Experiment No. 5
Apply appropriate Unsupervised Learning Technique on the Wholesale Customers Dataset
Date of Performance:21-08-2023
Date of Submission:25-09-2023

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

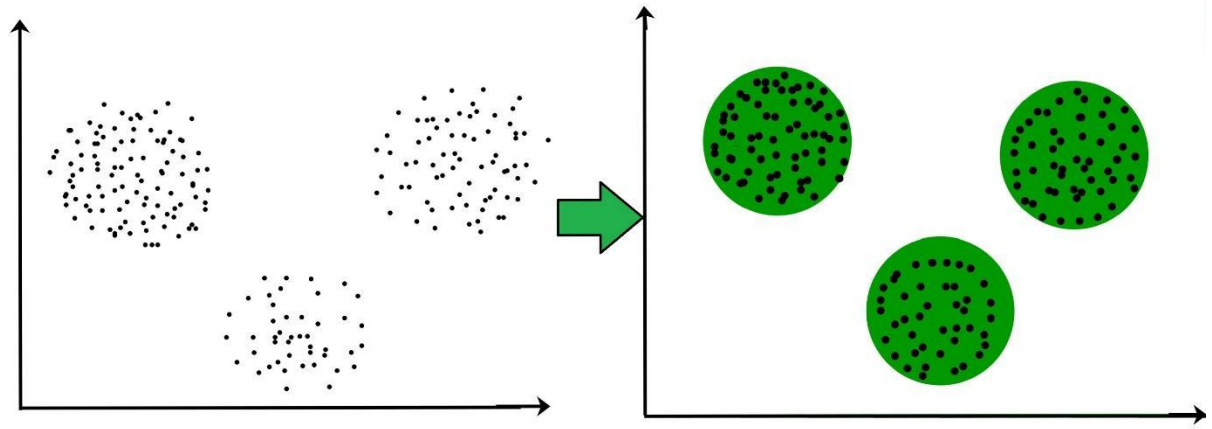
Objective: Able to perform various feature engineering tasks, apply Clustering Algorithm on the given dataset.

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)

Code:

Conclusion:

Use of the clustered data

● **Customer Segmentation:** Clustered data helps you understand different customer segments based on their purchasing behavior. We can design marketing campaigns that are more relevant to each cluster's preferences.

● **Product Recommendations:** By knowing which products are frequently purchased together within each cluster, we can make personalized product recommendations to customers.

● **Inventory Management:** Clustering can help optimize inventory management by ensuring that the right products are stocked in the right quantities to meet the preferences of each cluster.

● **Supply Chain Optimization**: We can optimize our supply chain operations by tailoring delivery schedules and routes to each cluster's needs.

● **Customer Retention**: By understanding the characteristics of each cluster, we can develop retention strategies that are more likely to resonate with specific customer groups.

● **Market Expansion:** Clustering can also help identify potential new markets or customer segments that are similar to existing clusters.

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

● **High-Value Customers**: If a delivery scheme offers premium or expedited delivery options, high-value customers who value convenience and are willing to pay more may respond positively.

● **Budget-Conscious Customers**: Customers in this segment may be more price-sensitive and may prefer a cost-effective or free standard delivery scheme.

● **Bulk Buyers**: If there is a segment of customers who prefer bulk purchases, they might benefit from delivery schemes that offer discounts for larger orders or specialized bulk delivery options.

● **Frequent Shoppers:** Customers who shop frequently may benefit from subscription-based or loyalty-based delivery schemes. These schemes can encourage repeat purchases and loyalty.

```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))


import pandas as pd

# Define a function to load the data
def load_data(path):
    try:
        df = pd.read_csv(path)
        print("Data loaded successfully!")
        return df
    except Exception as e:
        print(f"An error occurred: {e}")
        return None

# Path to the data file
path = '/content/Wholesale customers data.csv'

# Load the data
df = load_data(path)

# Display the first few rows of the DataFrame
print(df.head())
```

```
Data loaded successfully!
   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
```

```
print("Column names:")
print(df.columns)
```

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

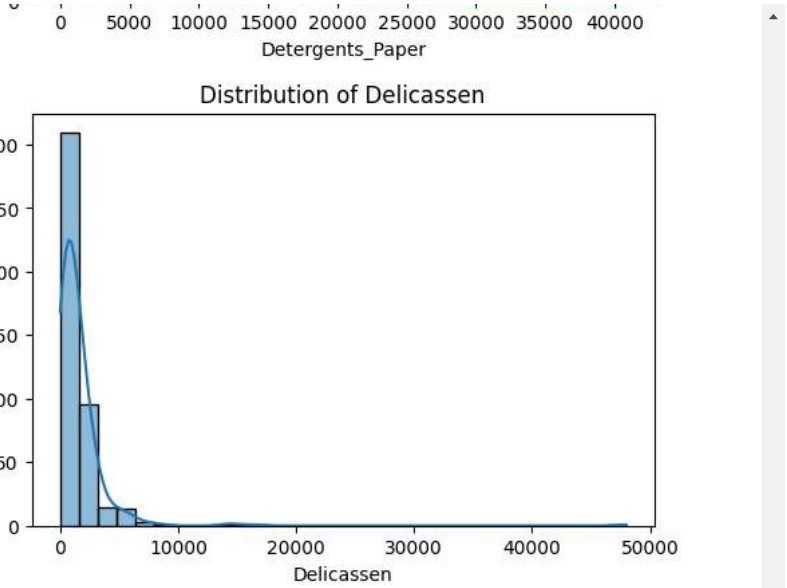
```
# Print the data types of each column
print("Data types:")
print(df.dtypes)
```

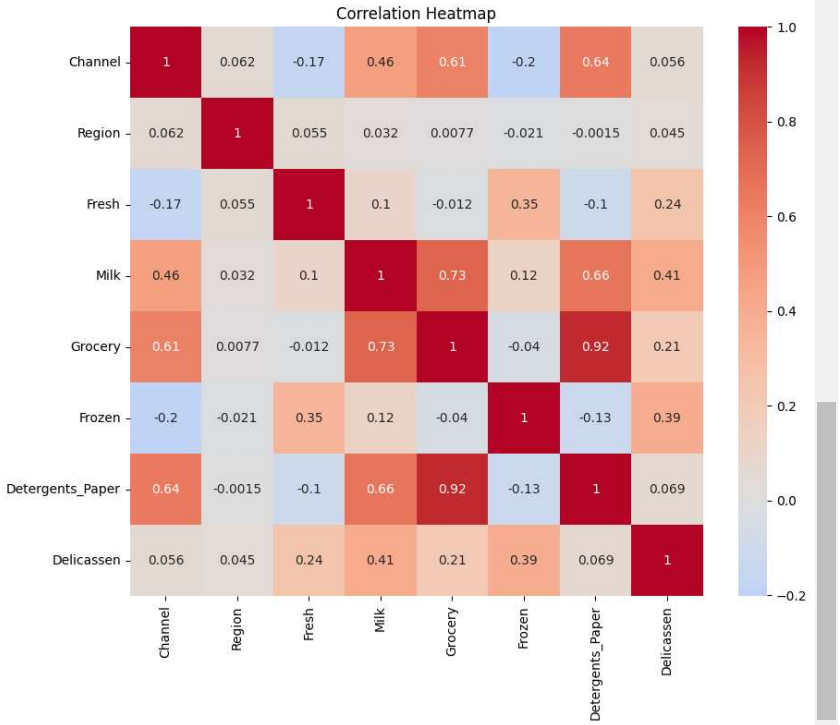
```
Data types:
Channel             int64
Region              int64
Fresh               int64
Milk                int64
Grocery             int64
Frozen              int64
Detergents_Paper    int64
Delicassen          int64
dtype: object
```

```
# Check for missing values
print("Missing values per column:")
print(df.isnull().sum())
```

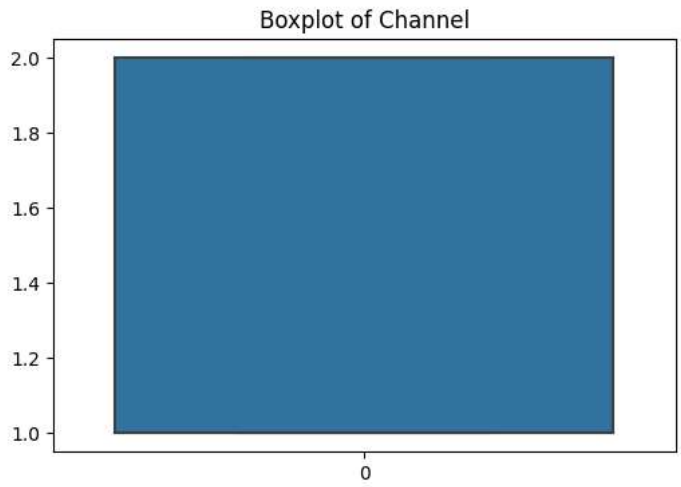
```
Missing values per column:
Channel             0
Region              0
Fresh               0
Milk                0
Grocery             0
Frozen              0
Detergents_Paper    0
Delicassen          0
dtype: int64
```

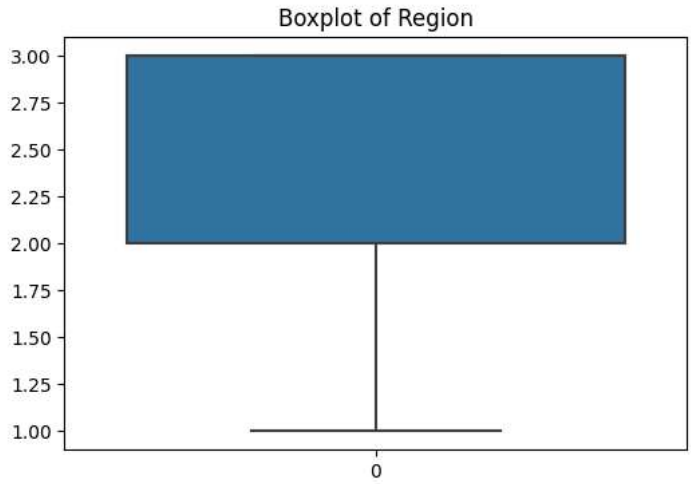
```
import matplotlib.pyplot as plt
import seaborn as sns

# Check descriptive statistics
print("Descriptive Statistics:")
print(df.describe())

# Check for duplicates
print("Number of duplicate rows: ", df.duplicated().sum())

# Distribution plots for each feature
for column in df.columns:
    plt.figure(figsize=(6, 4))
    sns.histplot(df[column], bins=30, kde=True)
    plt.title(f'Distribution of {column}')
    plt.show()

# Heatmap for correlation between variables
plt.figure(figsize=(10, 8))
sns.heatmap(df.corr(), annot=True, cmap='coolwarm', center=0)
plt.title('Correlation Heatmap')
plt.show()
```

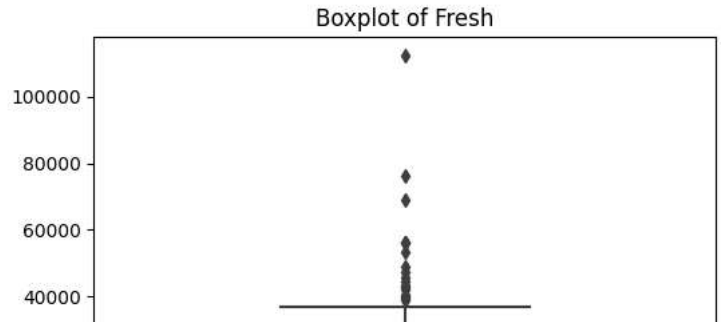

0 5000 10000 15000 20000 25000 30000 35000 40000
Detergents_Paper
Distribution of Delicassen
Count
0 50 100 150 200 250 300
0 10000 20000 30000 40000 50000
Delicassen


Correlation Heatmap
Channel 1 0.062 -0.17 0.46 0.61 -0.2 0.64 0.056
Region 0.062 1 0.055 0.032 0.0077 -0.021 -0.0015 0.045
Fresh -0.17 0.055 1 0.1 -0.012 0.35 -0.1 0.24
Milk 0.46 0.032 0.1 1 0.73 0.12 0.66 0.41
Grocery 0.61 0.0077 -0.012 0.73 1 -0.04 0.92 0.21
Frozen -0.2 -0.021 0.35 0.12 -0.04 1 -0.13 0.39
Detergents_Paper 0.64 -0.0015 -0.1 0.66 0.92 -0.13 1 0.069
Delicassen 0.056 0.045 0.24 0.41 0.21 0.39 0.069 1
Channel Region Fresh Milk Grocery Frozen Detergents_Paper Delicassen
1.0 0.8 0.6 0.4 0.2 0.0 −0.2

```
# checking for outliers
import seaborn as sns
import matplotlib.pyplot as plt

# Draw 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

# Detect and print 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)}')
```


Boxplot of Channel
2.0
1.8
1.6
1.4
1.2
1.0
0


Boxplot of Region
3.00
2.75
2.50
2.25
2.00
1.75
1.50
1.25
1.00
0


Boxplot of Fresh
100000
80000
60000
40000

```
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)

# Handle outliers for each feature
for column in df.columns:
    handle_outliers(df, column)


# Import necessary libraries
import seaborn as sns
import matplotlib.pyplot as plt

# Draw 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()

    # Draw distribution plots for all features
    for column in df.columns:
        plt.figure(figsize=(6, 4))
        sns.histplot(df[column], bins=30, kde=True)
        plt.title(f'Distribution 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

    # Detect and print 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)}')
```

```
     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 Delicassen: 0
```

```
# Check descriptive statistics
print("Descriptive Statistics:")
print(df.describe())

# Check for duplicates
print("Number of duplicate rows: ", df.duplicated().sum())

# Distribution plots for each feature
for column in df.columns:
    plt.figure(figsize=(6, 4))
    sns.histplot(df[column], bins=30, kde=True)
    plt.title(f'Distribution of {column}')
    plt.show()

# Heatmap for correlation between variables
plt.figure(figsize=(10, 8))
sns.heatmap(df.corr(), annot=True, cmap='coolwarm', center=0)
plt.title('Correlation Heatmap')
plt.show()
```

```
Descriptive Statistics:
          Channel      Region         Fresh          Milk      Grocery  \
count  440.000000  440.000000    440.000000    440.000000    440.00000
```

```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
df_scaled = pd.DataFrame(scaler.fit_transform(df), columns=df.columns)
```

```
75%      2.000000    3.000000  16933.750000   7190.250000  10655.75000
```

```
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()
```