



Experiment No. 5
Apply appropriate Unsupervised Learning Technique on the Wholesale Customers Dataset
Date of Performance:
Date of Submission:



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

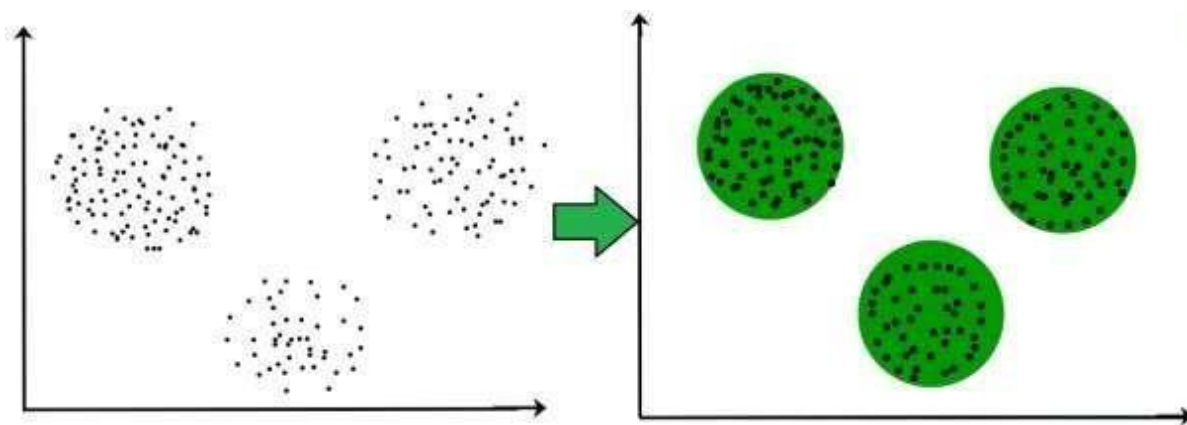
Objective: Able to perform various feature engineering tasks, apply Clustering Algorithm on the given dataset.

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:

```
import numpy as np

import pandas as pd

from IPython.display import display

from sklearn.model_selection import train_test_split

from sklearn.tree import DecisionTreeRegressor

from sklearn.decomposition import PCA

from sklearn.cluster import KMeans

from sklearn.metrics import silhouette_score

import seaborn as sns

import matplotlib.pyplot as plt

%matplotlib inline

try:

    data = pd.read_csv("../input/customers.csv")

    data.drop(['Region', 'Channel'], axis = 1, inplace = True)

    print("Wholesale customers dataset has {} samples with {} features each.".format(*data.shape))

except:

    print("Dataset could not be loaded. Is the dataset missing?")

import warnings

warnings.filterwarnings("ignore", category = UserWarning, module = "matplotlib")

from IPython import get_ipython
```



```
get_ipython().run_line_magic('matplotlib', 'inline')

import matplotlib.pyplot as plt

import matplotlib.cm as cm

import pandas as pd

import numpy as np

def pca_results(good_data, pca):

    dimensions = dimensions = ['Dimension {}'.format(i) for i in
range(1,len(pca.components_)+1)]

    components = pd.DataFrame(np.round(pca.components_, 4), columns =
list(good_data.keys()))

    components.index = dimensions

    ratios = pca.explained_variance_ratio_.reshape(len(pca.components_), 1)

    variance_ratios = pd.DataFrame(np.round(ratios, 4), columns = ['Explained Variance'])

    variance_ratios.index = dimensions

    fig, ax = plt.subplots(figsize = (14,8))

    components.plot(ax = ax, kind = 'bar');

    ax.set_ylabel("Feature Weights")

    ax.set_xticklabels(dimensions, rotation=0)

    for i, ev in enumerate(pca.explained_variance_ratio_):

        ax.text(i-0.40, ax.get_ylim()[1] + 0.05, "Explained Variance\n %.4f"%(ev))

    return pd.concat([variance_ratios, components], axis = 1)
```



```
def cluster_results(reduced_data, preds, centers, pca_samples):

    predictions = pd.DataFrame(preds, columns = ['Cluster'])

    plot_data = pd.concat([predictions, reduced_data], axis = 1)

    fig, ax = plt.subplots(figsize = (14,8))

    cmap = cm.get_cmap('gist_rainbow')

    for i, cluster in plot_data.groupby('Cluster'):

        cluster.plot(ax = ax, kind = 'scatter', x = 'Dimension 1', y = 'Dimension 2', \

                     color = cmap((i)*1.0/(len(centers)-1)), label = 'Cluster %i'%(i), s=30);

    for i, c in enumerate(centers):

        ax.scatter(x = c[0], y = c[1], color = 'white', edgecolors = 'black', \

                  alpha = 1, linewidth = 2, marker = 'o', s=200);

        ax.scatter(x = c[0], y = c[1], marker='$_d$'%(i), alpha = 1, s=100);

    ax.scatter(x = pca_samples[:,0], y = pca_samples[:,1], \

              s = 150, linewidth = 4, color = 'black', marker = 'x');

    ax.set_title("Cluster Learning on PCA-Reduced Data - Centroids Marked by\n\nTransformed Sample Data Marked by Black Cross");

def biplot(good_data, reduced_data, pca):

    fig, ax = plt.subplots(figsize = (14,8))

    ax.scatter(x=reduced_data.loc[:, 'Dimension 1'], y=reduced_data.loc[:, 'Dimension 2'],

              facecolors='b', edgecolors='b', s=70, alpha=0.5)

    feature_vectors = pca.components_.T

    arrow_size, text_pos = 7.0, 8.0,
```



```
for i, v in enumerate(feature_vectors):
```

```
    ax.arrow(0, 0, arrow_size*v[0], arrow_size*v[1],
```

```
            head_width=0.2, head_length=0.2, linewidth=2, color='red')
```

```
    ax.text(v[0]*text_pos, v[1]*text_pos, good_data.columns[i], color='black',
```

```
           ha='center', va='center', fontsize=18)
```

```
ax.set_xlabel("Dimension 1", fontsize=14)
```

```
ax.set_ylabel("Dimension 2", fontsize=14)
```

```
ax.set_title("PC plane with original feature projections.", fontsize=16);
```

```
return ax
```

```
def channel_results(reduced_data, outliers, pca_samples):
```

```
    try:
```

```
        full_data = pd.read_csv("../input/customers.csv")
```

```
    except:
```

```
        print("Dataset could not be loaded. Is the file missing?")
```

```
        return False
```

```
    channel = pd.DataFrame(full_data['Channel'], columns = ['Channel'])
```

```
    channel = channel.drop(channel.index[outliers]).reset_index(drop = True)
```

```
    labeled = pd.concat([reduced_data, channel], axis = 1)
```

```
    fig, ax = plt.subplots(figsize = (14,8))
```

```
    cmap = cm.get_cmap('gist_rainbow')
```

```
    labels = ['Hotel/Restaurant/Cafe', 'Retailer']
```

```
    grouped = labeled.groupby('Channel')
```



for i, channel in grouped:

```
channel.plot(ax = ax, kind = 'scatter', x = 'Dimension 1', y = 'Dimension 2', \
```

```
color = cmap((i-1)*1.0/2), label = labels[i-1], s=30);
```

for i, sample in enumerate(pca_samples):

```
ax.scatter(x = sample[0], y = sample[1], \
```

```
s = 200, linewidth = 3, color = 'black', marker = 'o', facecolors = 'none');
```

```
ax.scatter(x = sample[0]+0.25, y = sample[1]+0.3, marker='%d$(i)', alpha = 1, s=125);
```

```
ax.set_title("PCA-Reduced Data Labeled by 'Channel'\nTransformed Sample Data Circled");
```

```
def sampl_pop_plotting(sample):
```

```
fig, ax = plt.subplots(figsize=(10,5))
```

```
index = np.arange(sample.count())
```

```
bar_width = 0.3
```

```
opacity_pop = 1
```

```
opacity_sample = 0.3
```

```
rect1 = ax.bar(index, data.mean(), bar_width,
```

```
alpha=opacity_pop, color='g',
```

```
label='Population Mean')
```

```
rect2 = ax.bar(index + bar_width, sample, bar_width,
```

```
alpha=opacity_sample, color='k',
```

```
label='Sample')
```

```
ax.set_xlabel('Categories')
```

```
ax.set_ylabel('Total Purchase Cost')
```




```
ax.set_title('Sample vs Population Mean')
```

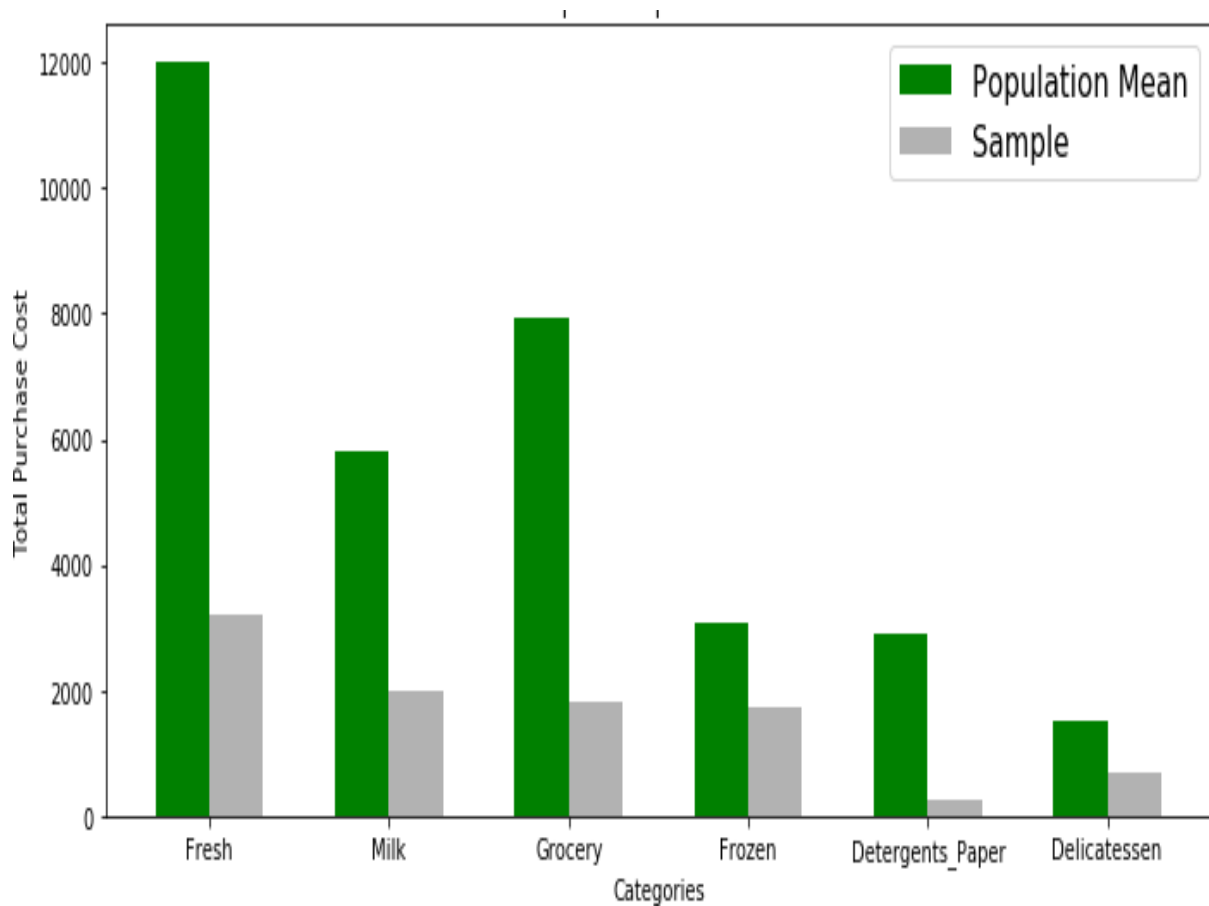
```
ax.set_xticks(index + bar_width / 2)
```

```
ax.set_xticklabels(samples.columns)
```

```
ax.legend(loc=0, prop={'size': 15})
```

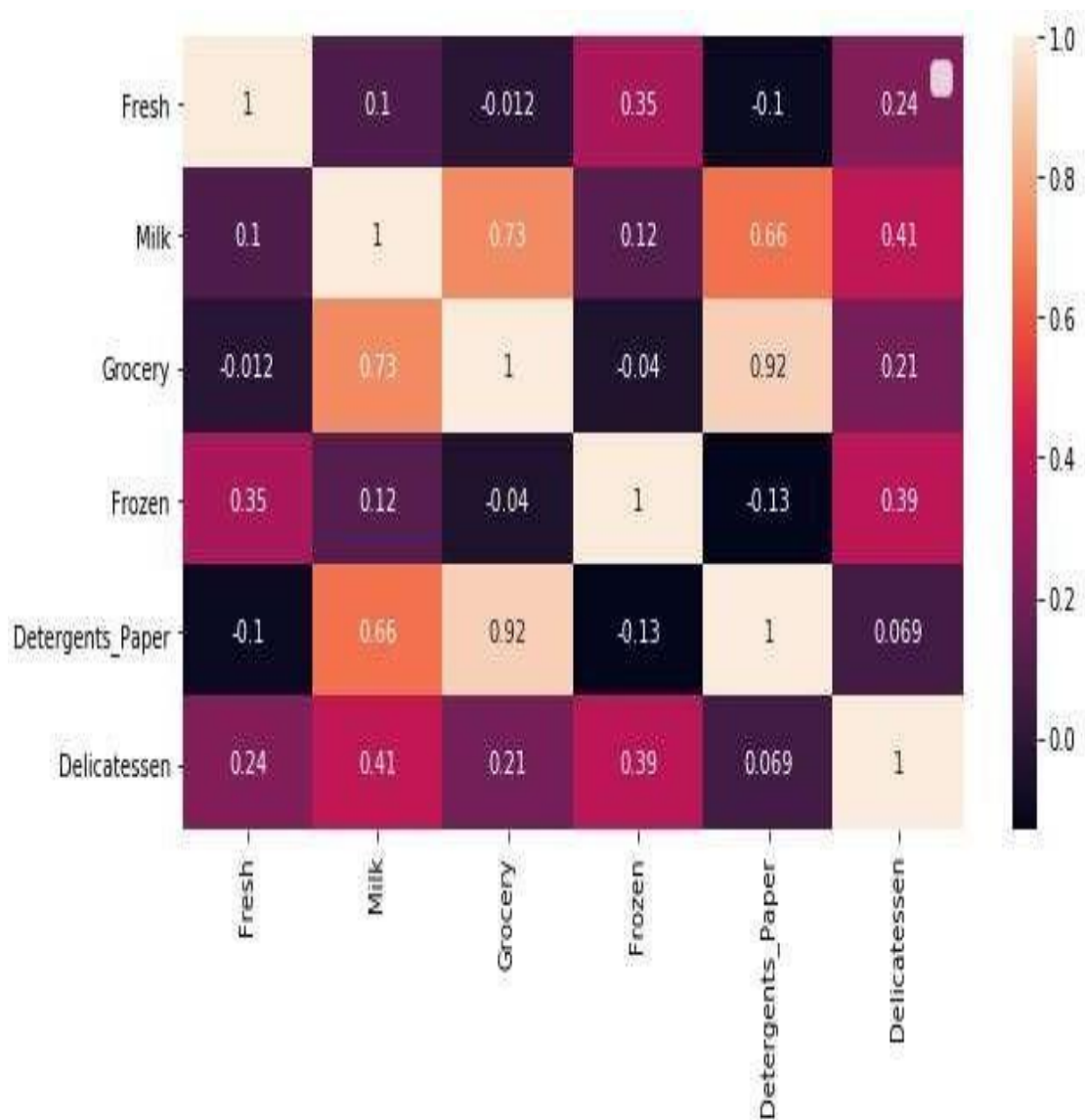
```
fig.tight_layout()
```

```
plt.show()
```



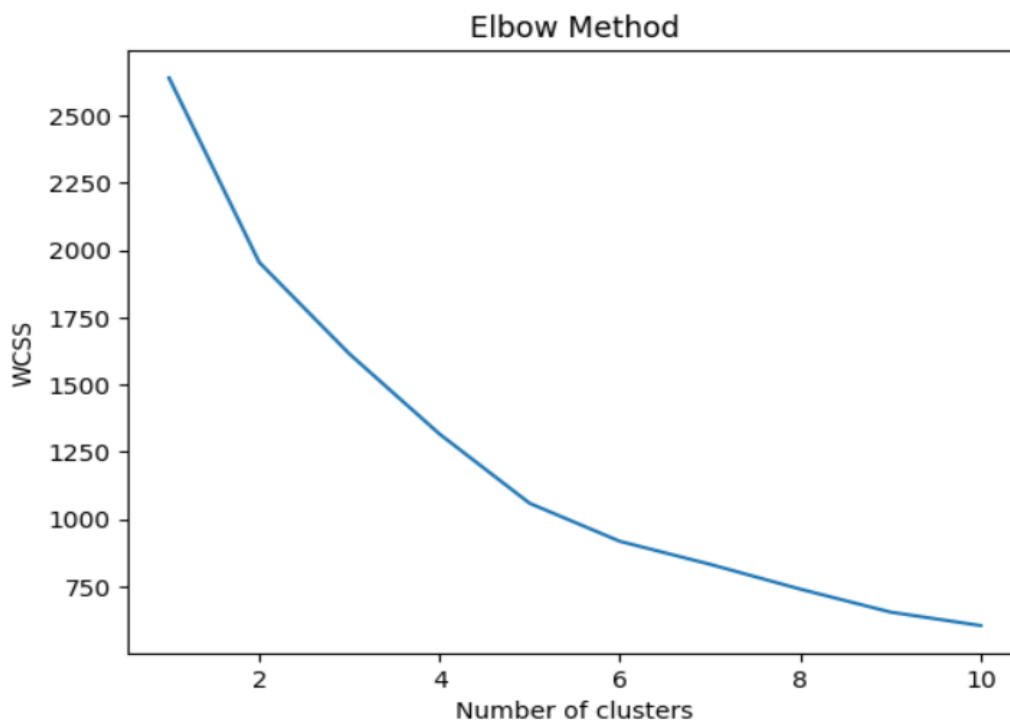


```
percentiles_data = 100*data.rank(pct=True)  
  
percentiles_samples = percentiles_data.iloc[indices]  
  
plt.subplots(figsize=(10,5))  
  
_ = sns.heatmap(percentiles_samples, annot=True)
```





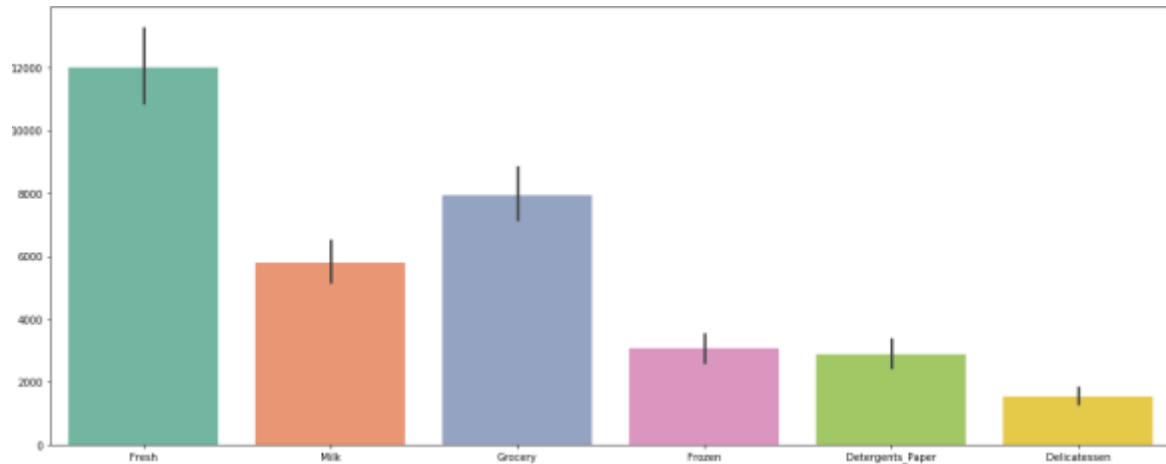
```
plt.plot(range(1, 11), wcss)
plt.title('Elbow Method')
plt.xlabel('Number of clusters')
plt.ylabel('WCSS') # Within-Cluster Sum of Squares
plt.show()
```





```
plt.figure(figsize = (20,8))
```

```
_ = sns.barplot(data=data, palette="Set2")
```



```
pca = PCA(n_components = 2, random_state=0)
```

```
pca.fit(good_data)
```

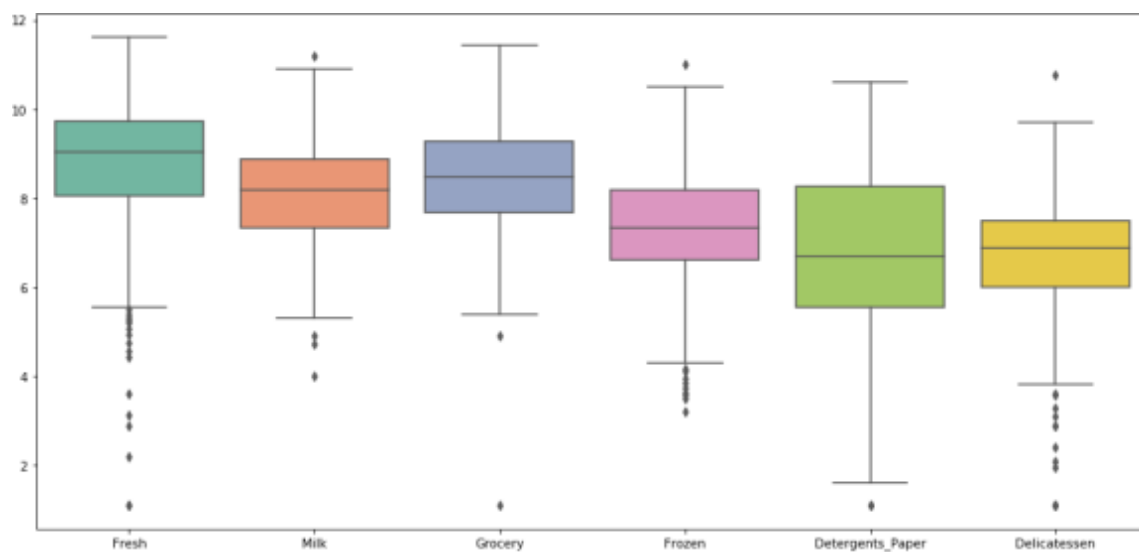
```
reduced_data = pca.transform(good_data)
```

```
pca_samples = pca.transform(log_samples)
```

```
reduced_data = pd.DataFrame(reduced_data, columns = ['Dimension 1', 'Dimension 2'])
```

```
plt.figure(figsize = (16,8))
```

```
_ = sns.boxplot(data=log_data, palette="Set2")
```





```
plt.scatter(data['Milk'], data['Grocery'],  
c=data['Cluster'])  
plt.xlabel('Milk')  
plt.ylabel('Grocery')  
plt.title('Clustering of Wholesale  
Customers')  
plt.show()
```



Conclusion:

1. How can you can make use of the clustered data?

Customer Segmentation: Cluster customers with similar behavior for targeted marketing and personalized recommendations.

Anomaly Detection: Identify outliers for fraud detection, quality control, and error identification.

Recommendation Systems: Use clusters to recommend products, content, or connections based on user preferences.

Content Categorization: Group documents with similar themes for content organization and topic modeling.

Dimensionality Reduction: Reduce feature dimensions for improved machine learning



model performance.

Resource Allocation: Optimize resource allocation and logistics by grouping similar resources or suppliers.

Image Segmentation: Segment images into regions with similar characteristics for computer vision tasks.

Healthcare: Group patients with similar profiles for disease diagnosis and personalized treatment.

Stock Market Analysis: Group stocks with similar metrics for portfolio diversification and trading strategies.

Social Network Analysis: Identify user communities based on shared interests for targeted advertising and community-building.

2. How the different groups of customers, the *customer segments*, may be affected differently by a specific delivery scheme?

- High-Volume Retailers: They prioritize fast and reliable deliveries to maintain sufficient stock levels. Delays may lead to out-of-stock situations and lost sales.
- Restaurants and Cafes: Freshness and flexibility in delivery schedules are crucial. They rely on timely deliveries of specific fresh ingredients.
- Small Retailers and Convenience Stores: They require cost-effective and flexible delivery options, often with smaller order quantities and shorter delivery windows.
- Online Retailers: Cost-effective and trackable deliveries are essential for servicing their online customers efficiently.
- Specialty Stores: They may prioritize specialized handling and quality of products upon delivery, potentially willing to pay more for such services.
- Local Markets and Farmers' Markets: They focus on local and sustainable sourcing, requiring a delivery scheme that supports these practices.