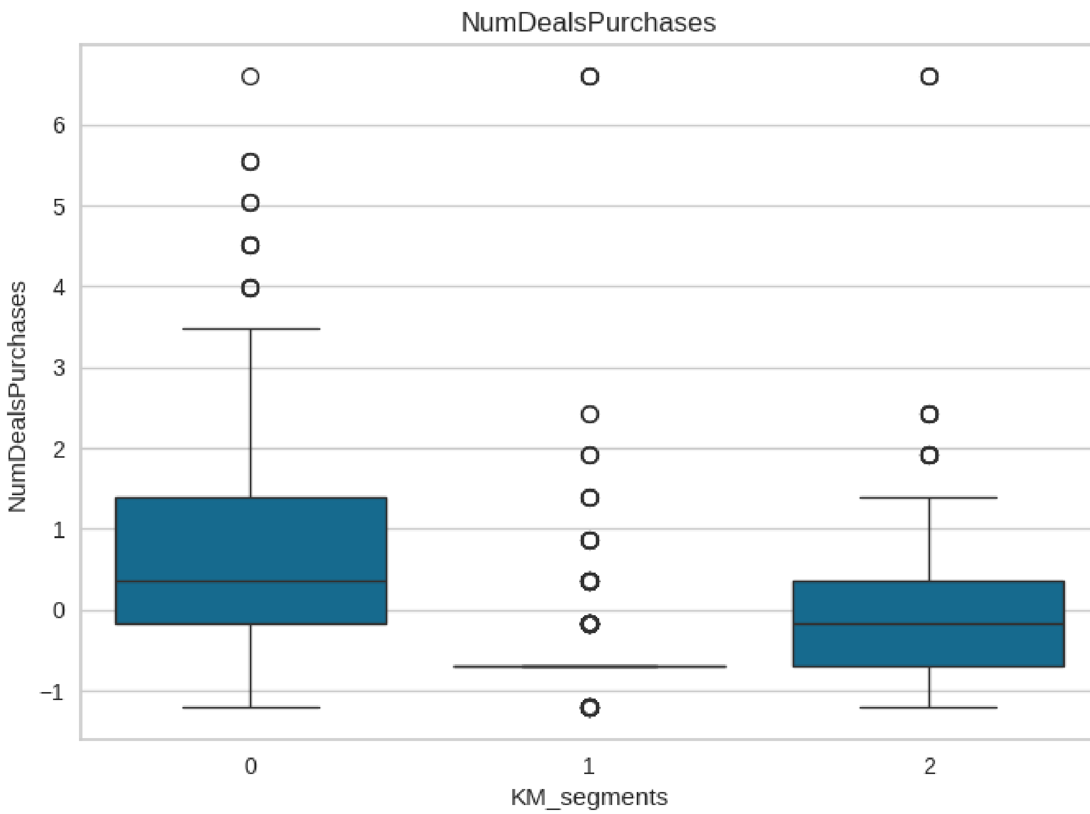

MntFruits
4
3
2
1
0
MntFruits
0
1
2
KM_segments

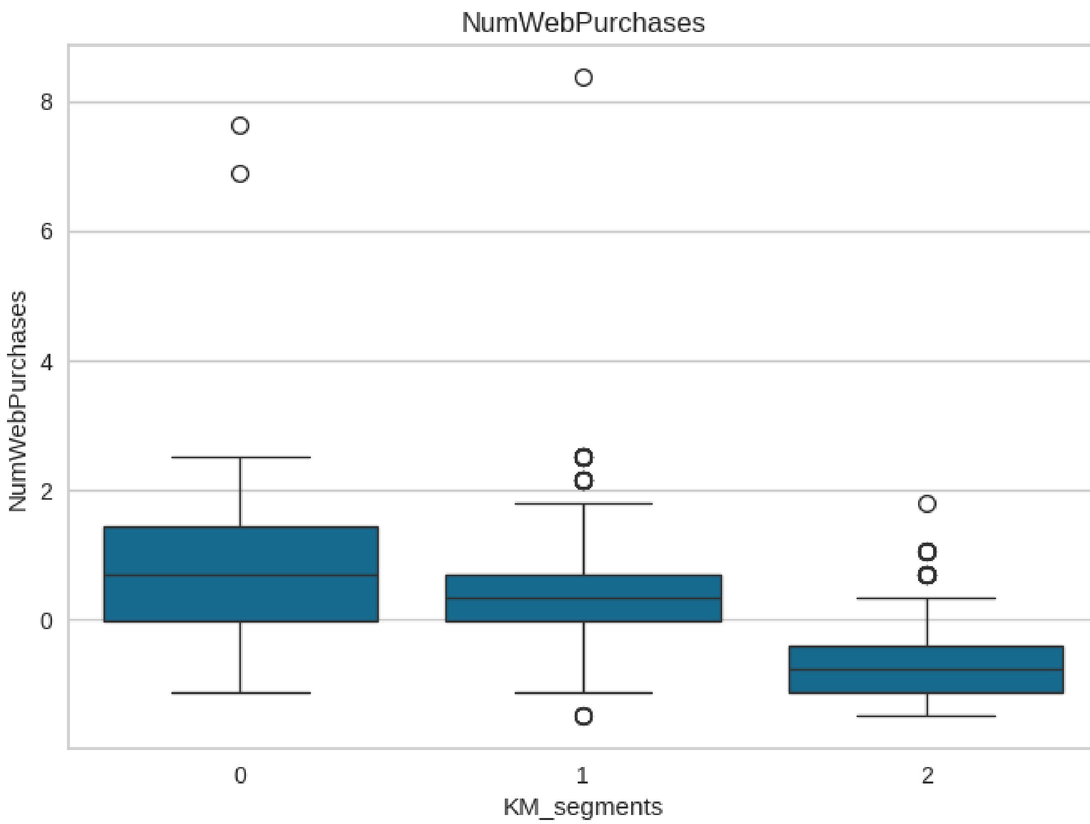

MntMeatProducts
MntMeatProducts
KM_segments
7
6
5
4
3
2
1
0
−1
0
1
2

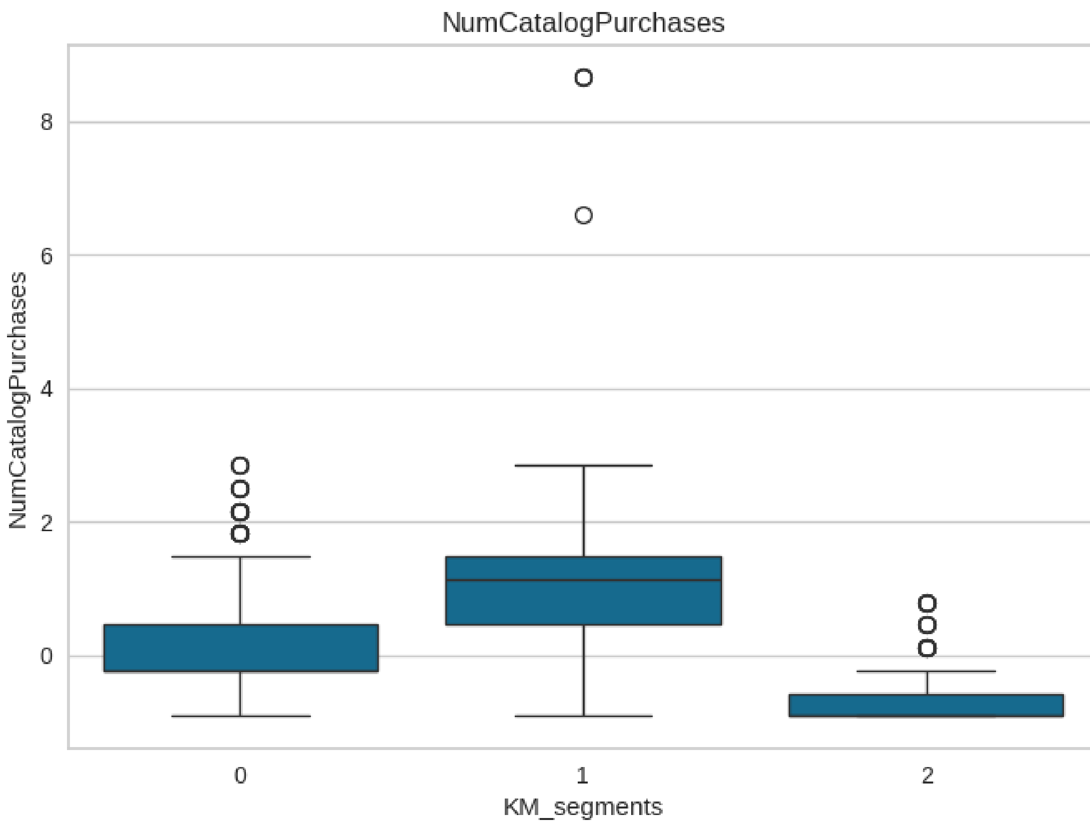

MntFishProducts
MntFishProducts
4
3
2
1
0
0
1
2
KM_segments

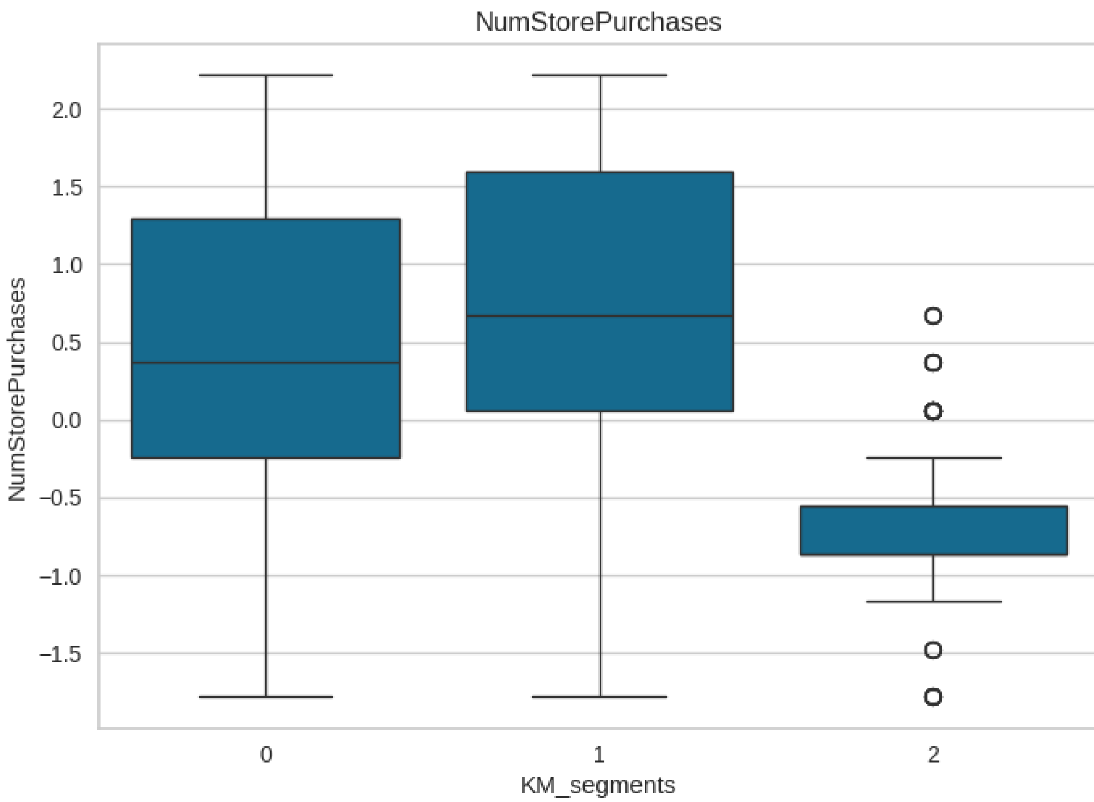

MntSweetProducts
MntSweetProducts
6
5
4
3
2
1
0
0
1
2
KM_segments

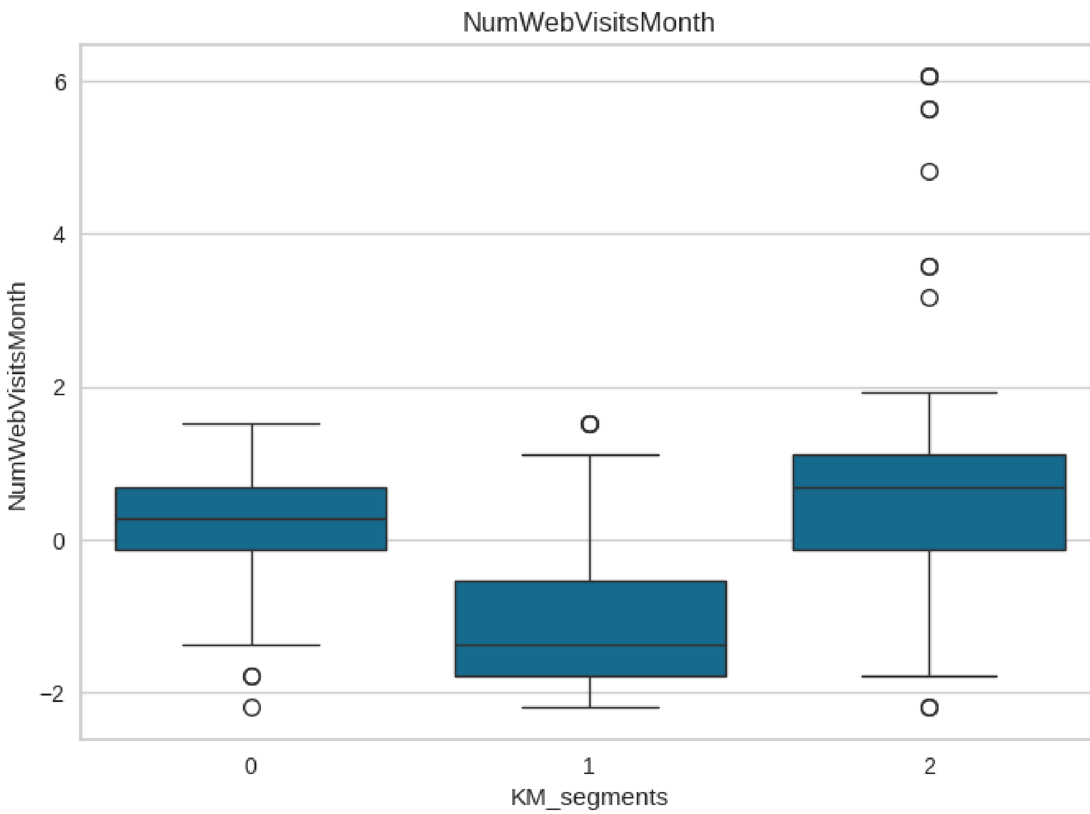

MntGoldProds
MntGoldProds
5
4
3
2
1
0
−1
0
1
2
KM_segments


NumDealsPurchases
NumDealsPurchases
KM_segments
0
1
2
6
5
4
3
2
1
0
−1


NumWebPurchases
8
6
4
2
0
NumWebPurchases
0
1
2
KM_segments


NumCatalogPurchases
NumCatalogPurchases
8
6
4
2
0
0
1
2
KM_segments

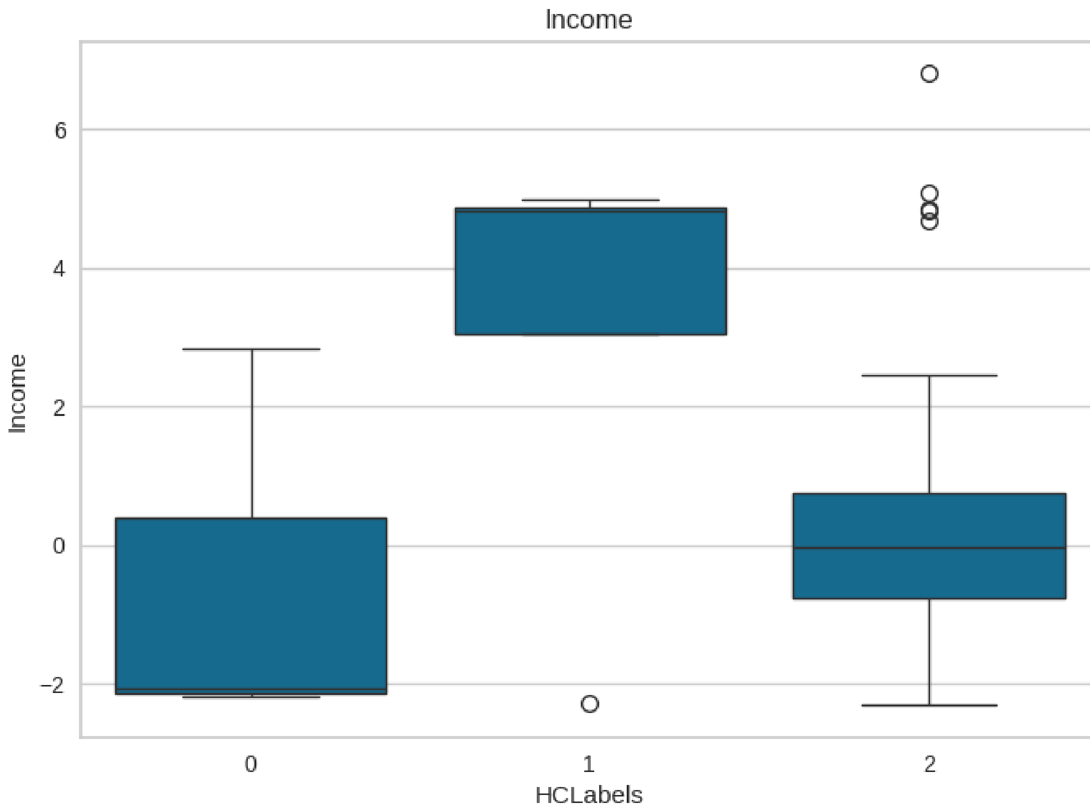

NumStorePurchases
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
NumStorePurchases
0
1
2
KM_segments

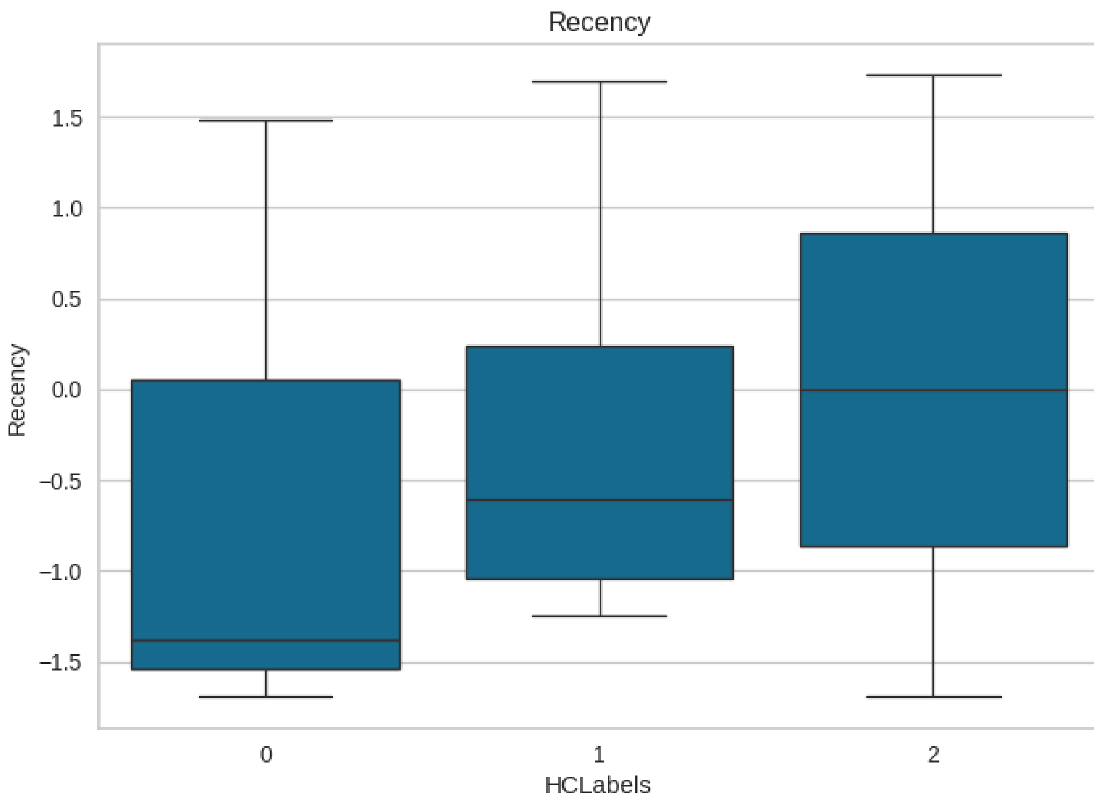


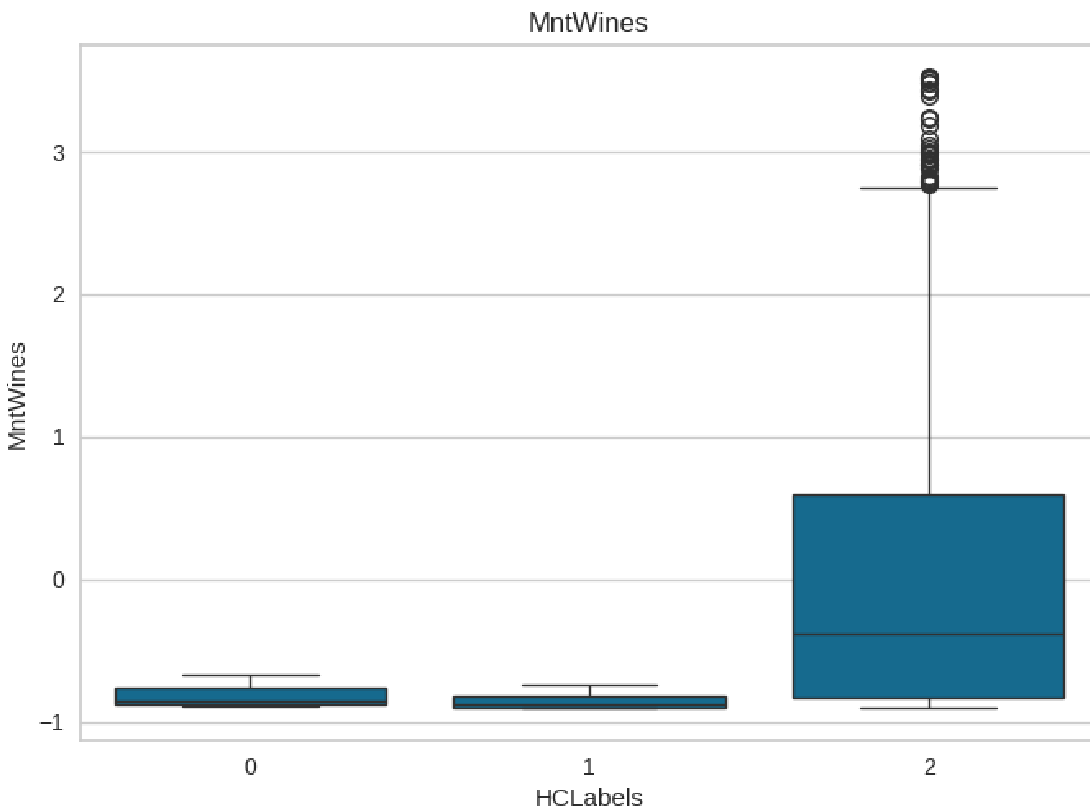
[74]: <pandas.io.formats.style.Styler at 0x781a596b3450>

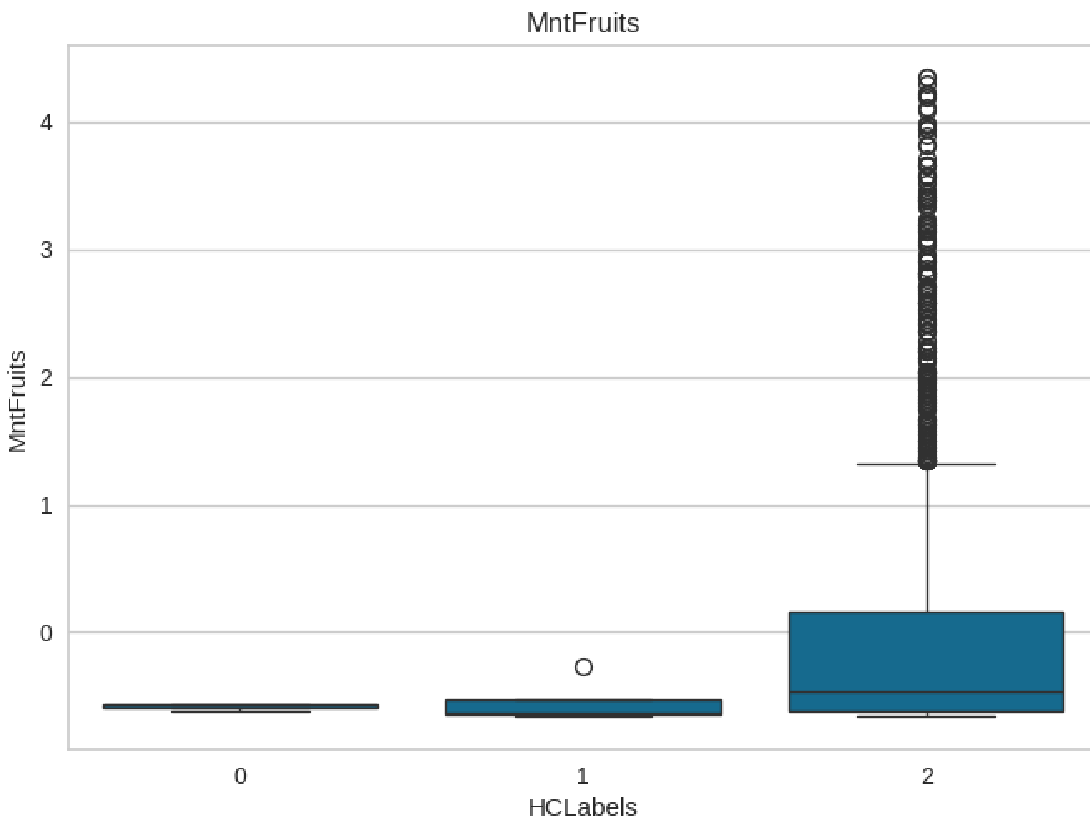
[49]:
```
#Agglomerative/Hierarchical Clustering Profile Analysis
dfHC = subset_scaled_df.copy()
dfHC['HCLabels'] = HCmodel.labels_

num_cols = cols_visualise + ['HCLabels']

mean = dfHC[num_cols].groupby('HCLabels').mean()
median = dfHC[num_cols].groupby('HCLabels').median()
df_hc = pd.concat([mean, median], axis = 0)

df_hc.index = ['group_0 Mean', 'group_1 Mean', 'group_2 Mean'
              , 'group_0 Median', 'group_1 Median' , 'group_2 Median']


for col in cols_visualise:
    sns.boxplot(x = 'HCLabels', y = col, data = dfHC)
    plt.title(col)
    plt.show()
```

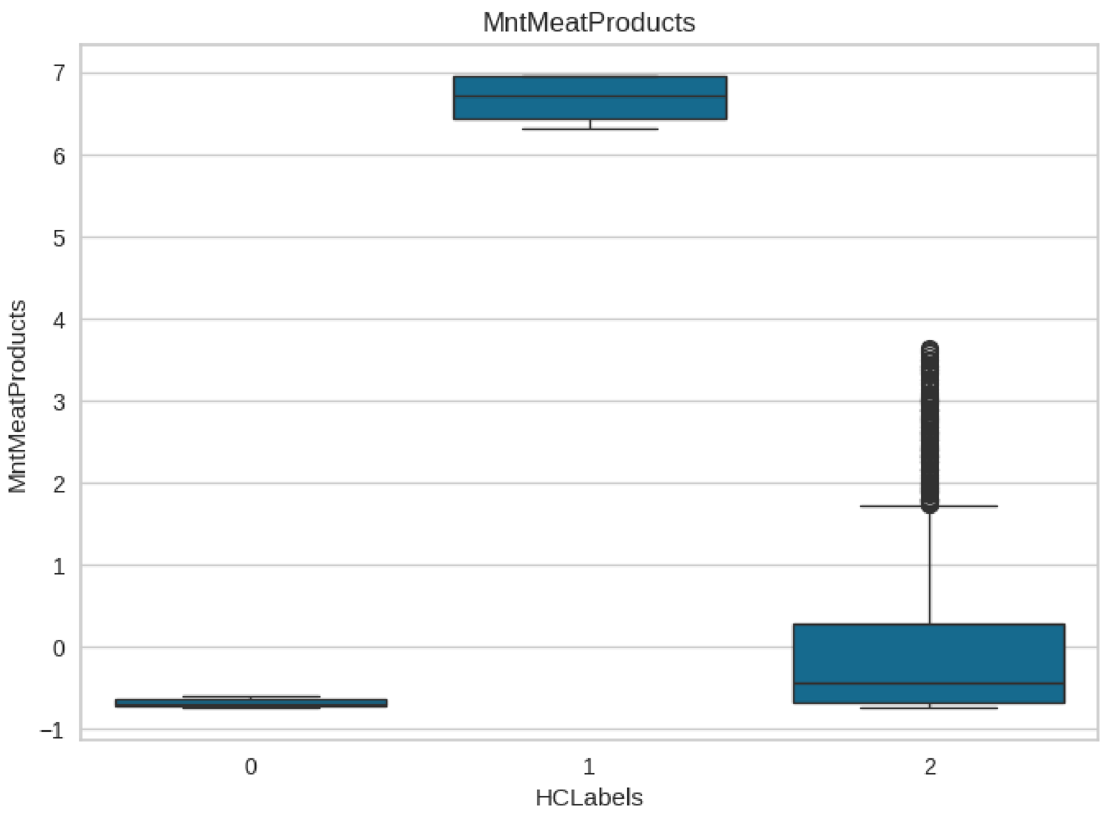
```python
hc_cluster_profile = dfHC.groupby("HCLabels").mean(numeric_only = True)
hc_cluster_profile["count_in_each_segment"] = (
    dfHC.groupby("HCLabels")["Income"].count().values
)
hc_cluster_profile.style.highlight_max(color = "purple", axis = 0)
```

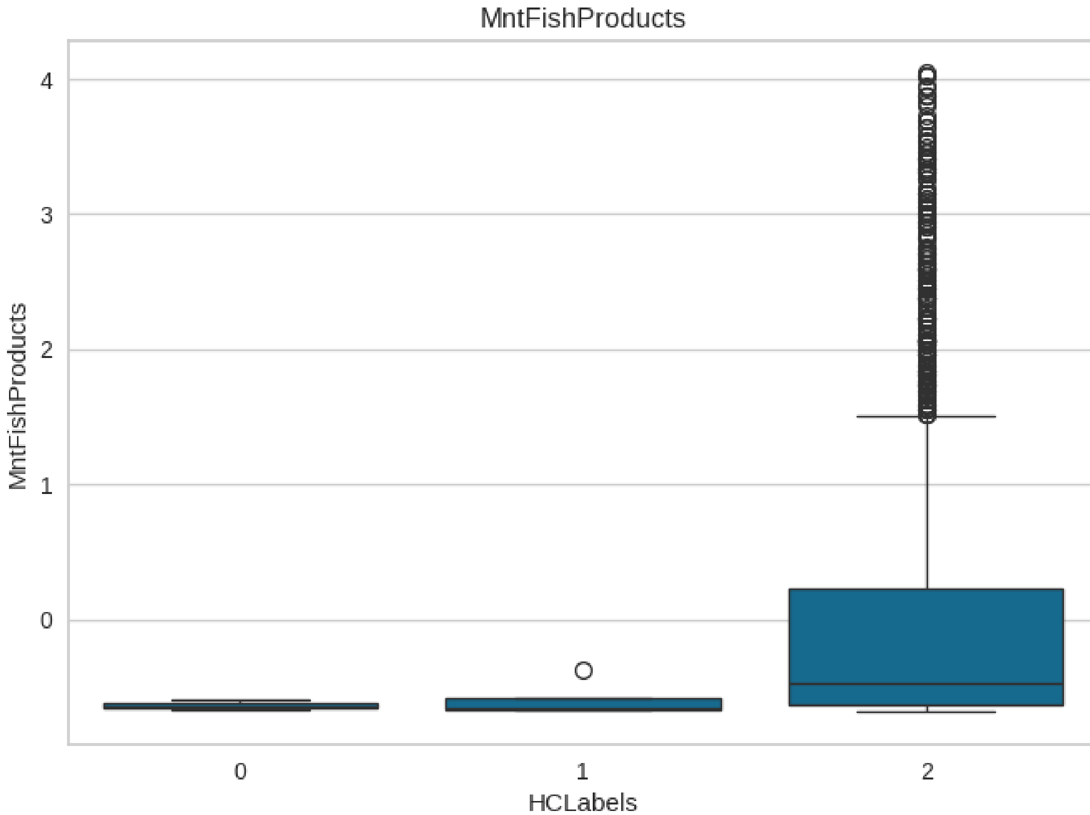


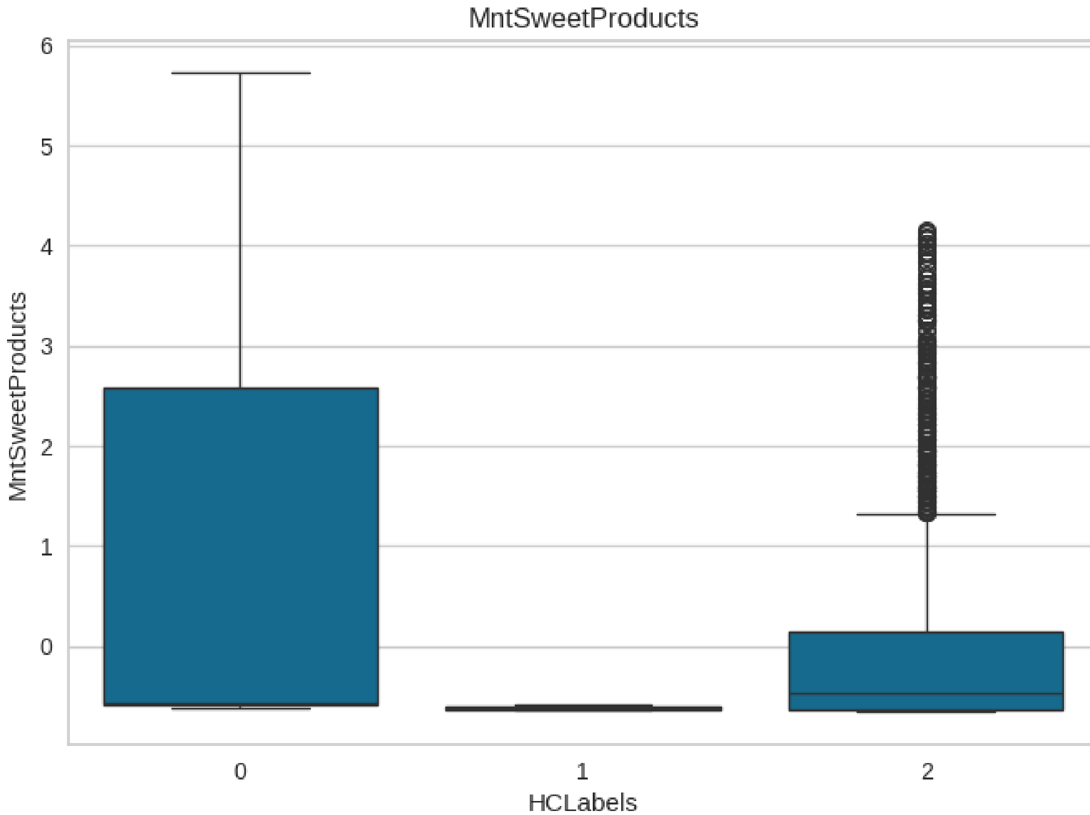

Recency
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
Recency
0
1
2
HCLabels

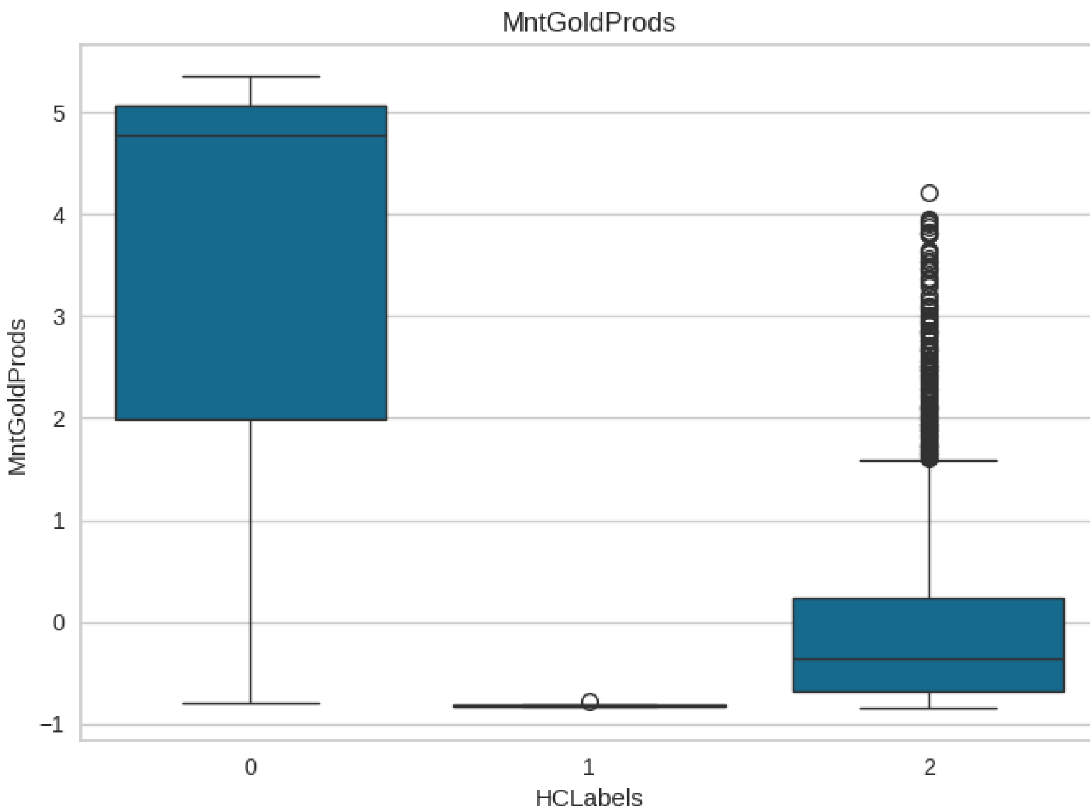

MntWines
MntWines
3
2
1
0
−1
0
1
2
HCLabels

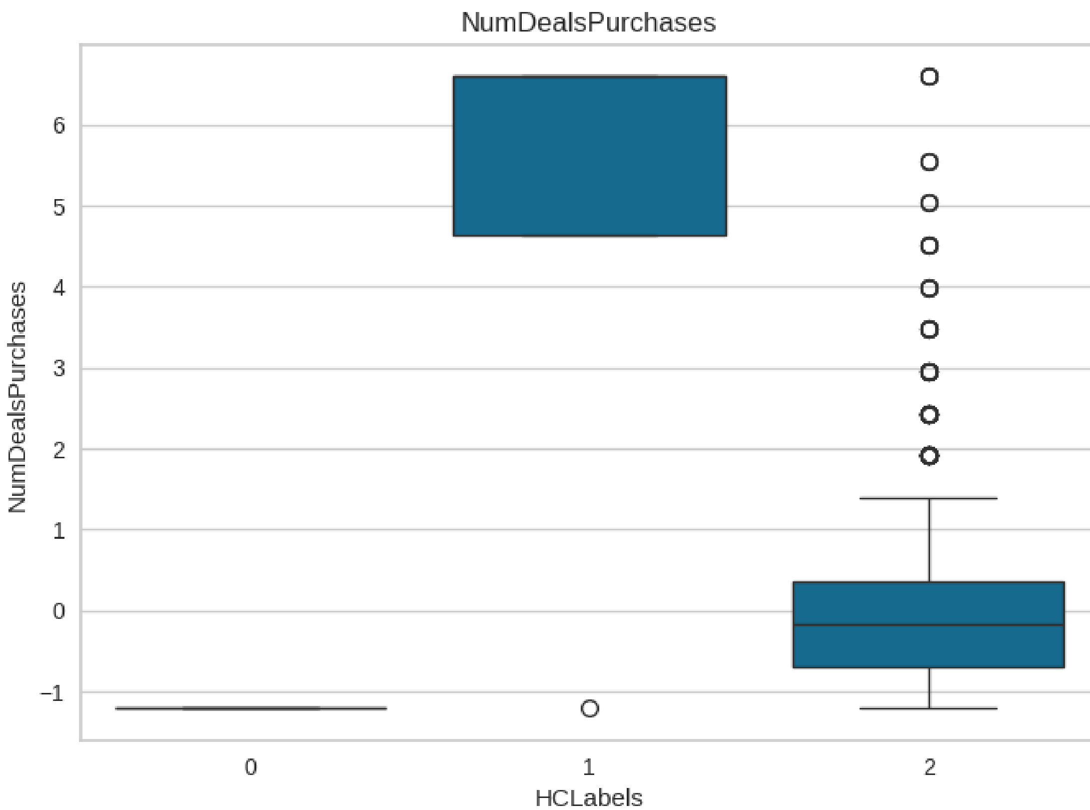

MntFruits
MntFruits
4
3
2
1
0
0
1
2
HCLabels

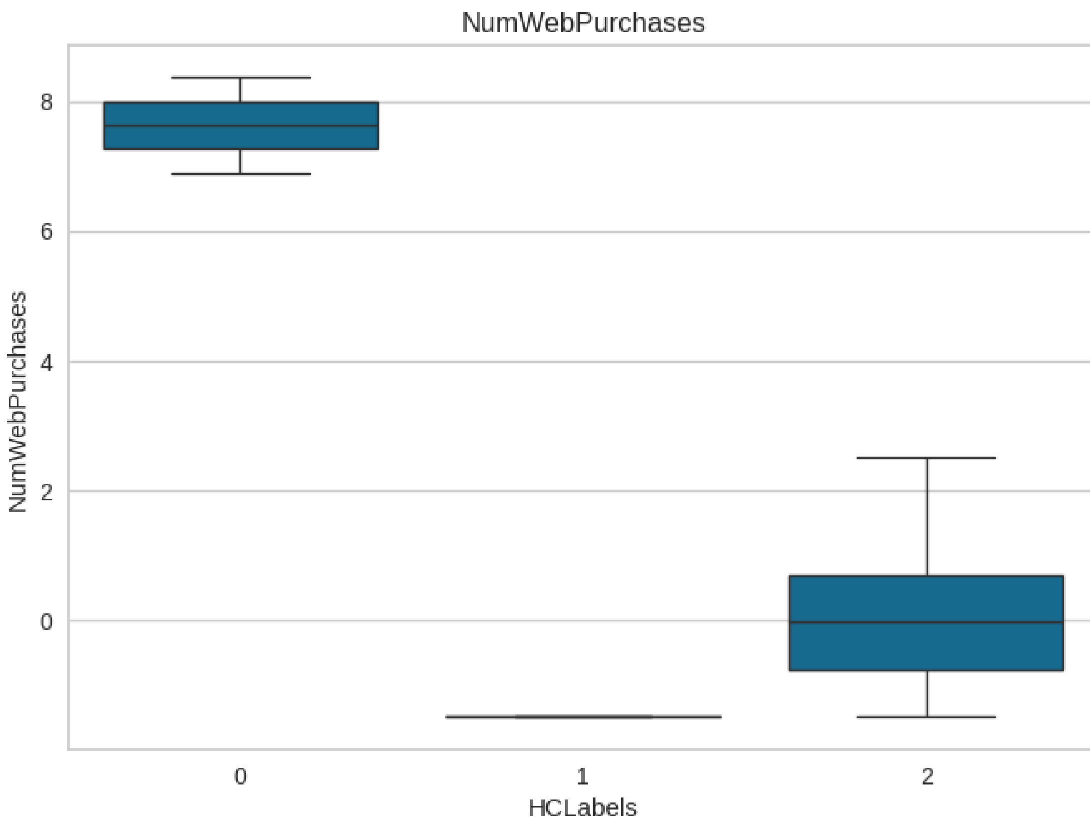

MntMeatProducts
MntMeatProducts
7
6
5
4
3
2
1
0
−1
0
1
2
HCLabels

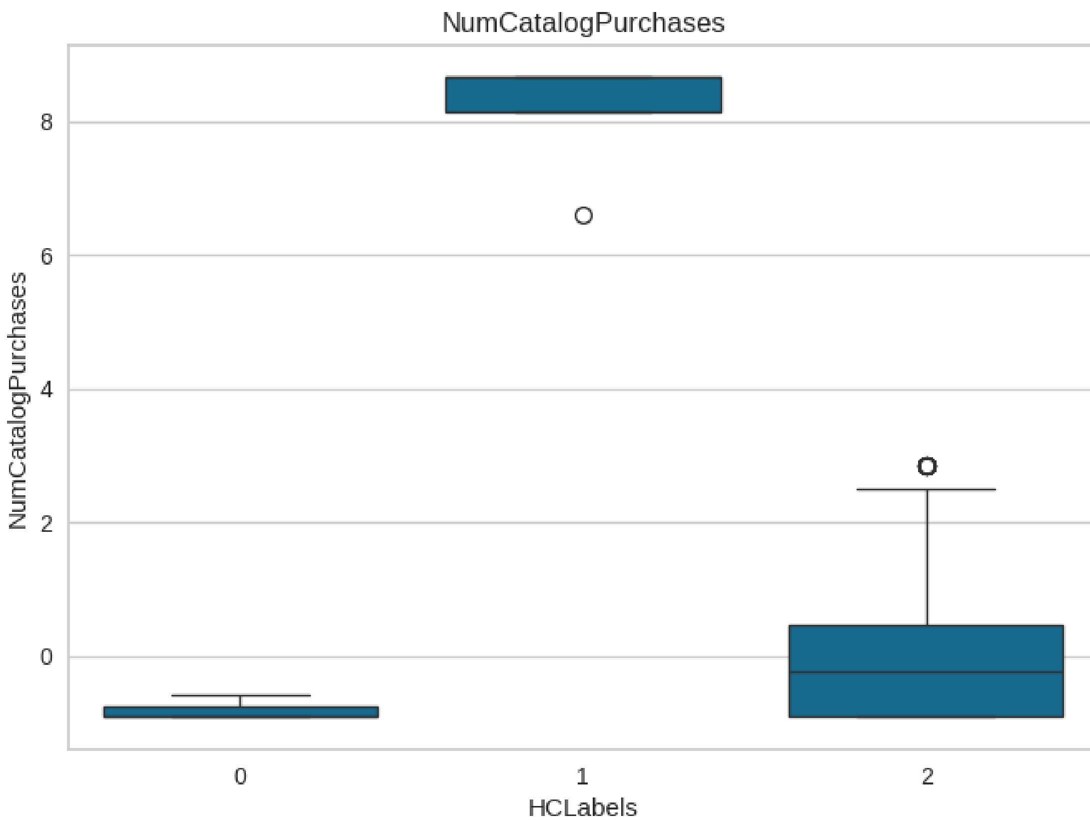

MntFishProducts
4
3
2
1
0
MntFishProducts
0
1
2
HCLabels

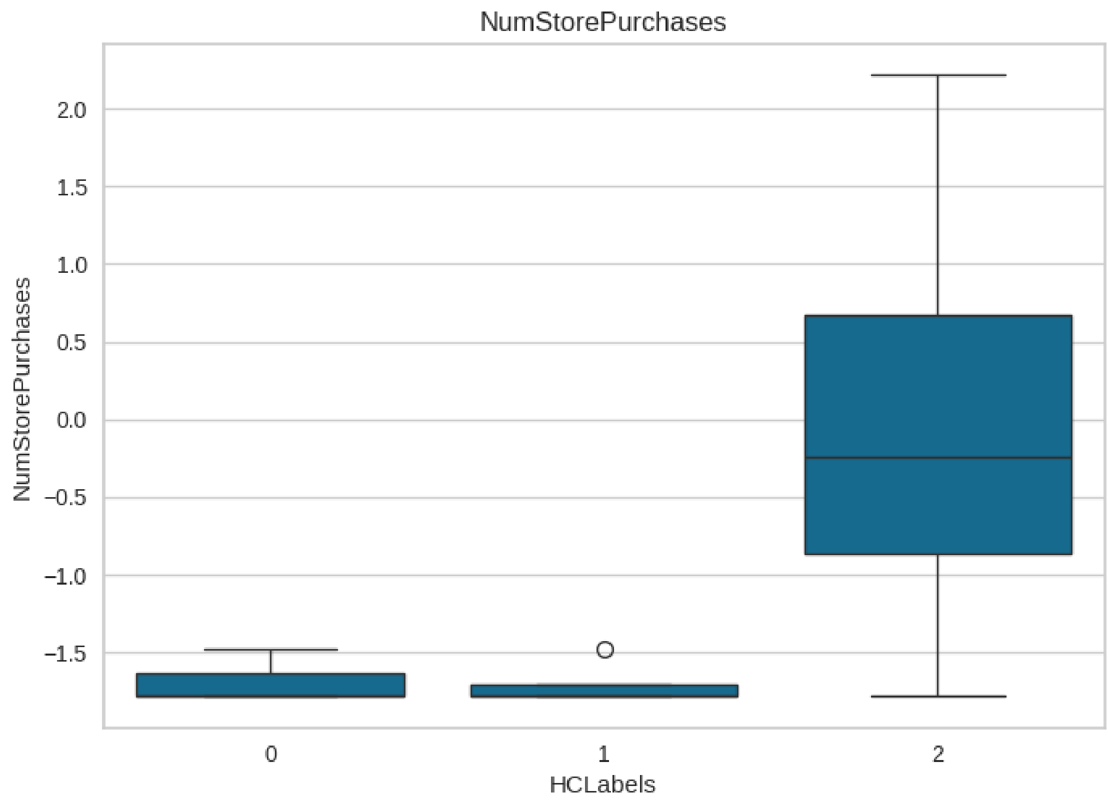

MntSweetProducts
MntSweetProducts
HCLabels
0
1
2
3
4
5
6

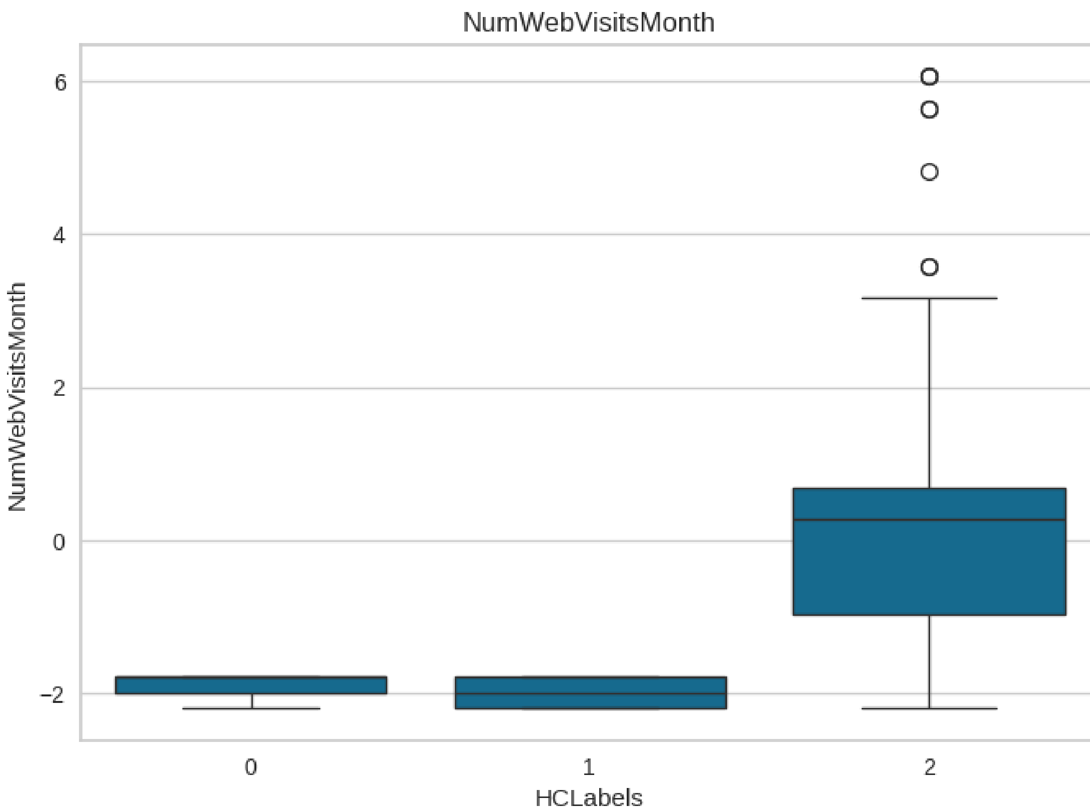

MntGoldProds
MntGoldProds
5
4
3
2
1
0
−1
0
1
2
HCLabels


NumDealsPurchases
NumDealsPurchases
6
5
4
3
2
1
0
−1
0
1
2
HCLabels


NumWebPurchases
8
6
4
2
0
NumWebPurchases
0
1
2
HCLabels


NumCatalogPurchases
NumCatalogPurchases
8
6
4
2
0
0
1
2
HCLabels


NumStorePurchases
NumStorePurchases
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
0
1
2
HCLabels



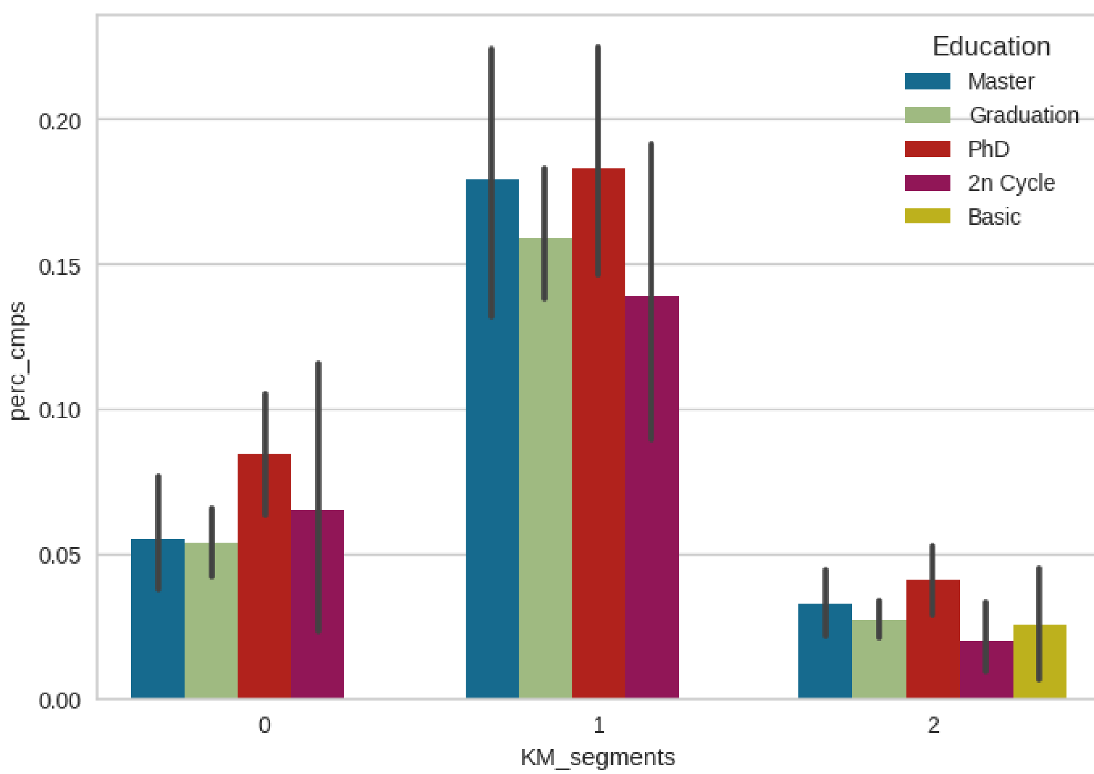
[49]: <pandas.io.formats.style.Styler at 0x781a58aef490>

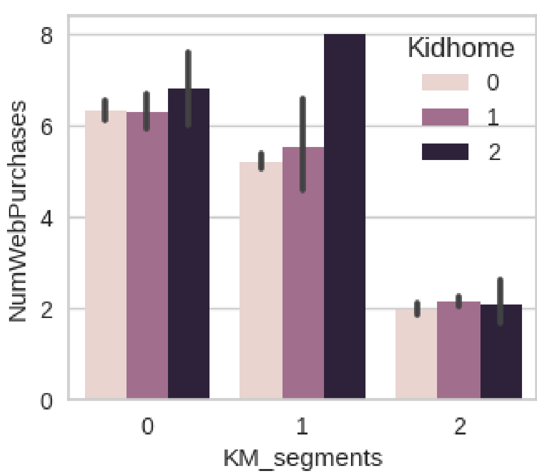
Observations: Based upon the boxplots of several features, the KMeans algorithm appears to have clustered the data better in this particular use case.

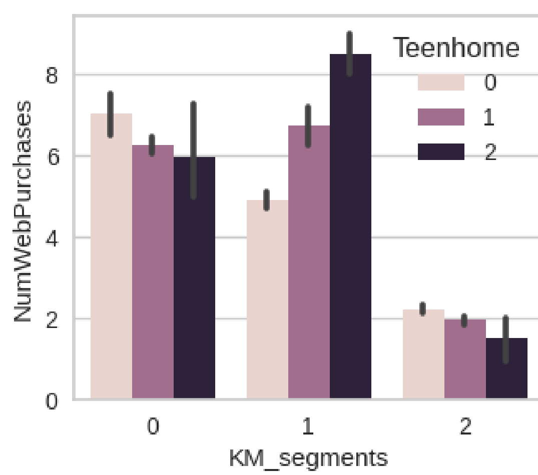
Question 15: Perform Cluster profiling on the data with the appropriate algorithm determined above using a barplot. What observations can be derived for each cluster from this plot?

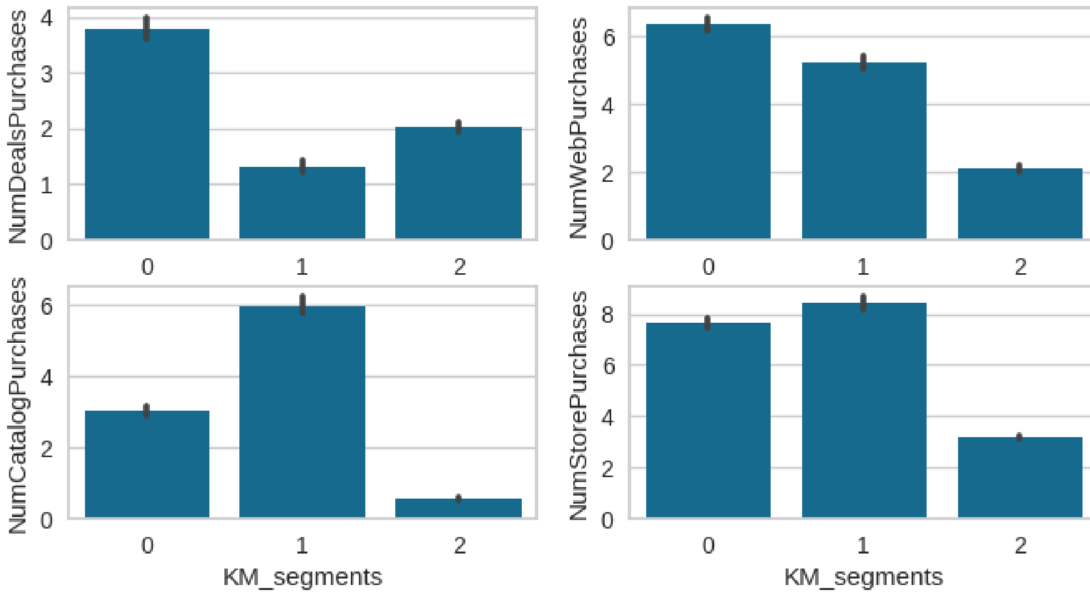
[86]:
```
sns.barplot(x='KM_segments',y='perc_cmps',hue='Education', data=df_cluster)
plt.show()

fig, axes = plt.subplots(1, 2, figsize=(8, 3))
axes = axes.flatten()
sns.barplot(x='KM_segments',y='NumWebPurchases',hue='Kidhome', data=df_cluster,␣
↪ax=axes[0])
sns.barplot(x='KM_segments',y='NumWebPurchases',hue='Teenhome',␣
↪data=df_cluster, ax=axes[1])
plt.show()

fig, axes = plt.subplots(2, 2, figsize=(8, 4))
```

```
axes = axes.flatten()
sns.barplot(x='KM_segments',y='NumDealsPurchases', data=df_cluster, ax=axes[0])
sns.barplot(x='KM_segments',y='NumWebPurchases',data=df_cluster, ax=axes[1])
sns.barplot(x='KM_segments',y='NumCatalogPurchases', data=df_cluster, ax=axes[2])
sns.barplot(x='KM_segments',y='NumStorePurchases',data=df_cluster, ax=axes[3])
plt.show()


#See how types of products order
fig, axes = plt.subplots(2, 3, figsize=(8, 8))
axes = axes.flatten()
sns.barplot(x='KM_segments',y='MntWines', data=df_cluster, ax=axes[0], palette='dark')
sns.barplot(x='KM_segments',y='MntMeatProducts',data=df_cluster, ax=axes[1],palette='dark')
sns.barplot(x='KM_segments',y='MntFishProducts', data=df_cluster, ax=axes[2],palette='dark')
sns.barplot(x='KM_segments',y='MntSweetProducts',data=df_cluster, ax=axes[3],palette='dark')
sns.barplot(x='KM_segments',y='MntGoldProds',data=df_cluster, ax=axes[4],palette='dark')
sns.barplot(x='KM_segments',y='NumWebVisitsMonth',data=df_cluster, ax=axes[5],palette='dark')
plt.show()

data_cleaned.head()
```

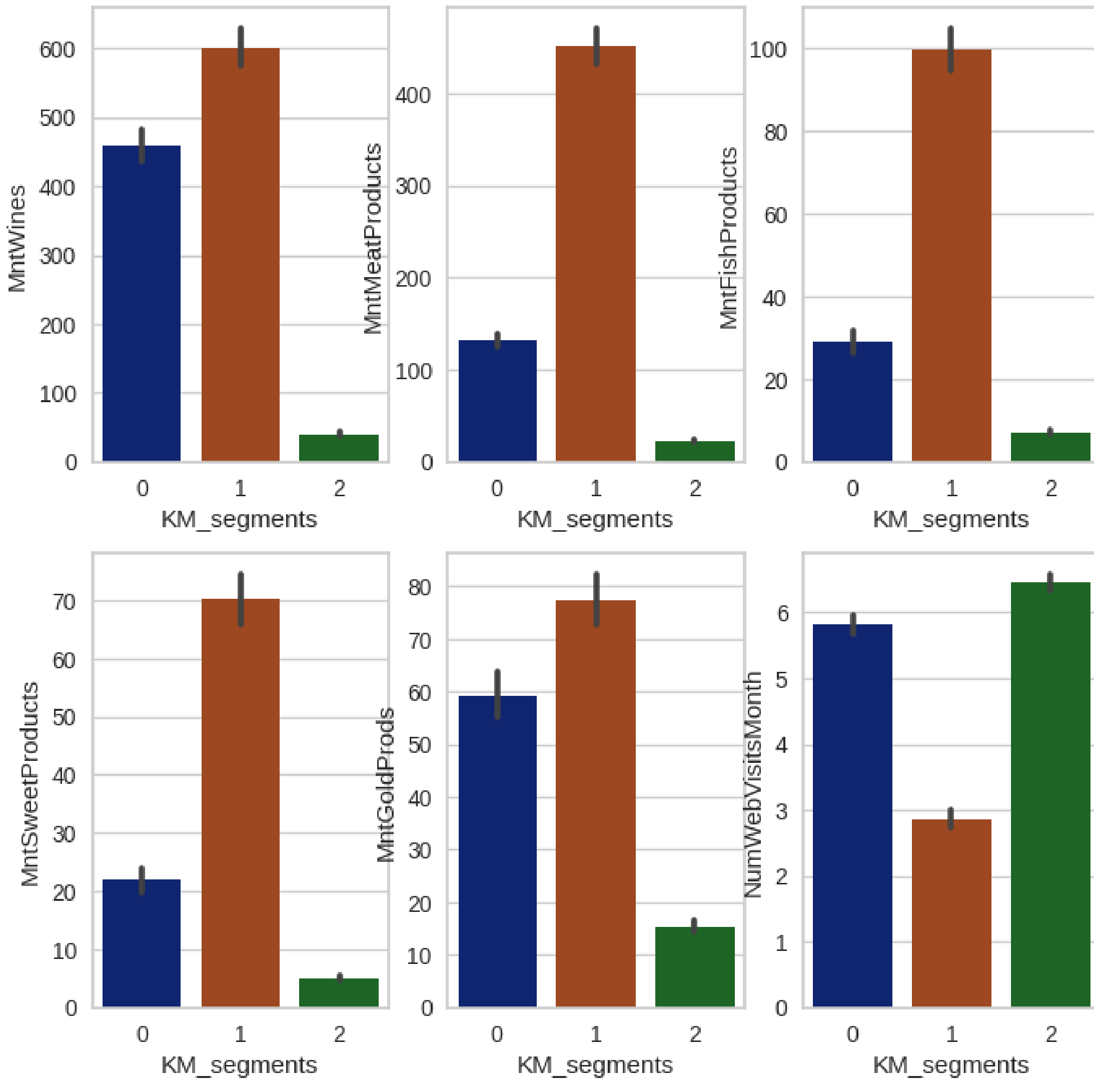

Education
Master
Graduation
PhD
2n Cycle
Basic
perc_cmps
0.20
0.15
0.10
0.05
0.00
0
1
2
KM_segments


Kidhome
0
1
2
NumWebPurchases
8
6
4
2
0
0
1
2
KM_segments


Teenhome
0
1
2
NumWebPurchases
8
6
4
2
0
0
1
2
KM_segments



NumDealsPurchases
0
1
2
3
4
0
1
2
NumWebPurchases
0
2
4
6
0
1
2
NumCatalogPurchases
0
2
4
6
0
1
2
KM_segments
NumStorePurchases
0
2
4
6
8
0
1
2
KM_segments




```
[86]:      ID  Year_Birth   Education Marital_Status   Income  Kidhome  Teenhome  \
0  5524        1957  Graduation         Single  58138.0        0         0
1  2174        1954  Graduation         Single  46344.0        1         1
2  4141        1965  Graduation       Together  71613.0        0         0
3  6182        1984  Graduation       Together  26646.0        1         0
4  5324        1981         PhD        Married  58293.0        1         0

  Dt_Customer  Recency  MntWines  MntFruits  MntMeatProducts  MntFishProducts  \
0  2012-09-04       58       635         88              546              172
1  2014-03-08       38        11          1                6                2
2  2013-08-21       26       426         49              127              111
3  2014-02-10       26        11          4               20               10
4  2014-01-19       94       173         43              118               46
```

```
   MntSweetProducts  MntGoldProds  NumDealsPurchases  NumWebPurchases  \
0                88            88                  3                8
1                 1             6                  2                1
2                21            42                  1                8
3                 3             5                  2                2
4                27            15                  5                5

   NumCatalogPurchases  NumStorePurchases  NumWebVisitsMonth  AcceptedCmp3  \
0                   10                  4                  7             0
1                    1                  2                  5             0
2                    2                 10                  4             0
3                    0                  4                  6             0
4                    3                  6                  5             0

   AcceptedCmp4  AcceptedCmp5  AcceptedCmp1  AcceptedCmp2  Complain  \
0             0             0             0             0         0
1             0             0             0             0         0
2             0             0             0             0         0
3             0             0             0             0         0
4             0             0             0             0         0

   Z_CostContact  Z_Revenue  Response  Days_Since_Enrollment  perc_cmps  \
0              3         11         1                    663   0.166667
1              3         11         0                    113   0.000000
2              3         11         0                    312   0.000000
3              3         11         0                    139   0.000000
4              3         11         0                    161   0.000000

   KM_segments
0            1
1            2
2            1
3            2
4            2
```

Observations:

- Customers in Cluster 1 (KM_segments), appear to utilize campaigns more.
- Customers in Clusters 0 and 1 tend to buy more wines.
- Customers in Cluster 0 tend to Purchase more deals.
- Customers in Cluster 2 do not purchase via catalog much.
- Customers in Cluster 1 purchase more products while visiting the web the least.
- Customers in Cluster 2 visit the web the most, giving the website the optimal way to target that audience.

- Customers with PhDs tend to participate in the campaigns more.
- Customers with kids or teens at home tend to prefer web purchases more than those without kids.

2.8 Business Recommedations

- We have seen that 3 clusters are distinctly formed using both methodologies and the clusters are analogous to each other.
- Cluster 0 are moderate spenders who tend to purchase more deals.
- Cluster 1 tends to do more purchasing of all products accross the board.
- Cluster 2 Customers purchase the least amount of products while visiting the website more often.

Here are **5–7 actionable business recommendations** based on the cluster profiling:

2.8.1 1. Focus on Retaining High-Value Customers (Cluster 1)

- **Offer Exclusive Loyalty Programs**: Provide tailored loyalty benefits, early access to products, and exclusive discounts to maintain engagement and drive repeat purchases.
- **Upsell and Cross-Sell**: Introduce premium products or bundles targeting their high spending patterns across product categories like wines, gold products, and meats.
- **Personalized Campaigns**: Create marketing campaigns around those with higher education. Extend the marketing for these campaigns into Universities and high educated areas.

2.8.2 2. Activate Potential in Moderate-Spending Customers (Cluster 0)

- **Incentivize Higher Engagement**: Offer targeted discounts or special offers to encourage increased spending and purchases across channels.
- **Educate About Products**: Provide content (emails, guides, or social media) showcasing the value and uniqueness of products they don't purchase frequently.
- **Improve Campaign Effectiveness**: Refine campaign messaging based on their moderate response rate to increase acceptance.

2.8.3 3. Reengage Low-Value Customers (Cluster 2)

- **Win-Back Campaigns**: Implement campaigns specifically aimed at bringing back inactive customers, such as offering steep discounts or limited-time offers.
- **Understand Barriers to Engagement**: Conduct surveys or collect feedback to identify reasons for their low purchases and disengagement.
- **Promote Entry-Level Products**: Introduce affordable or trial-sized products to ease them into higher spending.
- **Utilize Web Traffic**: Low-value customers tend to visit the website the most, leaving the website as an optimal way to target this group of people.

2.8.4 4. Convert Browsers into Buyers (Cluster 2)

- **Optimize Website Experience**: Since Cluster 2 has high website visits but low spending, improve website navigation, showcase popular products, and streamline the checkout process.
- **Targeted Digital Campaigns**: Retarget these users with ads or emails featuring products they browsed but didn't purchase.
- **Offer Online-Exclusive Discounts**: Provide web-only discounts or promotions to convert visits into purchases.

2.8.5 5. Strengthen Digital and Multi-Channel Strategies

- **Seamless Omni-Channel Experience**: Ensure a consistent shopping experience across all channels (web, catalog, and store) to encourage cross-channel engagement, especially for Clusters 0 and 1.
- **Digital Campaigns for All Clusters**: Focus on targeted digital campaigns, particularly for Clusters 0 and 2, as they have moderate to high online engagement.

2.8.6 6. Develop Campaigns to Boost Responses

- Use the insights from Clusters 0 and 1 (which show higher response rates) to refine campaign targeting and messaging. Emulate successful strategies used for Cluster 2 to increase responses across other segments.

2.8.7 7. Leverage Product-Specific Insights

- Promote popular categories (e.g., wines, gold products) to high-value clusters, while running introductory campaigns for less-engaged clusters to familiarize them with premium products.
- ## 2.9 In Cluster 1, we see that customers with 2 kids or teens tend to buy more wine. We can utilize this to keep them engaged, while altering the market image for wine to appeal to those without kids.

By focusing on these strategies, the company can enhance engagement, increase revenue, and strengthen customer loyalty across all clusters.

[ ]:

[ ]: