



Experiment No. 5
Apply appropriate Unsupervised Learning Technique on the Wholesale Customers Dataset
Date of Performance:21-08-23
Date of Submission:25-09-23



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

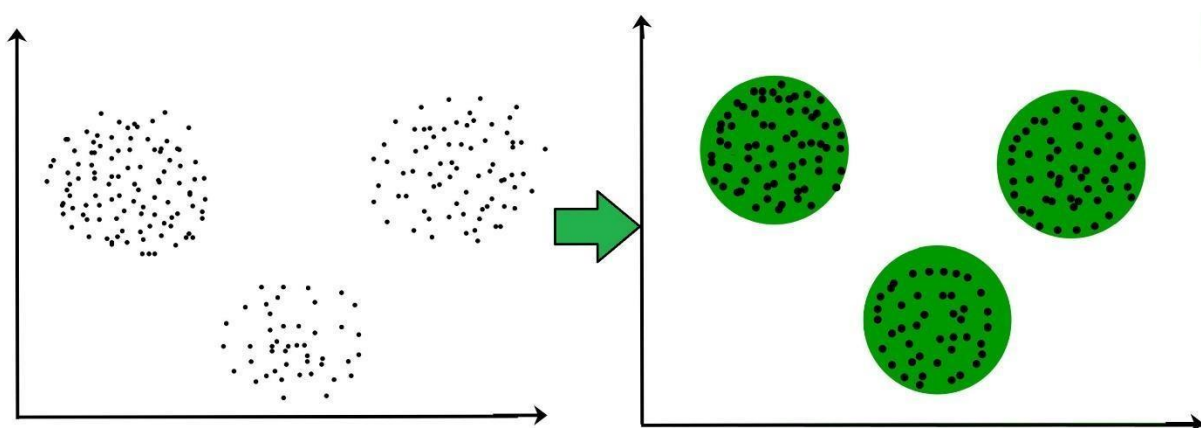
**Objective:** Able to perform various feature engineering tasks, apply Clustering Algorithm on the given dataset.

**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:**

Comment on the following based on the visualisation:

1. How may the clustered data be used?

Utilise clustered data to understand different client segments and create marketing strategies that are tailored to their needs.

Identify products that are regularly purchased in groups to get recommendations that are specifically tailored to you.

Inventory Optimisation: For effective stock management, optimise inventory based on cluster preferences.

Supply Chain Efficiency: Tailor supply chain procedures to each cluster's particular requirements.

Customer Retention: Create plans based on cluster traits to increase patronage.

Market Expansion: Look for potential for market expansion by clustering comparable client categories or new markets.

2. How various consumer segments, or customer groups, might be impacted differently by a particular distribution method?

Premium Delivery: High-value clients who value comfort and are prepared to pay more for expedited delivery.

Customers that are price conscious and favour standard or affordable delivery alternatives, including free options, are known as budget shoppers.

Bulk Buyers: Clientele who prefer to make larger purchases and may profit from special delivery services or discounts on bulk orders.

Customers that shop frequently and can take advantage of subscription- or loyalty-based delivery plans to promote 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('/content/Wholesale customers data.csv'):
```

```
    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())

```

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

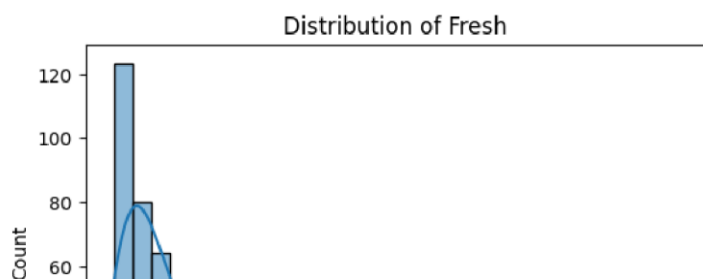
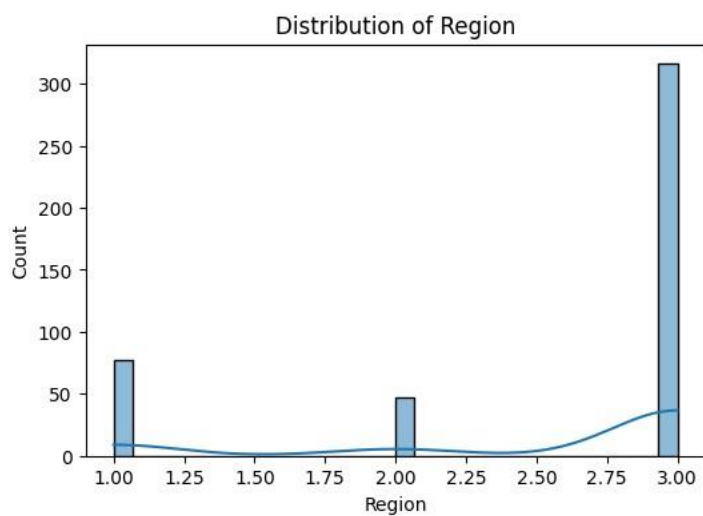
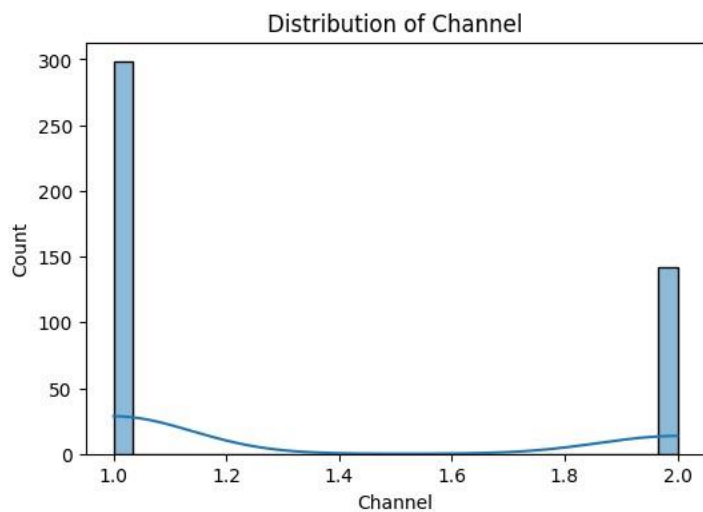
	Frozen	Detergents_Paper	Delicassen
count	440.000000	440.000000	440.000000
mean	3071.931818	2881.493182	1524.870455
std	4854.673333	4767.854448	2820.105937
min	25.000000	3.000000	3.000000
25%	742.250000	256.750000	408.250000
50%	1526.000000	816.500000	965.500000
75%	3554.250000	3922.000000	1820.250000
max	60869.000000	40827.000000	47943.000000

Number of duplicate rows: 0

```

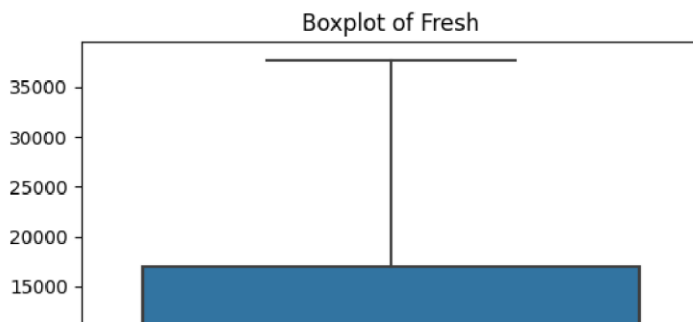
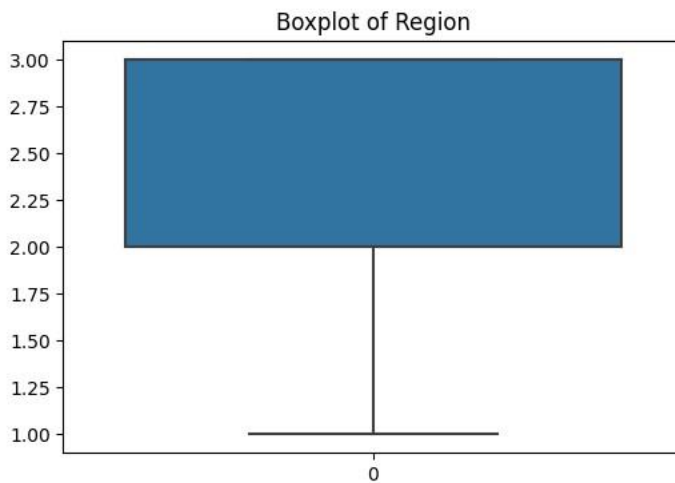
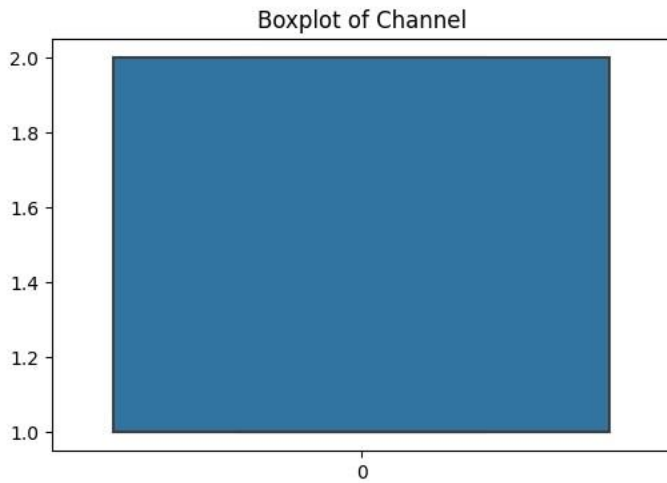
# Distribution plots for each feature for column in
df.columns:      plt.figure(figsize=(6, 4))
sns.histplot(df[column], bins=30, kde=True)
plt.title(f'Distribution of {column}')      plt.show() #
Heatmap for correlation between variables
plt.figure(figsize=(10, 8)) sns.heatmap(df.corr(),
annot=True, cmap='coolwarm', center=0)
plt.title('Correlation Heatmap') plt.show()

```



```
# 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)}')
```





```
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()
```

