



Vidyavardhini's College of Engineering & Technology
Department of Computer Engineering

Experiment No. 5
Apply appropriate Unsupervised Learning Technique on the Wholesale Customers Dataset
Date of Performance: 13-09-2023
Date of Submission: 05-10-2023



Vidyavardhini's College of Engineering & Technology

Department of Computer Engineering

Aim: Apply appropriate Unsupervised Learning Technique on the Wholesale Customers Dataset.

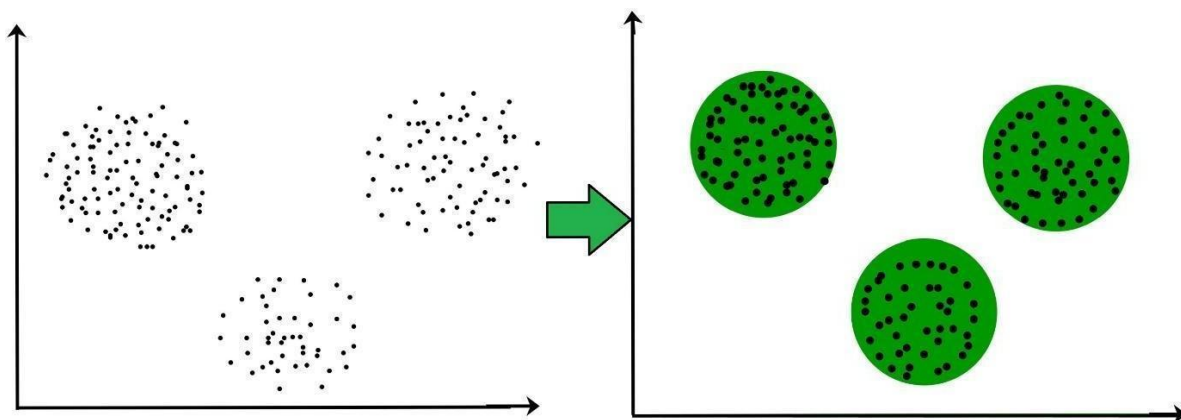
Objective: Able to perform various feature engineering tasks, apply Clustering Algorithm on the given dataset.

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 :

Use of the clustered data

Medical Research: Researchers might cluster patients with similar health conditions or risk factors to study the effectiveness of treatments or interventions for specific subgroups.

Image Recognition: In computer vision, clustered data can help identify objects in images by grouping similar features or patterns together.

Social Network Analysis: Analyzing clustered data can reveal communities or groups of users with similar interests in social networks, which is valuable for targeted advertising or understanding network dynamics.

Geographic Analysis: Spatial data can be clustered to identify regions with similar characteristics, such as income levels or population density, for urban planning or resource allocation.

Different groups of customers, the customer segments, may be affected differently by a specific delivery scheme

Demographic Segmentation:

Age groups: Younger customers might prioritize speed, while older customers may value reliability.

Income levels: High-income customers might prefer premium, same-day delivery services.

Geographic Location:

Different regions or countries may have varying preferences for delivery options.

Product Type:

Some products, like groceries, may require faster delivery, while others, like furniture, can be delivered over a longer timeframe.

Urban vs. Rural Customers:

Urban customers might prefer faster delivery options due to proximity to distribution centers.

Rural customers may prioritize cost-effective or reliable delivery due to longer transit times.

Frequent Shoppers vs. Occasional Shoppers:

Frequent shoppers may value subscription-based delivery services or loyalty programs.

Occasional shoppers might prefer pay-as-you-go options with lower upfront costs.

exp5

Imports

```
[1]: import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

```
[2]: df=pd.read_csv("/content/Wholesale customers data.csv")
```

```
[3]: df.head()
```

```
[3]:
```

	Channel	Region	Fresh	Milk	Grocery	Frozen	Detergents_Paper	Delicassen
0	2	3	12669	9656	7561	214	2674	1338
1	2	3	7057	9810	9568	1762	3293	1776
2	2	3	6353	8808	7684	2405	3516	7844
3	1	3	13265	1196	4221	6404	507	1788
4	2	3	22615	5410	7198	3915	1777	5185

```
[4]: df.shape
```

```
[4]: (440, 8)
```

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

```
<class 'pandas.core.frame.DataFrame'>
```

```
RangeIndex: 440 entries, 0 to 439
```

```
Data columns (total 8 columns):
```

#	Column	Non-Null Count	Dtype
0	Channel	440 non-null	int64
1	Region	440 non-null	int64
2	Fresh	440 non-null	int64
3	Milk	440 non-null	int64
4	Grocery	440 non-null	int64
5	Frozen	440 non-null	int64
6	Detergents_Paper	440 non-null	int64
7	Delicassen	440 non-null	int64

```
dtypes: int64(8)
memory usage: 27.6 KB
```

```
[6]: df.describe()
```

```
[6]:
```

	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

	Frozen	Detergents_Paper	Delicassen
count	440.000000	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

```
[7]: df.dtypes
```

```
[7]: Channel          int64
      Region          int64
      Fresh          int64
      Milk           int64
      Grocery        int64
      Frozen         int64
      Detergents_Paper int64
      Delicassen     int64
      dtype: object
```

```
[8]: df.isnull().sum()
```

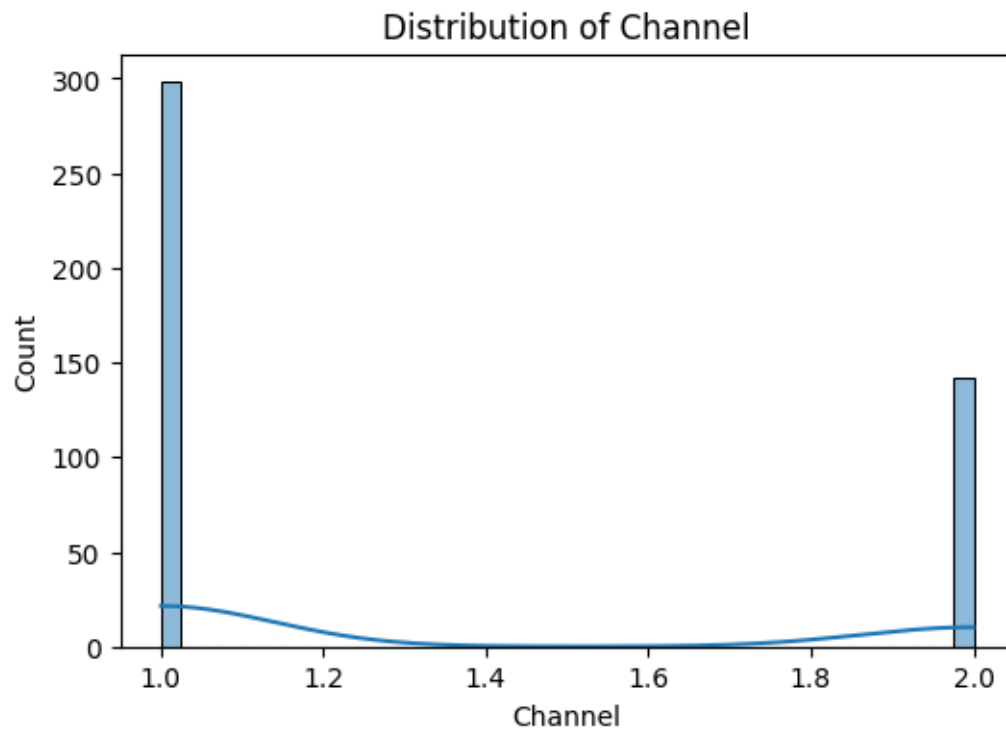
```
[8]: Channel          0
      Region          0
      Fresh           0
      Milk            0
      Grocery         0
      Frozen          0
      Detergents_Paper 0
      Delicassen      0
```

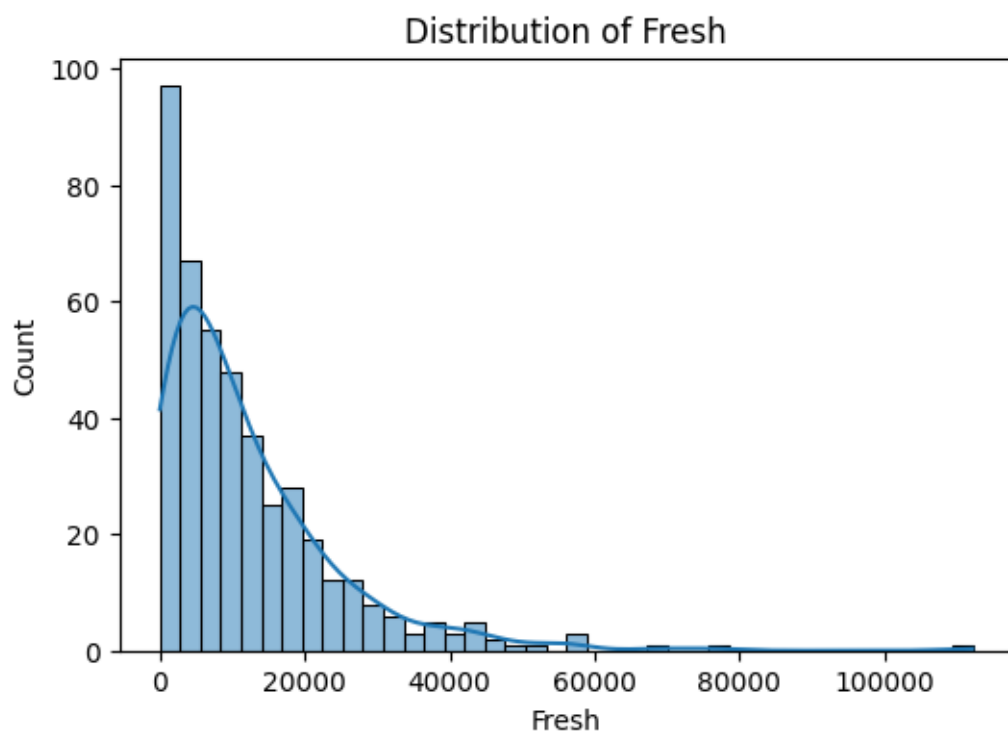
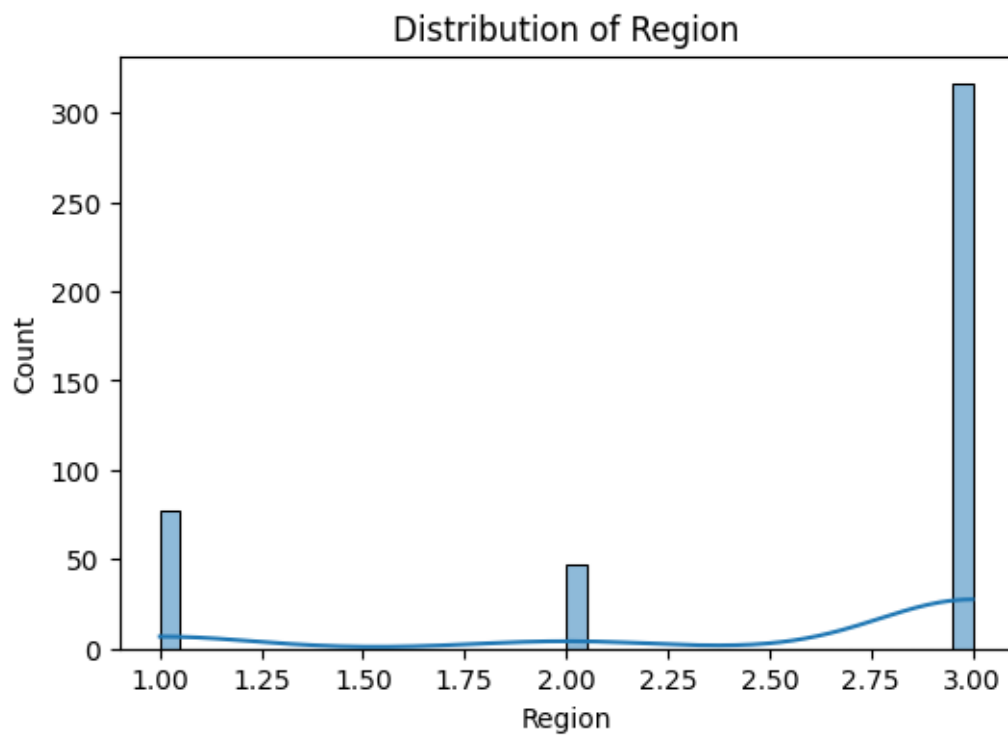
dtype: int64

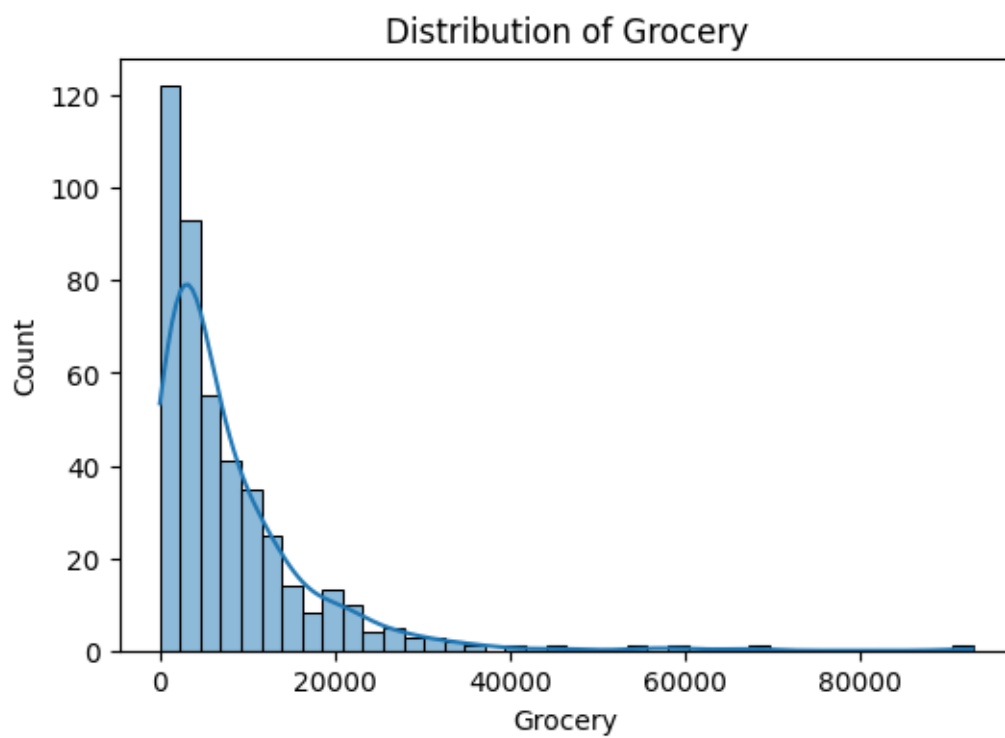
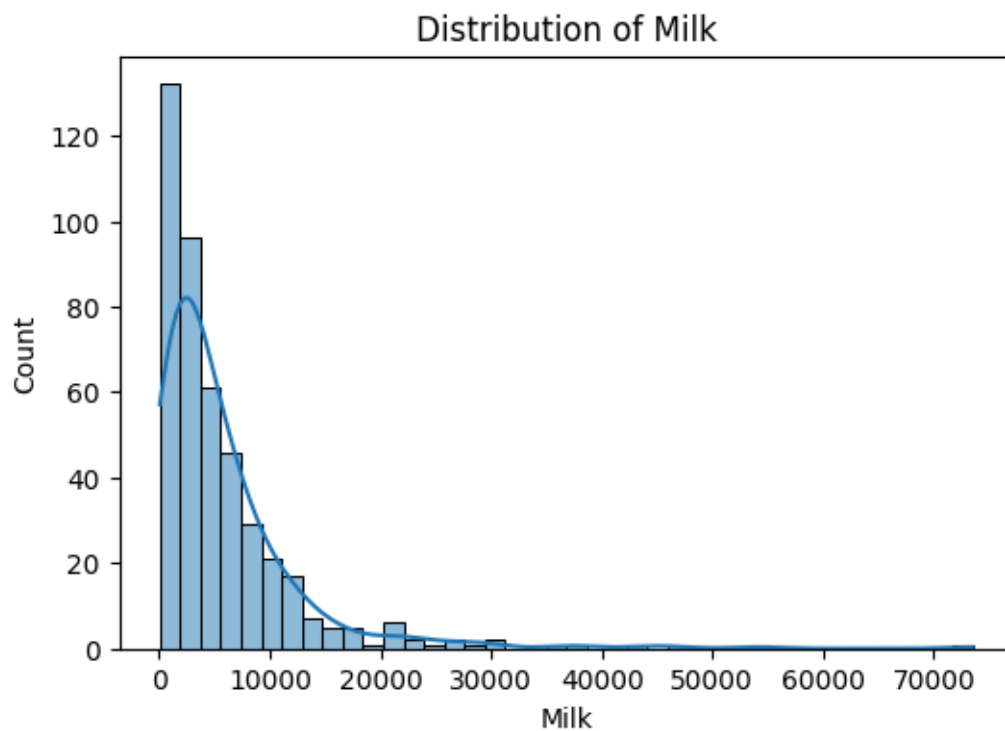
```
[9]: df.columns
```

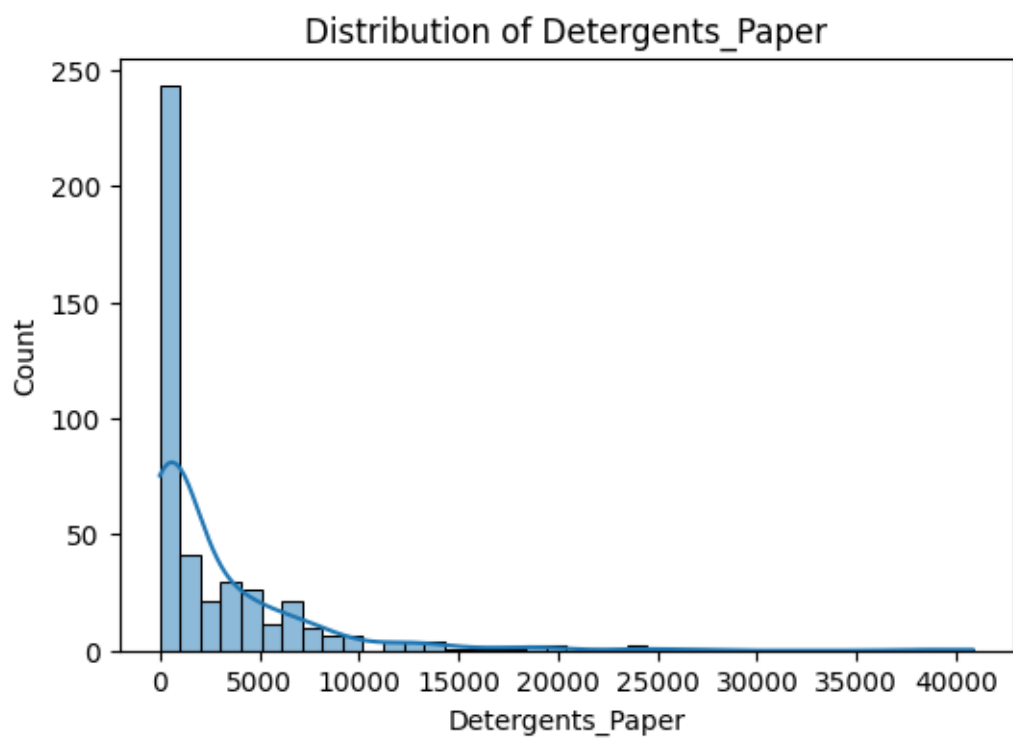
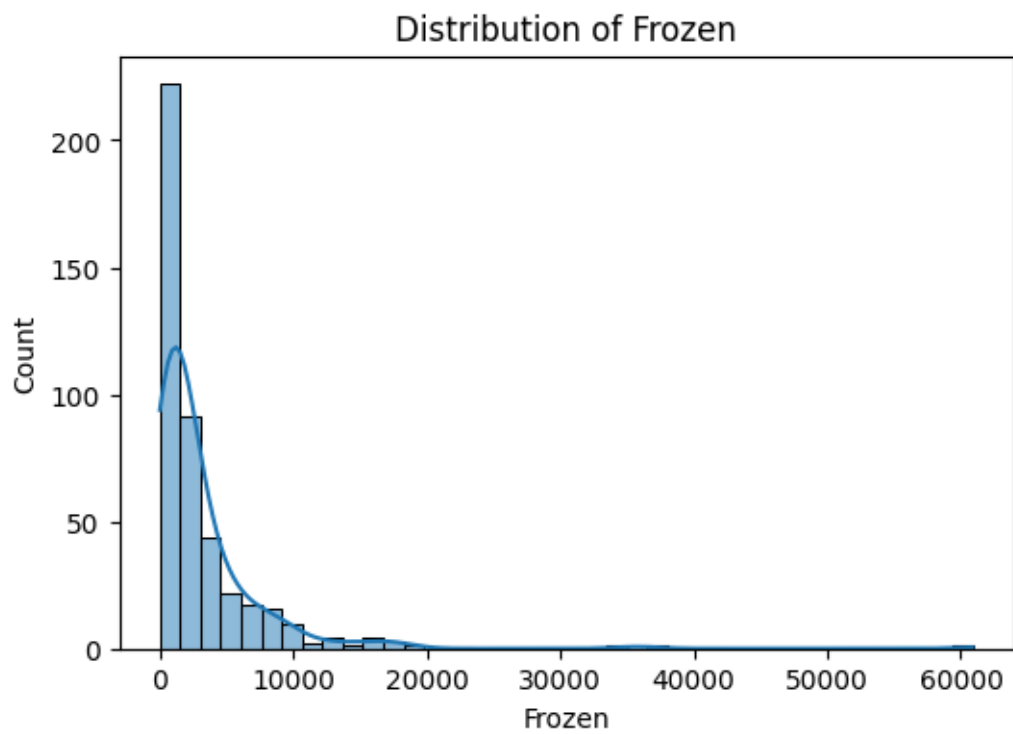
```
[9]: Index(['Channel', 'Region', 'Fresh', 'Milk', 'Grocery', 'Frozen',  
        'Detergents_Paper', 'Delicassen'],  
        dtype='object')
```

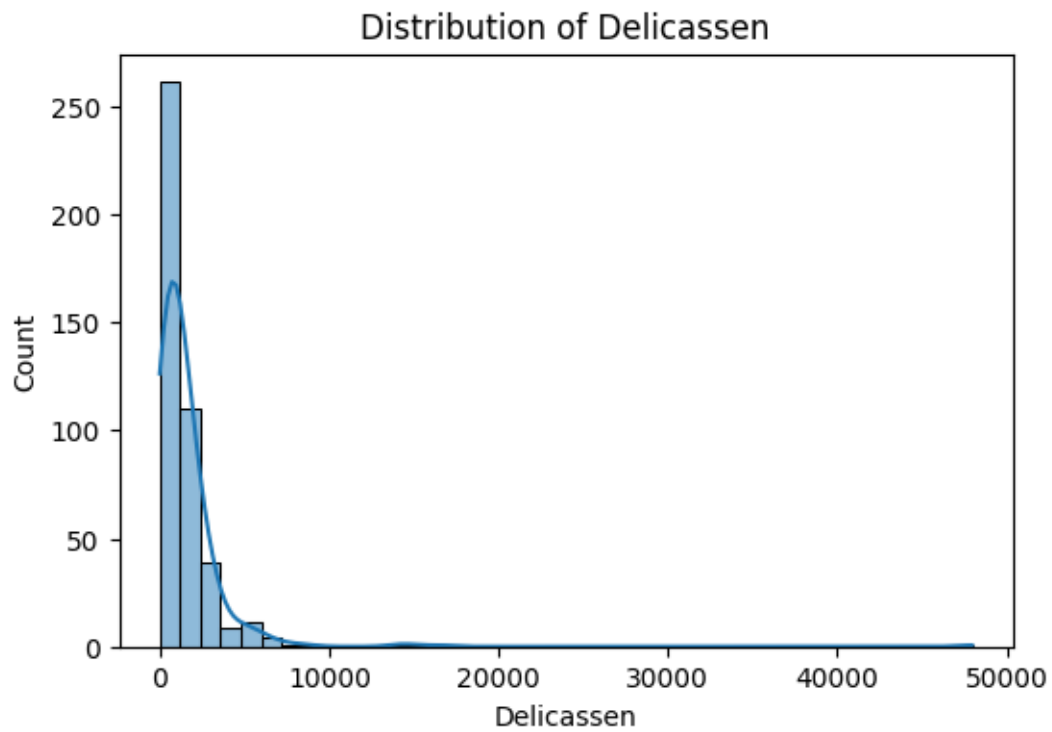
```
[10]: for column in df.columns:  
    plt.figure(figsize=(6, 4))  
    sns.histplot(df[column], bins=40, kde=True)  
    plt.title(f'Distribution of {column}')  
    plt.show()
```





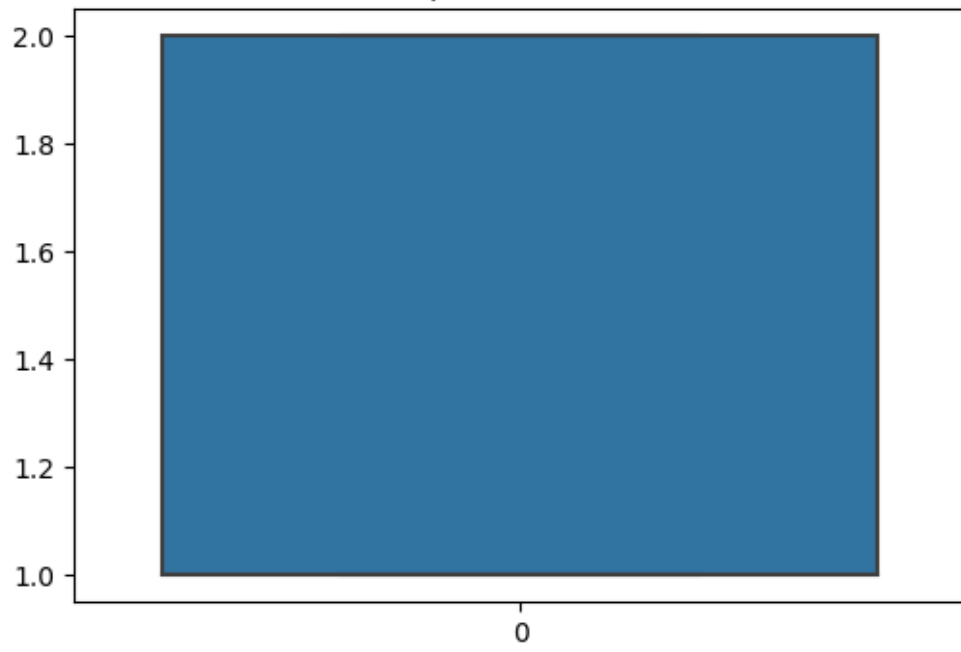




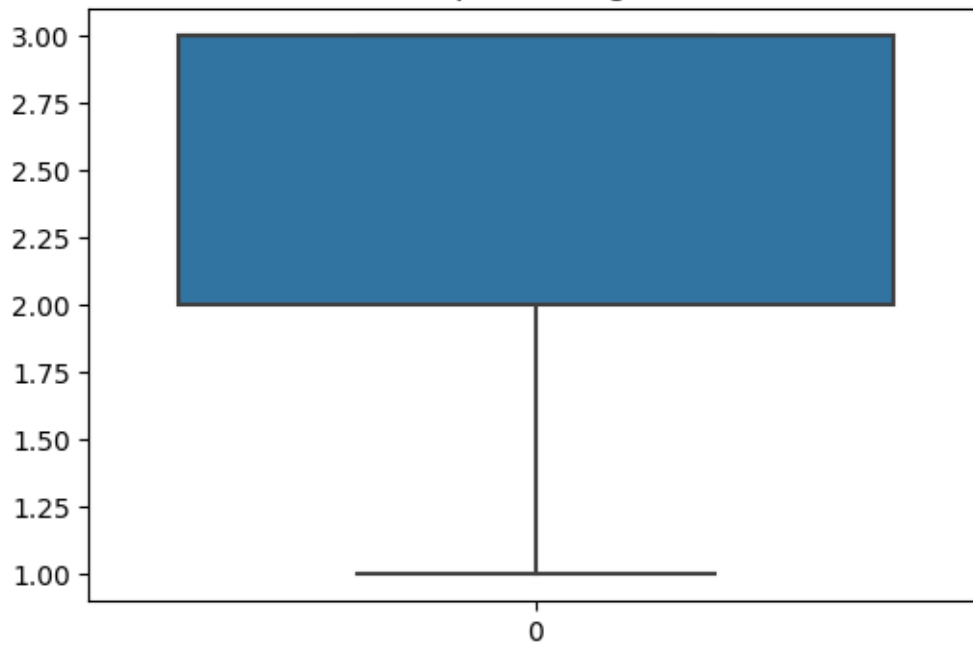


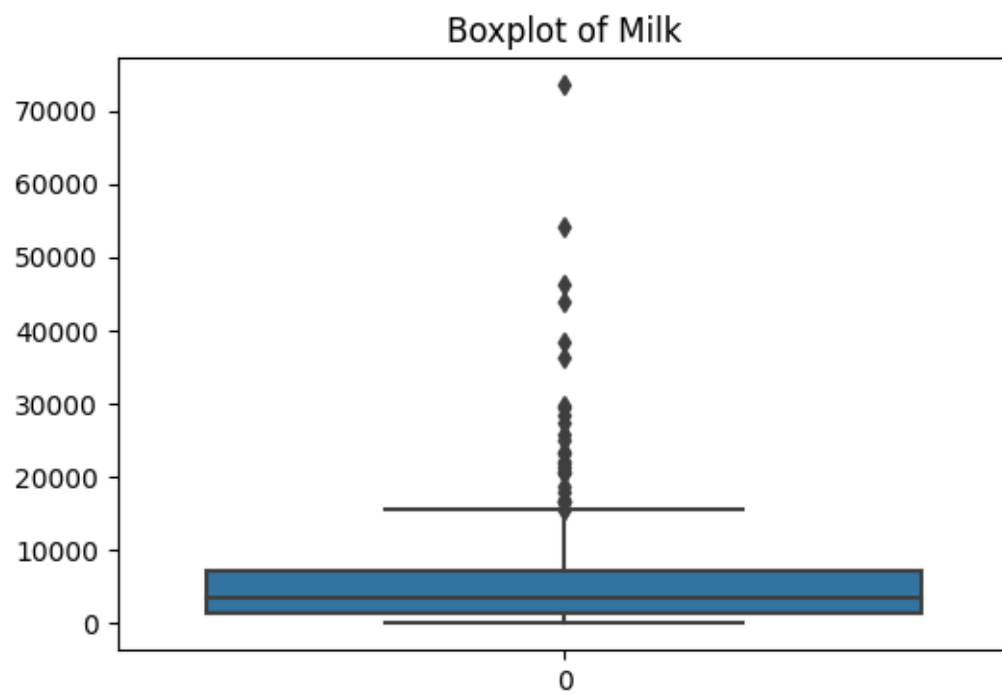
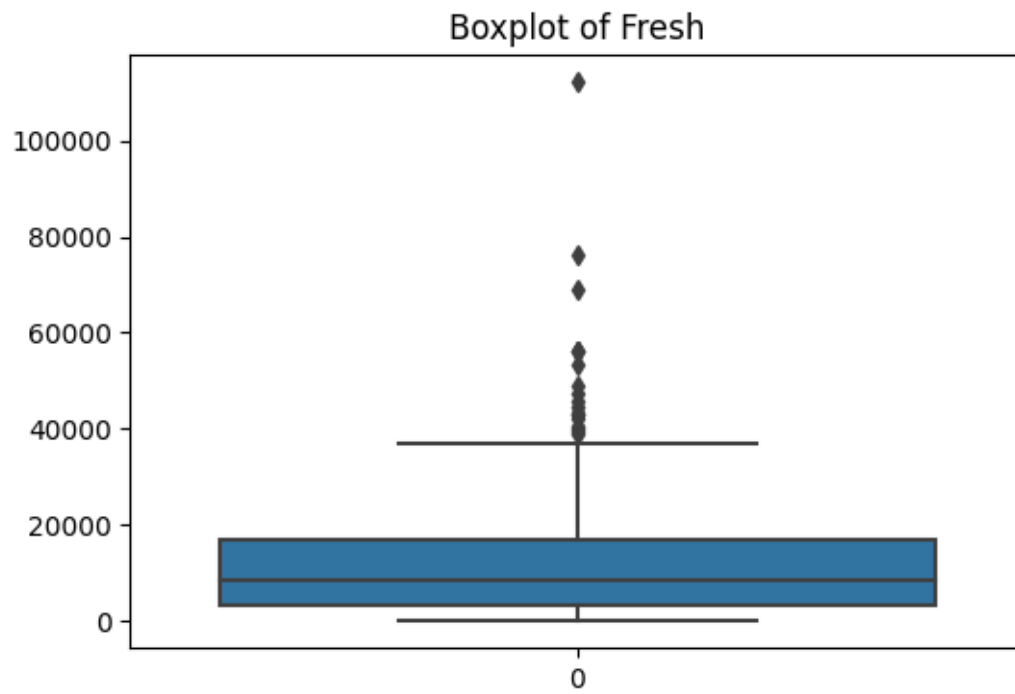
```
[11]: for column in df.columns:
plt.figure(figsize=(6, 4))
sns.boxplot(df[column])
plt.title(f'Boxplot of {column}')
plt.show()
```

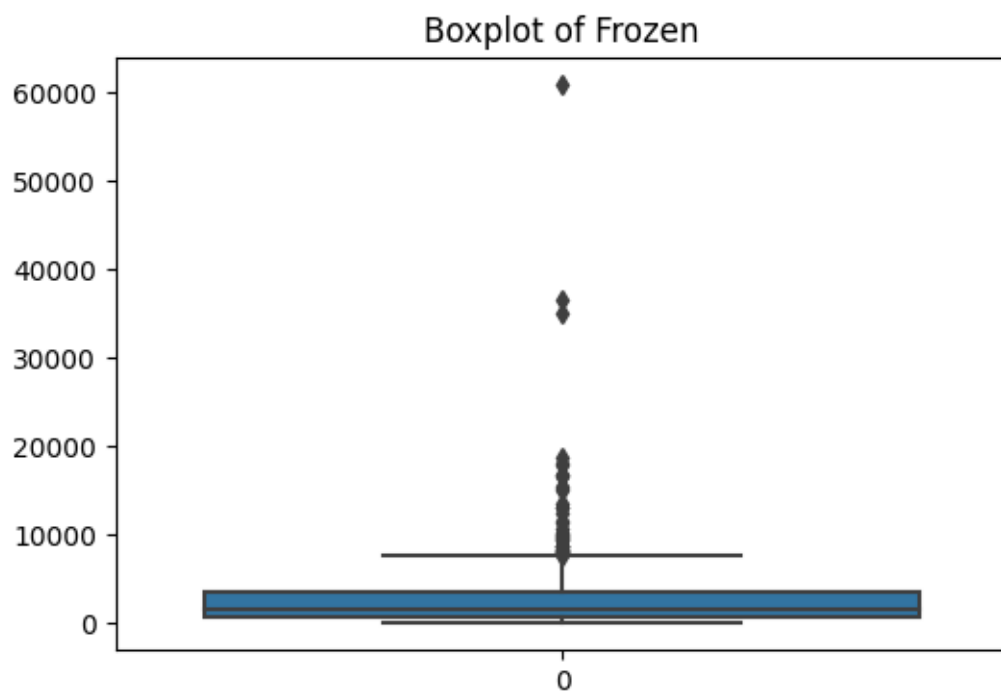
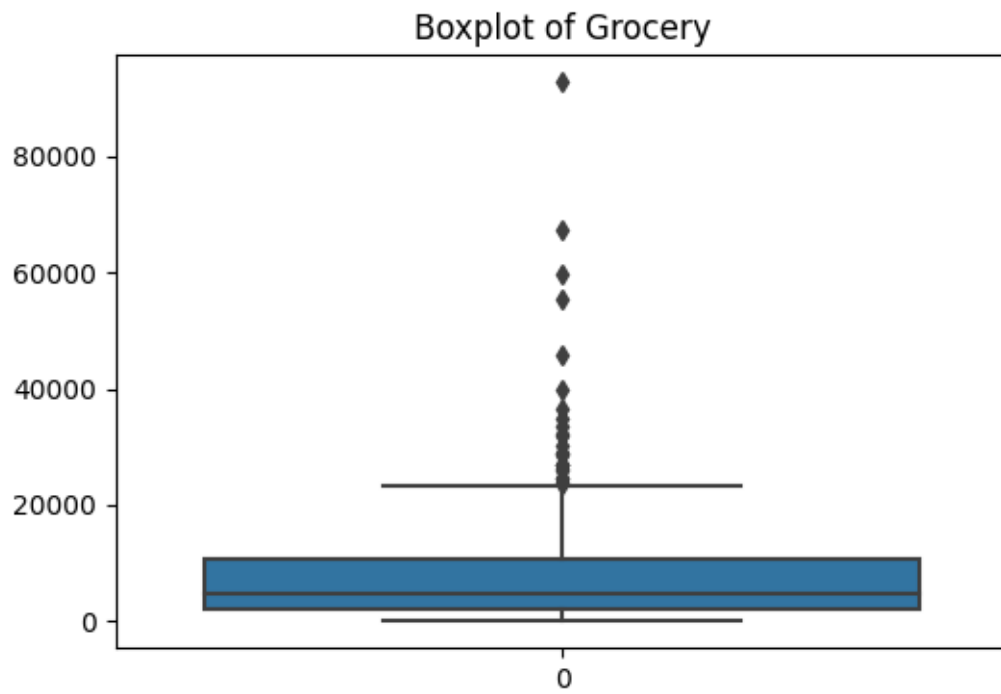
Boxplot of Channel

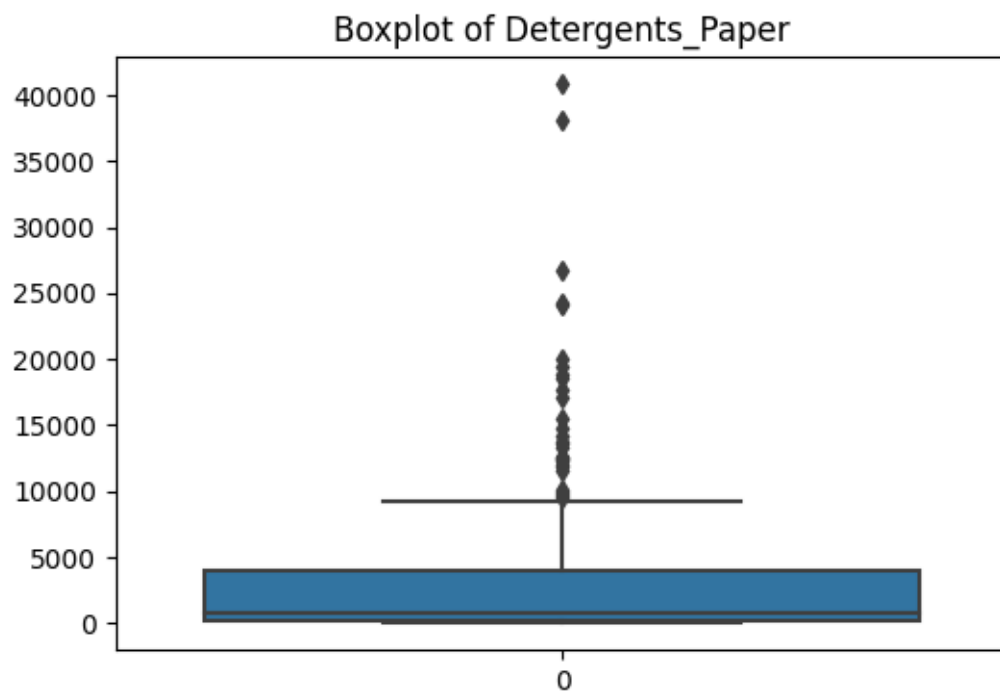


Boxplot of Region

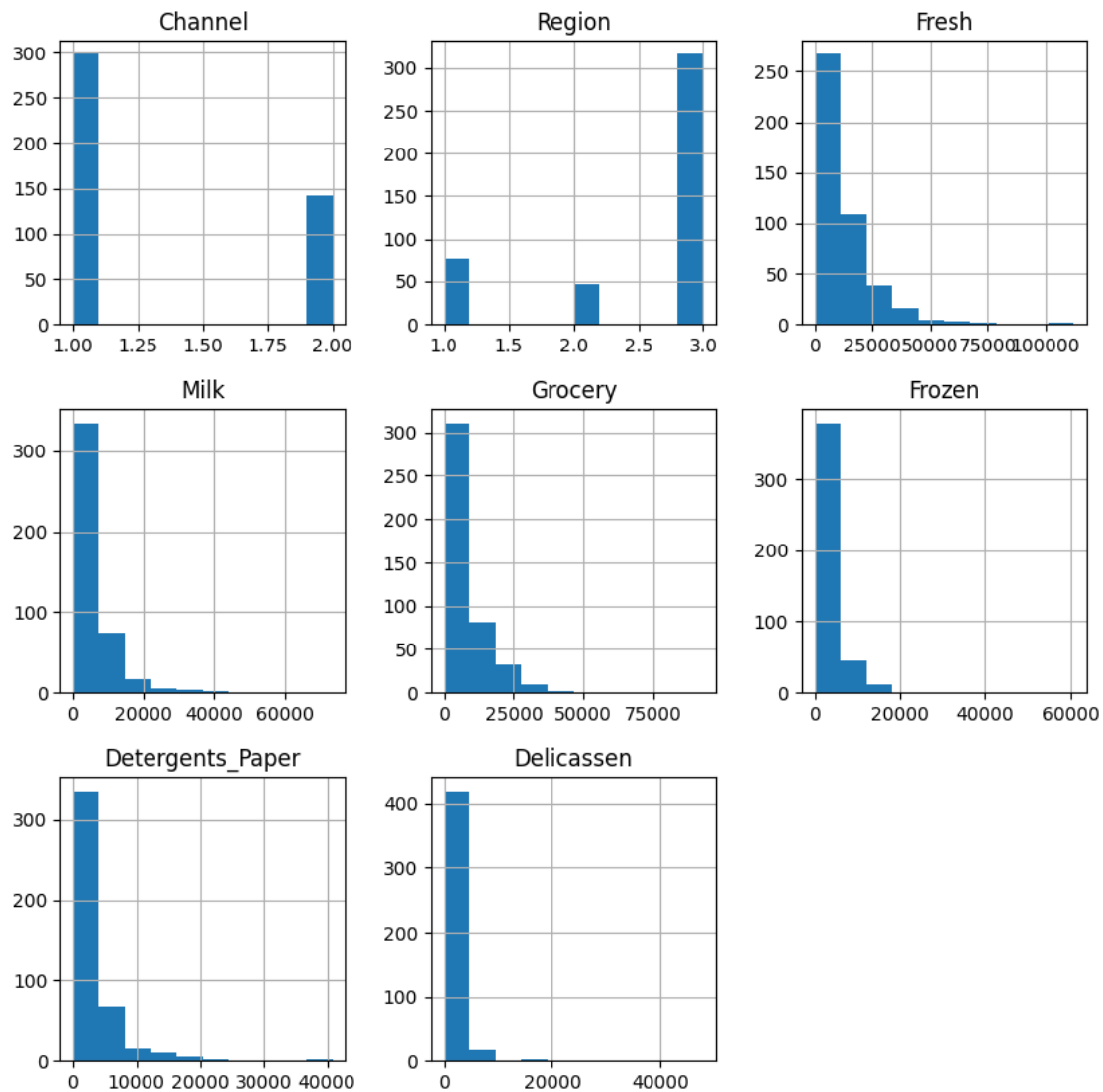








```
[13]: df.hist(bins=10, figsize=(10, 10))
plt.show()
```



```
[14]: 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)
```



```
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
```

```
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_)
```

[illegible]

```

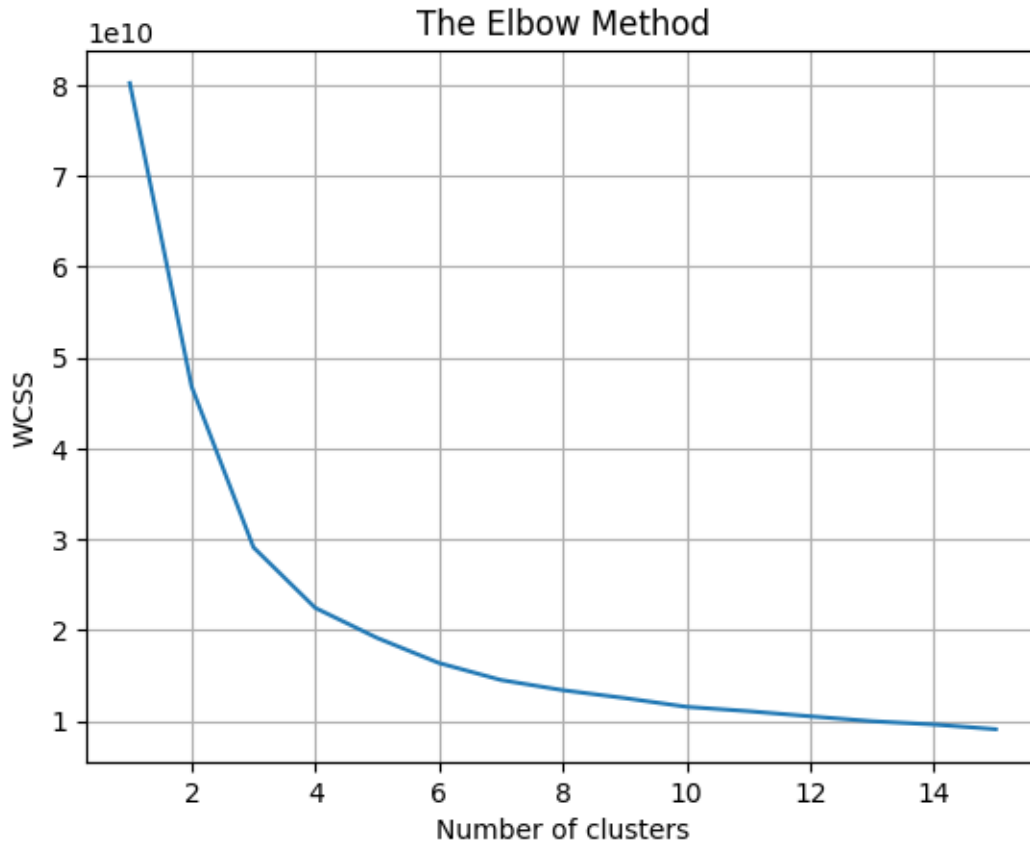
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870:
FutureWarning: The default value of `n_init` will change from 10 to 'auto' in
1.4. Set the value of `n_init` explicitly to suppress the warning
    warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870:
FutureWarning: The default value of `n_init` will change from 10 to 'auto' in
1.4. Set the value of `n_init` explicitly to suppress the warning
    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' in
1.4. Set the value of `n_init` explicitly to suppress the warning
    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' in
1.4. Set the value of `n_init` explicitly to suppress the warning
    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' in
1.4. Set the value of `n_init` explicitly to suppress the warning
    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' in
1.4. Set the value of `n_init` explicitly to suppress the warning
    warnings.warn(

```

```

[19]: 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()

```



```
[20]: from sklearn.cluster import KMeans
```

```
[21]: kmeans = KMeans(n_clusters=4, init='k-means++', random_state=42)
      kmeans.fit(df)
```

```
/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' in
1.4. Set the value of `n_init` explicitly to suppress the warning
  warnings.warn(
```

```
[21]: KMeans(n_clusters=4, random_state=42)
```

```
[22]: cluster_labels = kmeans.labels_
      df['Cluster'] = cluster_labels
      print(df['Cluster'].unique())
```

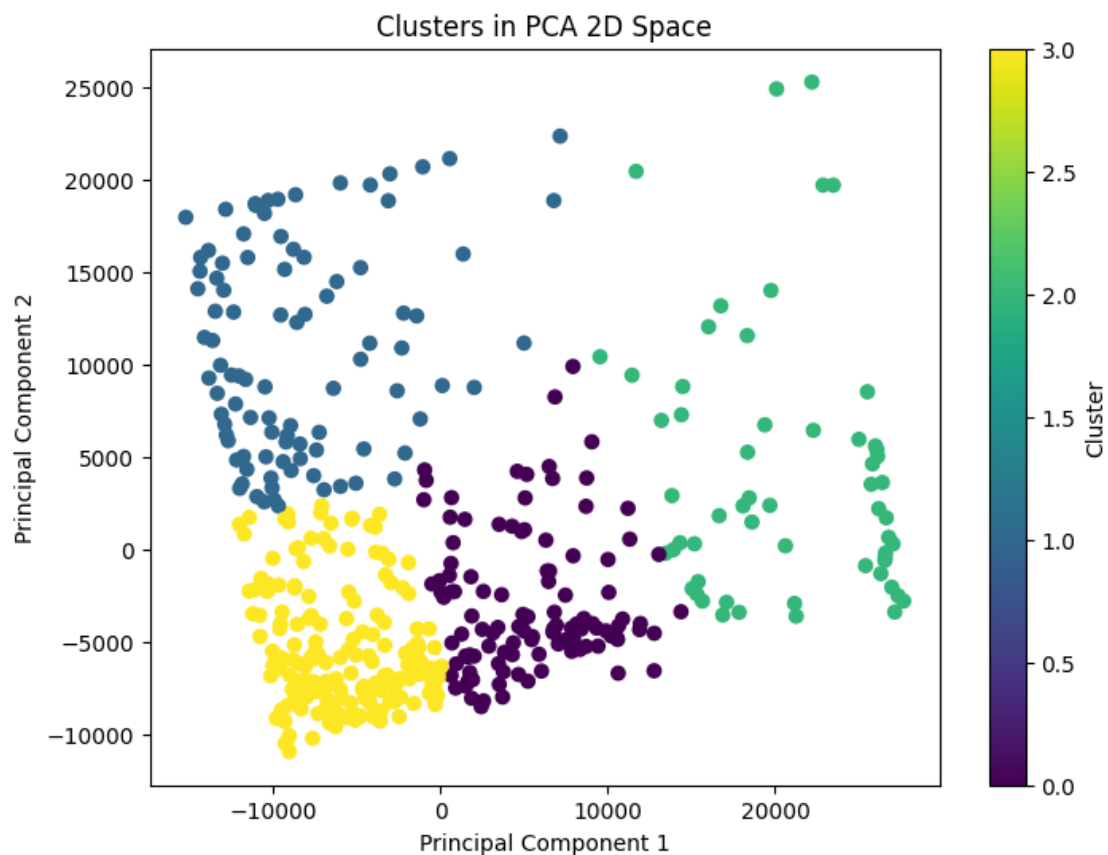
```
[0 1 3 2]
```

```
[23]: from sklearn.decomposition import PCA
      import matplotlib.pyplot as plt
```

```
[24]: pca = PCA(n_components=2)
principalComponents = pca.fit_transform(df.drop('Cluster', axis=1))

[25]: PCA_components = pd.DataFrame(principalComponents, columns=['Principal_
↪Component 1', 'Principal Component 2'])
PCA_components['Cluster'] = df['Cluster']

[26]: 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()
```



```
[27]: cluster_means = df.groupby('Cluster').mean()
cluster_means = cluster_means.transpose()
for feature in cluster_means.index:
    cluster_means.loc[feature].plot(kind='bar', figsize=(7,5))
```

```
plt.title(feature)
plt.ylabel('Mean Value')
plt.xticks(ticks=range(4), labels=['Cluster 0', 'Cluster 1', 'Cluster 2', 'Cluster 3'])
plt.show()
```

