|  |
| --- |
| Experiment No. 5 |
| Apply appropriate Unsupervised Learning Technique on the  Wholesale Customers Dataset |
| Date of Performance: 14/08/2023 |
| Date of Submission: 21/08/2023 |

Aim: Apply appropriate Unsupervised Learning Technique on the Wholesale CustomersDataset.

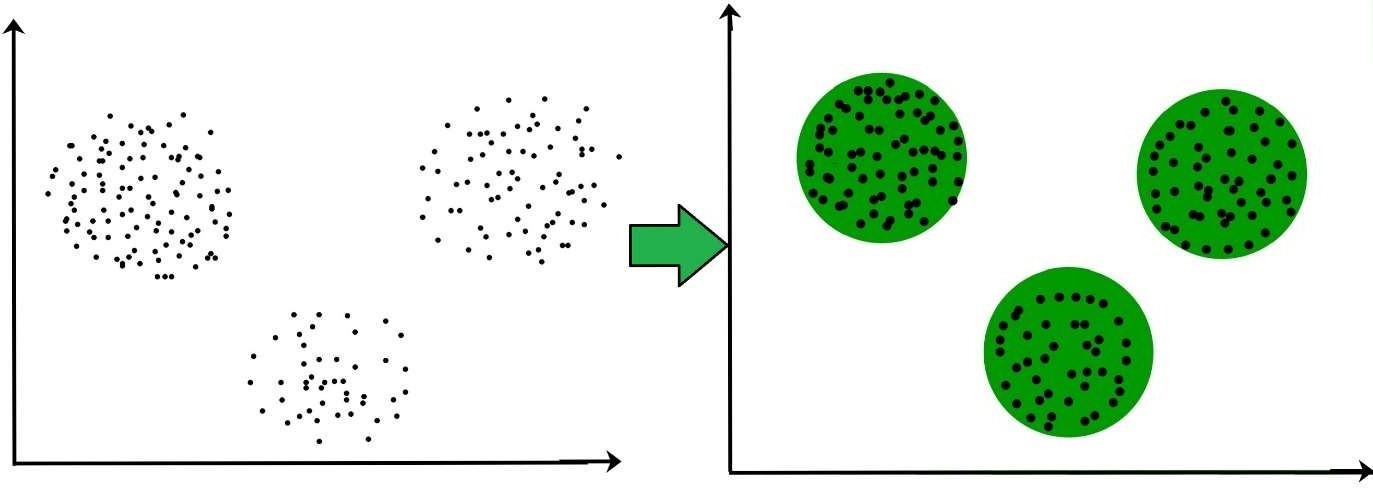
Objective: Able to perform various feature engineering tasks, apply Clustering Algorithm on the given dataset.

Theory:

It is basically a type of unsupervised learning method. An unsupervised learning method is a method in which we draw references from datasets consisting of input data without labeled responses. Generally, it is used as a process to find meaningful structure, explanatory underlying processes, generative features, and groupings inherent in a set of examples.

Clustering is the task of dividing the population or data points into a number of groups such that data points in the same groups are more similar to other data points in the same group and dissimilar to the data points in other groups. It is basically a collection of objects on the basis of similarity and dissimilarity between them.

For example: The data points in the graph below clustered together can be classified into one single group. We can distinguish the clusters, and we can identify that there are 3 clusters in the below picture.



Dataset:

This data set refers to clients of a wholesale distributor. It includes the annual spending in monetary units (m.u.) on diverse product categories. The wholesale distributor operating in different regions of Portugal has information on annual spending of several items in their stores across different regions and channels. The dataset consist of 440 large retailers annual spending on 6 different varieties of product in 3 different regions (lisbon , oporto, other) and across different sales channel ( Hotel, channel)

Detailed overview of dataset

Records in the dataset = 440 ROWS

Columns in the dataset = 8 COLUMNS

FRESH: annual spending (m.u.) on fresh products (Continuous)

MILK:- annual spending (m.u.) on milk products (Continuous)

GROCERY:- annual spending (m.u.) on grocery products (Continuous)

FROZEN:- annual spending (m.u.) on frozen products (Continuous)

DETERGENTS\_PAPER :- annual spending (m.u.) on detergents and paper products

(Continuous)

DELICATESSEN:- annual spending (m.u.)on and delicatessen products (Continuous);

CHANNEL: - sales channel Hotel and Retailer REGION:- three regions ( Lisbon, Oporto, Other)

Conclusion: In this analysis, Principal Component Analysis (PCA), an unsupervised learning technique, was applied t\o the Wholesale Customers dataset. The primary objective was dimensionality reduction and gaining insights into data variance. The cumulative explained variance plot illustrated diminishing returns, suggesting that a select number of principal components capture most of the dataset's variance. By studying this plot, we can decide how many principal components to retain, simplifying the data while preserving essential information. PCA is a valuable tool for preprocessing data, revealing patterns, and can be used in tasks like clustering and data visualization.

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

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| df = pd.read\_csv('customers.csv') print(df) | | | | |  |  |  |  |
| 0  1  2  3  4 ..  435  436  437  438  439  0  1  2  3  4 ..  435  436  437  438  439  [440 | Channel Region  2  2  2  1  2  ... ...  1  1  2  1 1  Delicatessen 1338  1776  7844  1788  5185 ... 2204  2346  1867  2125  52  rows x 8 columns | 3  3  3  3  3  3  3  3  3 3  ] | Fresh  12669 7057  6353 13265  22615 ... 29703  39228  14531  10290 2787 | Milk  9656  9810  8808  1196  5410 ...  12051  1431  15488  1981 1698 | Grocery  7561  9568  7684  4221  7198 ...  16027  764  30243  2232 2510 | Frozen  214 1762  2405  6404  3915 ...  13135  4510  437  1038 65 | Detergents\_Paper  2674  3293  3516  507  1777  ...  182  93  14841  168 477 | \ |
| df.head() | | | |  |  |  |  |  |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1. 2 2. 2 3. 2 4. 1 5. 2 | 3  3  3  3  3 | 12669  7057  6353  13265  22615 | 9656  9810  8808  1196  5410 | 7561  9568  7684  4221  7198 | 214  1762  2405  6404  3915 | 2674  3293  3516  507  1777 | 1338  1776  7844  1788  5185 |
| print("Data Types") df.dtypes |  |  |  |  |  |  |  |
| Data Types  Channel Region  Fresh  Milk  Grocery  Frozen  Detergents\_Paper Delicatessen dtype: object | int64 int64 int64 int64 int64 int64 int64 int64 | |  |  |  |  |  |
| print("Missing values per column:") print(df.isnull().sum()) | | | |  |  |  |  |

Channel Region Fresh Milk Grocery Frozen Detergents\_Paper Delicatessen

Missing values per column:

|  |  |
| --- | --- |
| Channel | 0 |
| Region | 0 |
| Fresh | 0 |
| Milk | 0 |
| Grocery | 0 |
| Frozen | 0 |
| Detergents\_Paper | 0 |
| Delicatessen dtype: int64 | 0 |

print("Descriptive Statistics:") print(df.describe())

print("Number of duplicate rows: ", df.duplicated().sum())

Descriptive Statistics:

Channel Region Fresh Milk Grocery \

count 440.000000 440.000000 440.000000 440.000000 440.000000

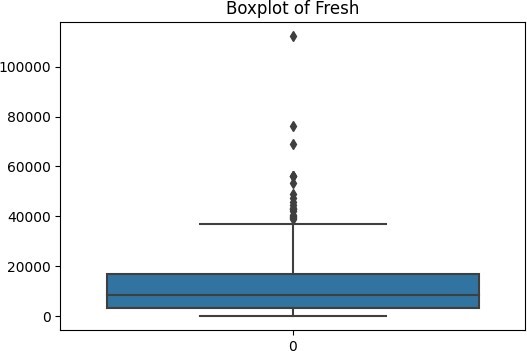
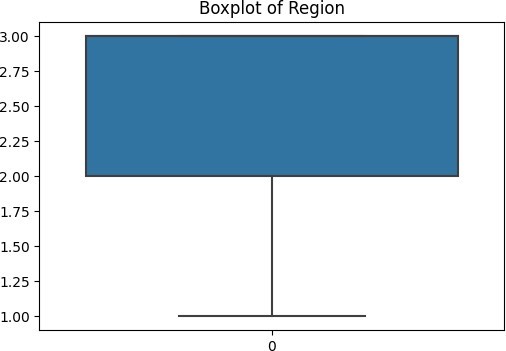
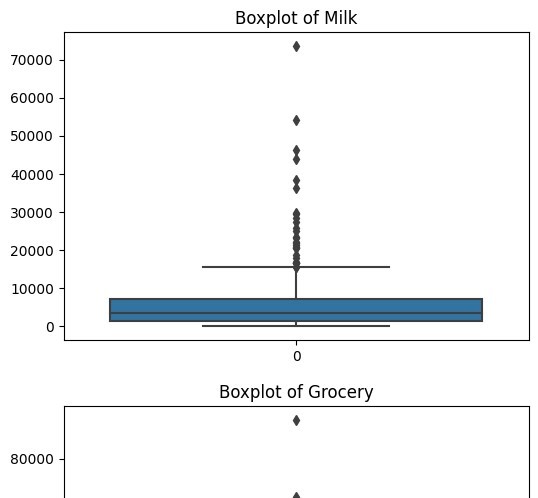
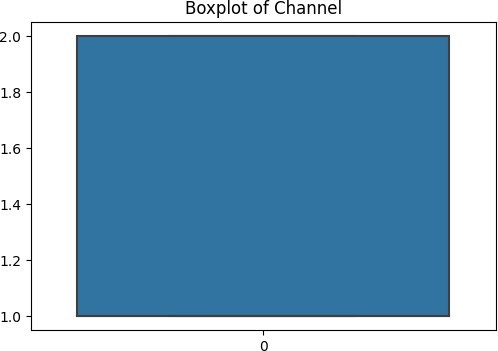
|  |  |  |  |
| --- | --- | --- | --- |
| mean | 1.322727 | 2.543182 12000.297727 5796.265909 | 7951.277273 |
| std | 0.468052 | 0.774272 12647.328865 7380.377175 | 9503.162829 |
| min | 1.000000 | 1.000000 3.000000 55.000000 | 3.000000 |
| 25% | 1.000000 | 2.000000 3127.750000 1533.000000 | 2153.000000 |
| 50% | 1.000000 | 3.000000 8504.000000 3627.000000 | 4755.500000 |
| 75% | 2.000000 | 3.000000 16933.750000 7190.250000 | 10655.750000 |
| max | 2.000000 | 3.000000 112151.000000 73498.000000 | 92780.000000 |
| count | Frozen  440.000000 | Detergents\_Paper Delicatessen 440.000000 440.000000 |  |
| mean | 3071.931818 | 2881.493182 1524.870455 |  |
| std | 4854.673333 | 4767.854448 2820.105937 |  |
| min | 25.000000 | 3.000000 3.000000 |  |
| 25% | 742.250000 | 256.750000 408.250000 |  |
| 50% | 1526.000000 | 816.500000 965.500000 |  |
| 75% | 3554.250000 | 3922.000000 1820.250000 |  |
| max | 60869.000000 | 40827.000000 47943.000000 |  |
| Number of duplicate rows: 0 | | |

|  |
| --- |
| import seaborn as sns import matplotlib.pyplot as plt  # boxplots for all features for column in df.columns: plt.figure(figsize=(6, 4)) sns.boxplot(df[column]) plt.title(f'Boxplot of {column}') plt.show()  # Function to detect outliers def detect\_outliers(dataframe, column):  Q1 = dataframe[column].quantile(0.25)  Q3 = dataframe[column].quantile(0.75)  IQR = Q3 - Q1  outliers = dataframe[(dataframe[column] < Q1 - 1.5\*IQR) | (dataframe[column] > Q3 + 1.5\*IQR)] return outliers  # number of outliers for each feature for column in df.columns:  outliers = detect\_outliers(df, column) print(f'Number of outliers in {column}: {len(outliers)}') |

df.corr()

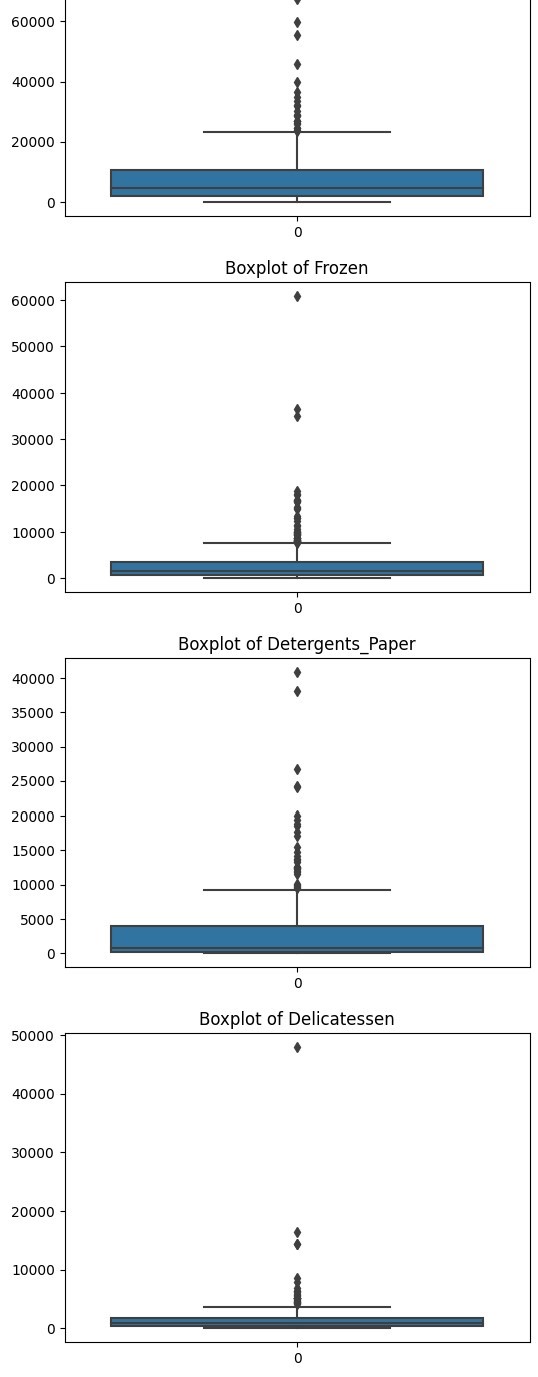
|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Channel | Region | Fresh | Milk | Grocery | Frozen Detergents\_Paper Delicatessen | | |
| Channel | 1.000000 | 0.062028 | -0.169172 | 0.460720 | 0.608792 | -0.202046 | 0.636026 | 0.056011 |
| Region | 0.062028 | 1.000000 | 0.055287 | 0.032288 | 0.007696 | -0.021044 | -0.001483 | 0.045212 |
| Fresh | -0.169172 | 0.055287 | 1.000000 | 0.100510 | -0.011854 | 0.345881 | -0.101953 | 0.244690 |
| Milk | 0.460720 | 0.032288 | 0.100510 | 1.000000 | 0.728335 | 0.123994 | 0.661816 | 0.406368 |
| Grocery | 0.608792 | 0.007696 | -0.011854 | 0.728335 | 1.000000 | -0.040193 | 0.924641 | 0.205497 |
| Frozen | -0.202046 | -0.021044 | 0.345881 | 0.123994 | -0.040193 | 1.000000 | -0.131525 | 0.390947 |
| Detergents\_Paper | 0.636026 | -0.001483 | -0.101953 | 0.661816 | 0.924641 | -0.131525 | 1.000000 | 0.069291 |
| Delicatessen | 0.056011 | 0.045212 | 0.244690 | 0.406368 | 0.205497 | 0.390947 | 0.069291 | 1.000000 |

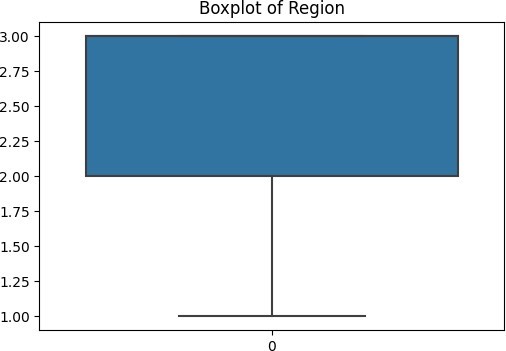
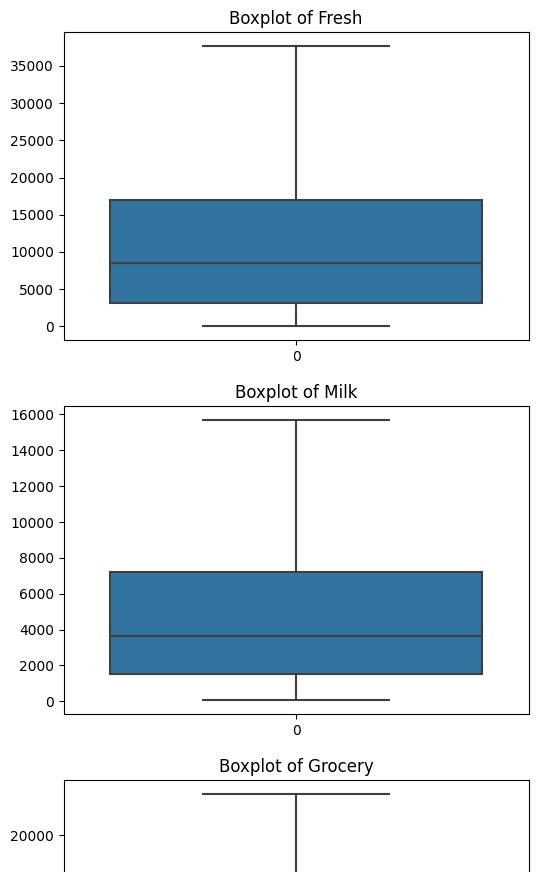
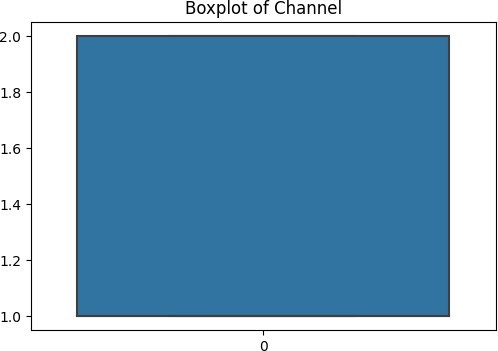




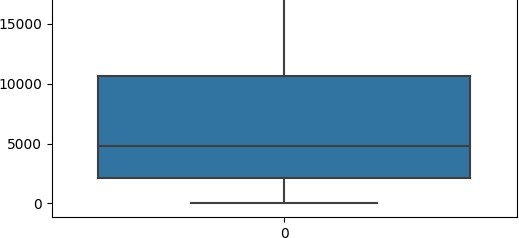
|  |
| --- |
| for column in df.columns: plt.figure(figsize=(6, 4)) sns.boxplot(df[column])  plt.title(f'Boxplot of {column}')Number of outliers in Channel: 0  plt.show()Number of outliers in Region: 0  Number of outliers in Fresh: 20  Number of outliers in Milk: 28  Number of outliers in Grocery: 24  Number of outliers in Frozen: 43  Number of outliers in Detergents\_Paper: 30 Number of outliers in Delicatessen: 27 |

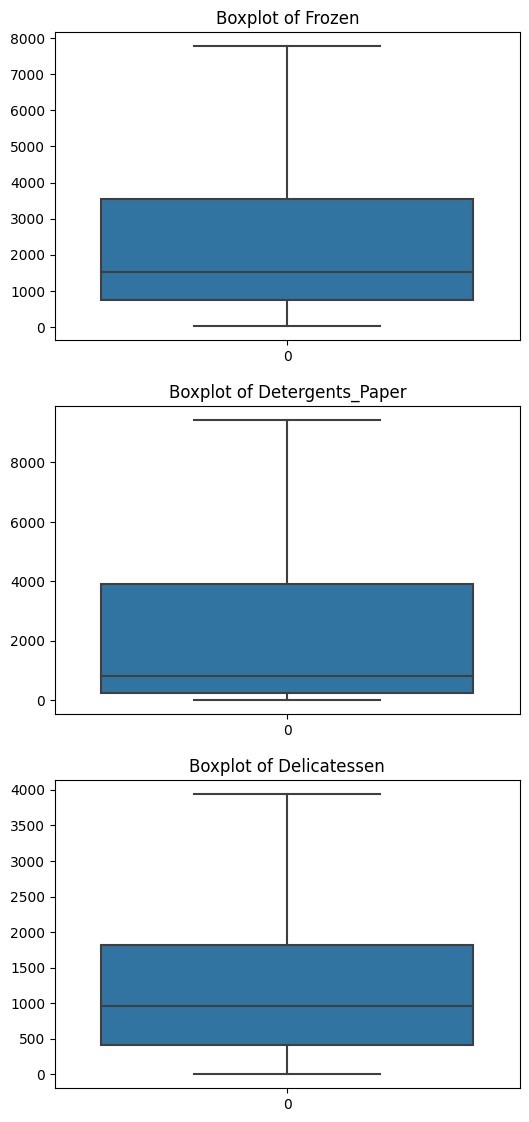
|  |
| --- |
| def handle\_outliers(dataframe, column):  Q1 = dataframe[column].quantile(0.25)  Q3 = dataframe[column].quantile(0.75)  IQR = Q3 - Q1  lower\_limit = Q1 - 1.5\*IQR upper\_limit = Q3 + 1.5\*IQR dataframe[column] = dataframe[column].apply(lambda x: upper\_limit if x > upper\_limit else lower\_limit if x < lower\_limit else x)  for column in df.columns:  handle\_outliers(df, column) |





|  |
| --- |
| def detect\_outliers(dataframe, column):  Q1 = dataframe[column].quantile(0.25)  Q3 = dataframe[column].quantile(0.75)  IQR = Q3 - Q1  outliers = dataframe[(dataframe[column] < Q1 - 1.5\*IQR) | (dataframe[column] > Q3 + 1.5\*IQR)] return outliers  for column in df.columns:  outliers = detect\_outliers(df, column)  print(f'Number of outliers in {column}: {len(outliers)}')  Number of outliers in Channel: 0  Number of outliers in Region: 0 Number of outliers in Fresh: 0  Number of outliers in Milk: 0  Number of outliers in Grocery: 0  Number of outliers in Frozen: 0  Number of outliers in Detergents\_Paper: 0 Number of outliers in Delicatessen: 0 |



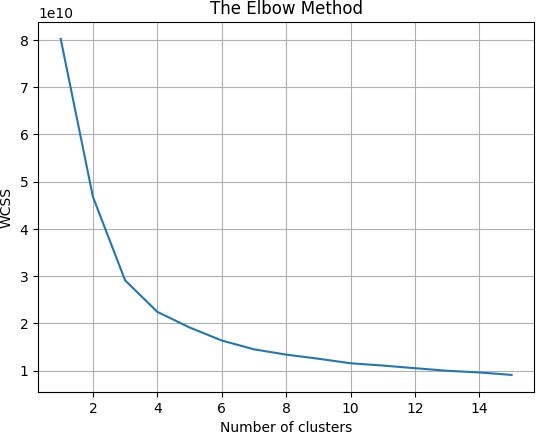


from sklearn.preprocessing import StandardScaler

scaler = StandardScaler() df\_scaled = pd.DataFrame(scaler.fit\_transform(df), columns=df.columns)

|  |
| --- |
| from sklearn.cluster import KMeans import matplotlib.pyplot as plt  # Calculate WCSS for different number of clusters wcss = [] max\_clusters = 15 for i in range(1, max\_clusters+1):  kmeans = KMeans(n\_clusters=i, init='k-means++', random\_state=42) kmeans.fit(df) wcss.append(kmeans.inertia\_)  # Plot the WCSS values plt.plot(range(1, max\_clusters+1), wcss) plt.title('The Elbow Method') plt.xlabel('Number of clusters') plt.ylabel('WCSS') plt.grid(True) plt.show() |

/usr/local/lib/python3.10/dist-packages/sklearn/cluster/\_kmeans.py:870: FutureWarning: The default value of `n\_init` will change from 10 to 'auto' warnings.warn(



from sklearn.cluster import KMeans

# Build the model

kmeans = KMeans(n\_clusters=4, init='k-means++', random\_state=42)

kmeans.fit(df)

# Get cluster labels

/usr/local/lib/python3.10/dist-packages/sklearn/cluster/\_kmeans.py:870: FutureWarning: The default value of `n\_init` will change from 10 to 'auto' warnings.warn(

/usr/local/lib/python3.10/dist-packages/sklearn/cluster/\_kmeans.py:870: FutureWarning: The default value of `n\_init` will change from 10 to 'auto' warnings.warn(

/usr/local/lib/python3.10/dist-packages/sklearn/cluster/\_kmeans.py:870: FutureWarning: The default value of `n\_init` will change from 10 to 'auto' warnings.warn(

/usr/local/lib/python3.10/dist-packages/sklearn/cluster/\_kmeans.py:870: FutureWarning: The default value of `n\_init` will change from 10 to 'auto' warnings.warn(

/usr/local/lib/python3.10/dist-packages/sklearn/cluster/\_kmeans.py:870: FutureWarning: The default value of `n\_init` will change from 10 to 'auto' warnings.warn(

/usr/local/lib/python3.10/dist-packages/sklearn/cluster/\_kmeans.py:870: FutureWarning: The default value of `n\_init` will change from 10 to 'auto' warnings.warn(

/usr/local/lib/python3.10/dist-packages/sklearn/cluster/\_kmeans.py:870: FutureWarning: The default value of `n\_init` will change from 10 to 'auto' warnings.warn(

/usr/local/lib/python3.10/dist-packages/sklearn/cluster/\_kmeans.py:870: FutureWarning: The default value of `n\_init` will change from 10 to 'auto' warnings.warn(

/usr/local/lib/python3.10/dist-packages/sklearn/cluster/\_kmeans.py:870: FutureWarning: The default value of `n\_init` will change from 10 to 'auto' warnings.warn(

/usr/local/lib/python3.10/dist-packages/sklearn/cluster/\_kmeans.py:870: FutureWarning: The default value of `n\_init` will change from 10 to 'auto' warnings.warn(

/usr/local/lib/python3.10/dist-packages/sklearn/cluster/\_kmeans.py:870: FutureWarning: The default value of `n\_init` will change from 10 to 'auto' warnings.warn(

/usr/local/lib/python3.10/dist-packages/sklearn/cluster/\_kmeans.py:870: FutureWarning: The default value of `n\_init` will change from 10 to 'auto' warnings.warn(

/usr/local/lib/python3.10/dist-packages/sklearn/cluster/\_kmeans.py:870: FutureWarning: The default value of `n\_init` will change from 10 to 'auto' warnings.warn(

/usr/local/lib/python3.10/dist-packages/sklearn/cluster/\_kmeans.py:870: FutureWarning: The default value of `n\_init` will change from 10 to 'auto' warnings.warn(

cluster\_labels = kmeans.labels\_

# Add cluster labels to your original dataframe df['Cluster'] = cluster\_labels

/usr/local/lib/python3.10/dist-packages/sklearn/cluster/\_kmeans.py:870: FutureWarning: The default value of `n\_init` will change from 10 to 'auto' warnings.warn(

Cluster Sizes:

3 176

1. 112
2. 94
3. 58

Name: Cluster, dtype: int64

Cluster 0

Channel Region Fresh Milk Grocery \

count 112.000000 112.000000 112.000000 112.000000 112.000000 mean 1.214286 2.535714 16051.205357 3135.813616 4211.589286 std 0.412170 0.781873 3763.633078 2524.464860 3150.441587 min 1.000000 1.000000 10379.000000 134.000000 3.000000 25% 1.000000 2.000000 12419.750000 1283.500000 1970.500000

50% 1.000000 3.000000 16195.000000 2252.000000 3203.000000

|  |
| --- |
| from sklearn.decomposition import PCA import matplotlib.pyplot as plt  # Apply PCA and fit the features selected pca = PCA(n\_components=2) principalComponents = pca.fit\_transform(df.drop('Cluster', axis=1))  # Create a DataFrame with the two components  PCA\_components = pd.DataFrame(principalComponents, columns=['Principal Component 1', 'Principal Component 2']) |

|  |
| --- |
| df['Cluster'] = kmeans.labels\_  # Check the size of each cluster print("Cluster Sizes:\n", df['Cluster'].value\_counts())  # Check the characteristics of each cluster for i in range(4):  print("\nCluster ", i) print(df[df['Cluster'] == i].describe()) |

75% 1.000000 3.000000 18830.250000 4537.000000 5700.250000 max 2.000000 3.000000 24929.000000 15676.125000 14982.000000



Frozen Detergents\_Paper Delicatessen Cluster

count 112.000000 112.000000 112.000000 112.0 mean 2988.859375 994.785714 1229.573661 0.0 std 2531.352938 1245.589613 963.527882 0.0 min 118.000000 3.000000 51.000000 0.0 25% 1018.750000 188.500000 514.250000 0.0

50% 2157.500000 456.500000 879.000000 0.0

75% 4276.000000 1404.000000 1804.500000 0.0 max 7772.250000 6707.000000 3938.250000 0.0

Cluster 1

Channel Region Fresh Milk Grocery \

count 94.000000 94.000000 94.000000 94.000000 94.000000 mean 1.893617 2.489362 5331.893617 10454.450798 17196.140957 std 0.309980 0.799794 5111.448153 3937.245330 4905.345002 min 1.000000 1.000000 18.000000 1266.000000 8852.000000 25% 2.000000 2.000000 1409.500000 7576.000000 12563.250000

50% 2.000000 3.000000 4047.000000 10601.000000 16596.000000

75% 2.000000 3.000000 7870.500000 14316.500000 22288.500000 max 2.000000 3.000000 22925.000000 15676.125000 23409.875000

Frozen Detergents\_Paper Delicatessen Cluster count 94.000000 94.000000 94.000000 94.0 mean 1496.428191 6936.898936 1547.364362 1.0 std 1538.882840 2383.035957 1176.131062 0.0 min 25.000000 241.000000 3.000000 1.0 25% 438.500000 5274.250000 680.000000 1.0

50% 973.000000 6931.500000 1366.500000 1.0 75% 1900.000000 9419.875000 2157.750000 1.0 max 7772.250000 9419.875000 3938.250000 1.0

Cluster 2

Channel Region Fresh Milk Grocery \

count 58.000000 58.000000 58.000000 58.000000 58.000000 mean 1.172414 2.655172 32136.810345 5973.515086 7309.012931 std 0.381039 0.714554 5122.024937 4808.223223 5915.174661 min 1.000000 1.000000 22647.000000 286.000000 471.000000 25% 1.000000 3.000000 27207.500000 2393.000000 2726.250000

50% 1.000000 3.000000 31664.000000 4347.000000 5259.500000 75% 1.000000 3.000000 37642.750000 7829.500000 9344.000000

|  |
| --- |
| # Concatenate the clusters labels to the DataFrame PCA\_components['Cluster'] = df['Cluster']  # Plot the clustered dataset plt.figure(figsize=(8,6)) plt.scatter(PCA\_components['Principal Component 1'], PCA\_components['Principal Component 2'], c=PCA\_components['Cluster']) plt.title('Clusters in PCA 2D Space') plt.xlabel('Principal Component 1') plt.ylabel('Principal Component 2') plt.colorbar(label='Cluster') plt.show() |

