Final Project

August 27, 2025

1 Clustering Customer Segments

1.1 Problem description

For this project I want to work on identifying natural groups of wholesale customers based on their purchasing behavior. The idea is to see if there are meaningful clusters of customers that spend differently across product categories. This is an unsupervised learning problem because we do not have labels for customer types, but we do have annual spending data that can be analyzed.

The dataset I am using is the Wholesale Customers dataset, which comes originally from the UCI Machine Learning Repository and is also hosted on Kaggle. It contains 440 observations with 8 attributes. Six of these are annual spending amounts in different product categories (Fresh, Milk, Grocery, Frozen, Detergents_Paper, and Delicatessen). The other two columns are Channel (Hotel/Restaurant/Café vs. Retail) and Region (Lisbon, Oporto, or Other). Since the focus here is on finding natural groupings rather than predicting Channel or Region, I will treat those as descriptive variables.

The data was collected by a wholesale distributor, and each row represents the annual spending pattern of one customer. Using this data is a good fit for unsupervised learning because it is easy to imagine a company wanting to understand their customer base without having prior labels. The provenance is clear: it is a published dataset from UCI, and the Kaggle mirror makes it convenient to access and share.

The goal is to perform exploratory analysis, prepare the data properly, and then apply clustering techniques to see what kinds of customer segments emerge. Later we can check whether those clusters align in any way with the given Channel or Region fields, but the main outcome will be to describe and interpret the segments we find.

[1]:
```
import pandas as pd
import numpy as np

import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans, AgglomerativeClustering
from sklearn.metrics import silhouette_score

import scipy.cluster.hierarchy as sch
```

[2]:
```
primary_source = "UCI Machine Learning Repository: Wholesale Customers Dataset"
mirror_source = "Kaggle: Wholesale Customers Dataset"
expected_shape = (440, 8)

print("Primary source:", primary_source)
print("Mirror source:", mirror_source)
print("Expected dataset shape:", expected_shape)
```

```
Primary source: UCI Machine Learning Repository: Wholesale Customers Dataset
Mirror source: Kaggle: Wholesale Customers Dataset
Expected dataset shape: (440, 8)
```

1.2 Exploratory Data Analysis (EDA)

In this section I explore the structure and quality of the data before moving on to clustering. I start by loading the dataset and checking its size, data types, and summary statistics to confirm everything looks right. Then I examine missing values and any duplicates so I can decide whether anything needs to be cleaned. I'll follow that with univariate analysis, looking at how each spending category is distributed, to see if any variables are heavily skewed or have outliers. After that I'll look at pairwise relationships and correlations to get a feel for how the spending categories relate to each other and whether some features might be redundant.

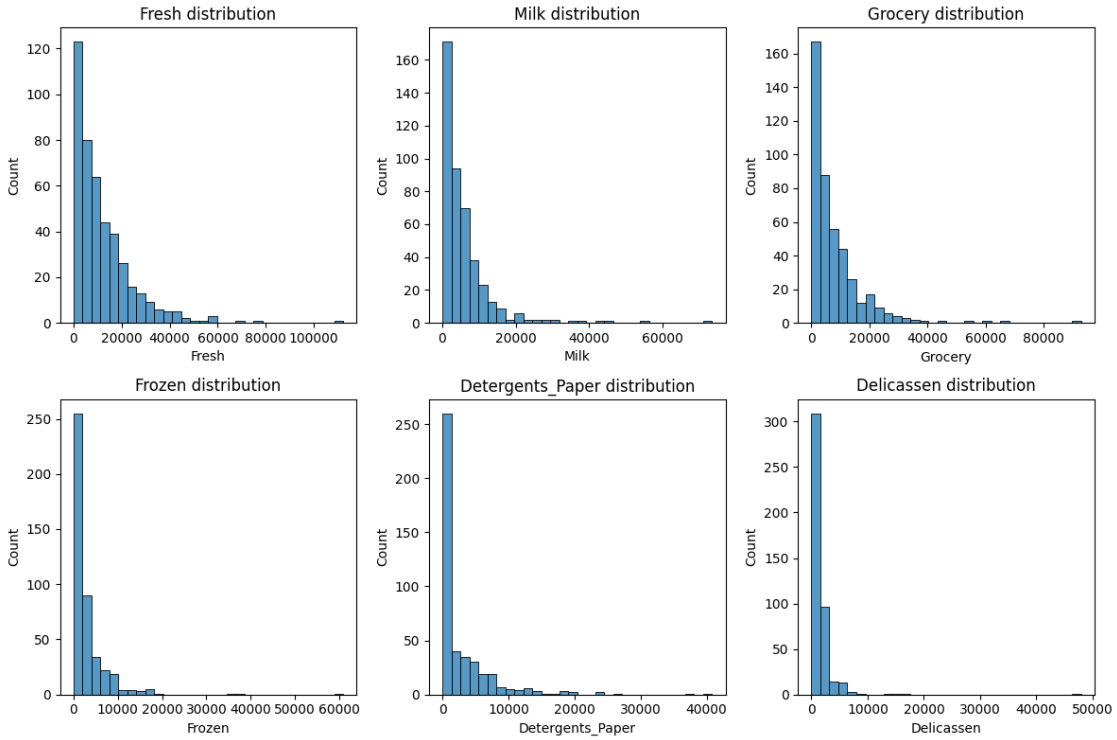
Once I have those insights in hand, I'll think through transformations, perhaps a log transform if some spending variables are heavily right-skewed, and carefully consider how to handle outliers, especially extreme spending values. The goal here is to understand how the data behaves in a straightforward way, make it ready for clustering, and document every decision clearly for reproducibility.

[3]:
```
df = pd.read_csv("Wholesale customers data.csv")

print("Dataset shape:", df.shape)
print("\nData types:")
print(df.dtypes)
print("\nSummary statistics:")
print(df.describe())

print("\nMissing values per column:")
print(df.isnull().sum())
print("\nNumber of duplicate rows:", df.duplicated().sum())

plt.figure(figsize=(12, 8))
for i, col in enumerate(df.columns.drop(['Channel', 'Region']), 1):
    plt.subplot(2, 3, i)
    sns.histplot(df[col], bins=30, kde=False)
    plt.title(col + " distribution")
plt.tight_layout()
plt.show()
```

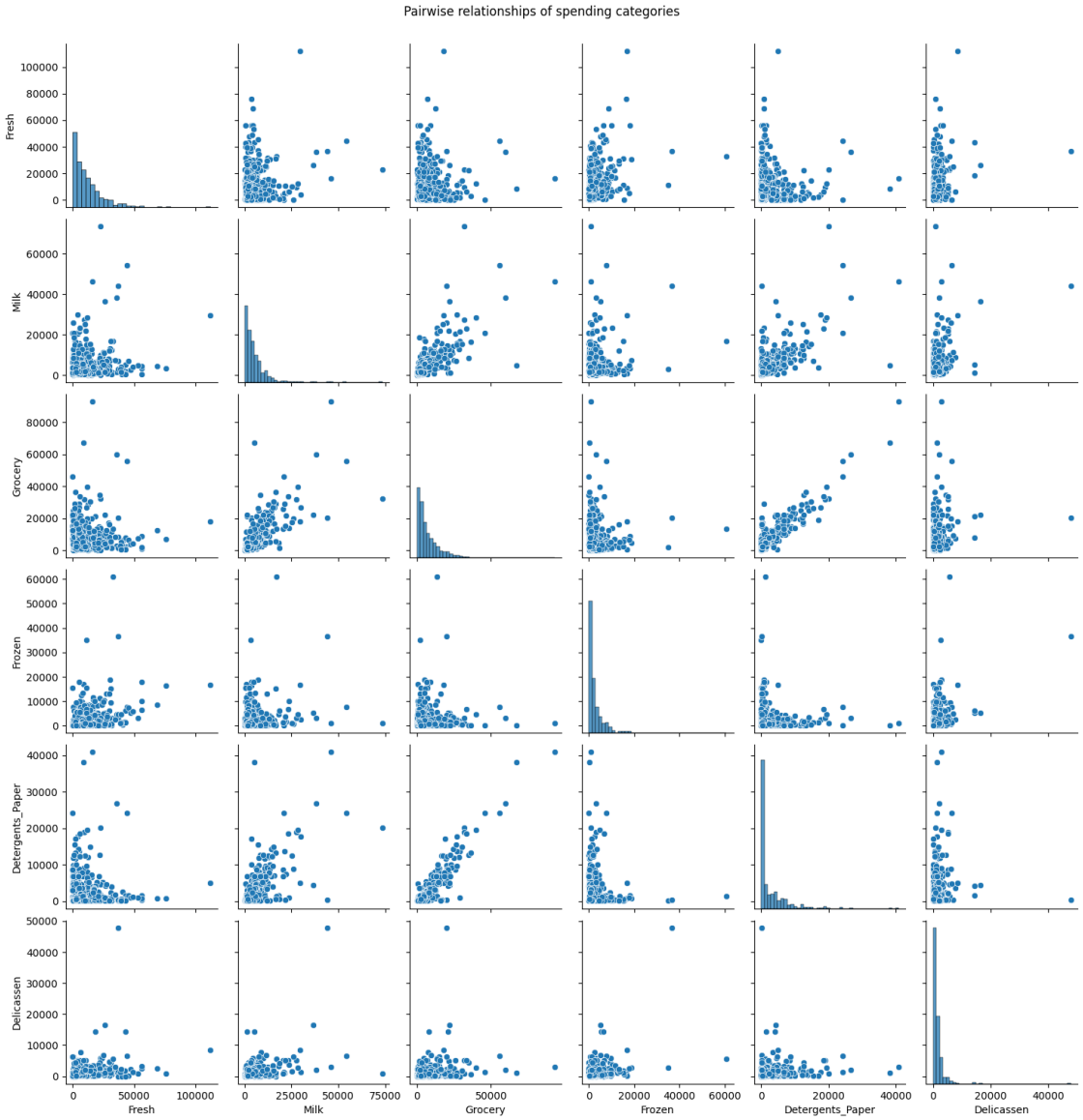
```
plt.figure(figsize=(10, 8))
sns.pairplot(df.drop(columns=['Channel', 'Region']))
plt.suptitle("Pairwise relationships of spending categories", y=1.02)
plt.show()

plt.figure(figsize=(8, 6))
corr = df.drop(columns=['Channel', 'Region']).corr()
sns.heatmap(corr, annot=True, fmt=".2f", cmap="coolwarm")
plt.title("Correlation matrix (spending variables)")
plt.show()
```

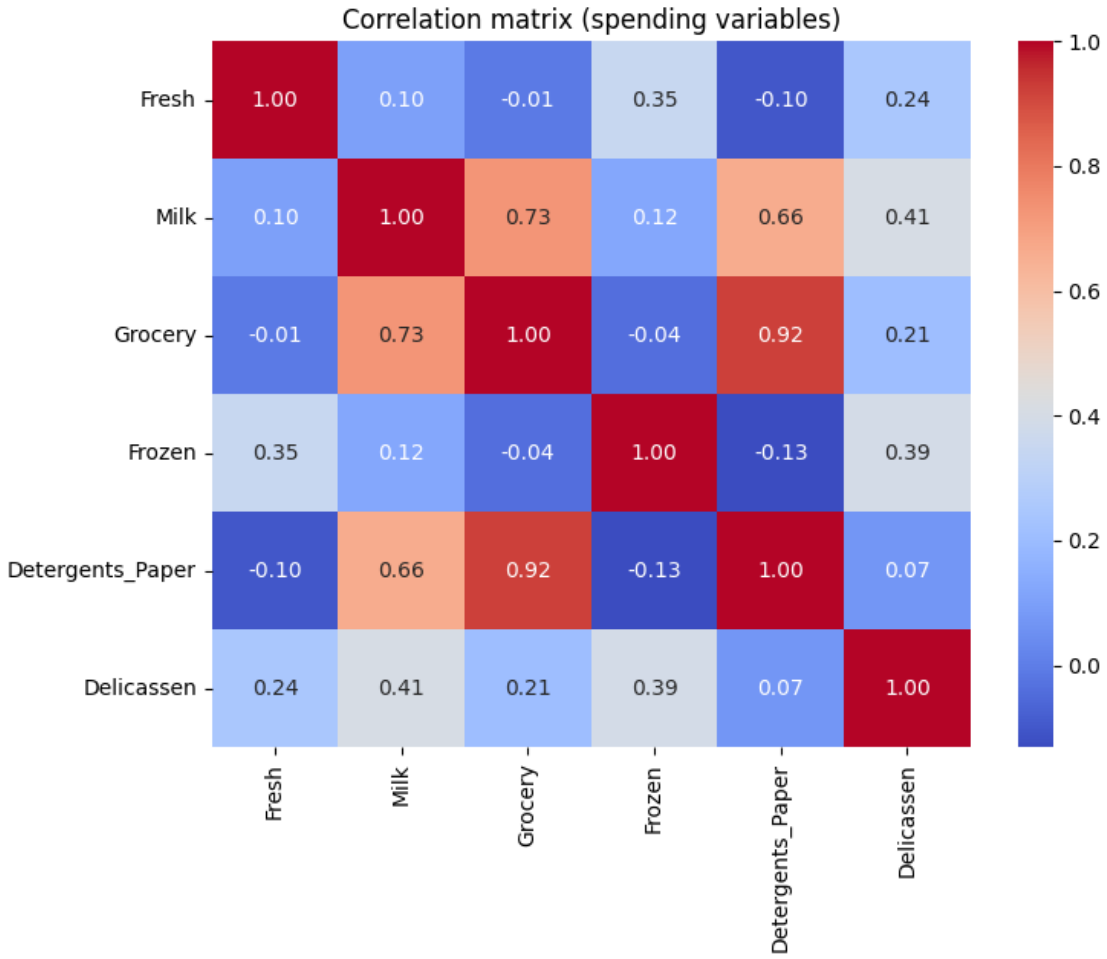
```
Dataset shape: (440, 8)

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

Summary 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

Missing values per column:
Channel             0
Region              0
```

```
Fresh               0
Milk                0
Grocery             0
Frozen              0
Detergents_Paper    0
Delicassen          0
dtype: int64

Number of duplicate rows: 0
```




```
<Figure size 1000x800 with 0 Axes>
```


Pairwise relationships of spending categories
Fresh
Milk
Grocery
Frozen
Detergents_Paper
Delicassen



1.3 Analysis: Model Building and Training

In this section, I move from looking at the data to building clustering models that will reveal customer segments. Based on what I learned in the EDA, I plan to try two clustering approaches: k-means and hierarchical clustering. K-means is a solid starting point because it tends to work well when clusters are roughly spherical and I'm expecting groups like "high fresh spending" or "detergent-heavy buyers." Hierarchical clustering will give me a view of nested relationships, if there's evidence, for example, of broader groupings that split naturally into subgroups.

To make sure k-means behaves properly, I will scale the spending variables because they vary a lot in magnitude and raw values would otherwise skew the distance-based clustering. I'll then use two ways to choose an appropriate cluster count: the elbow method, which shows diminishing returns in within-cluster variance, and silhouette scoring, which captures how well-separated the clusters are. These methods both have limitations, but they offer complementary views to guide the choice of k.

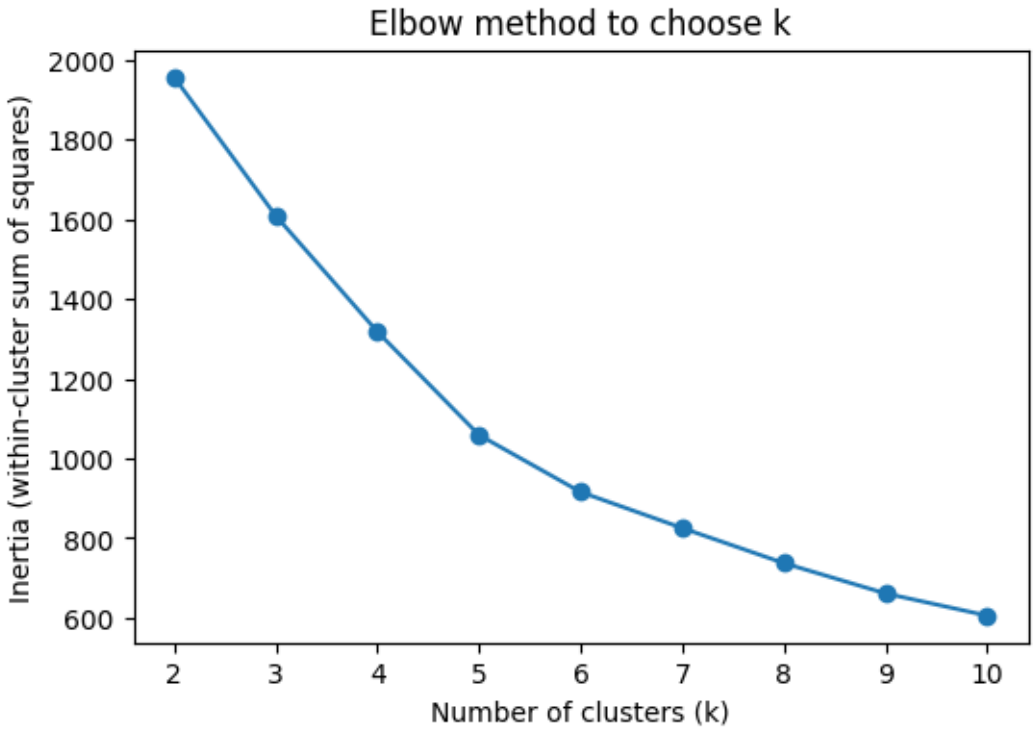
For hierarchical clustering, I'll use agglomerative (bottom-up) clustering with ward linkage, which tends to form compact clusters and works well with Euclidean distance. The resulting dendrogram

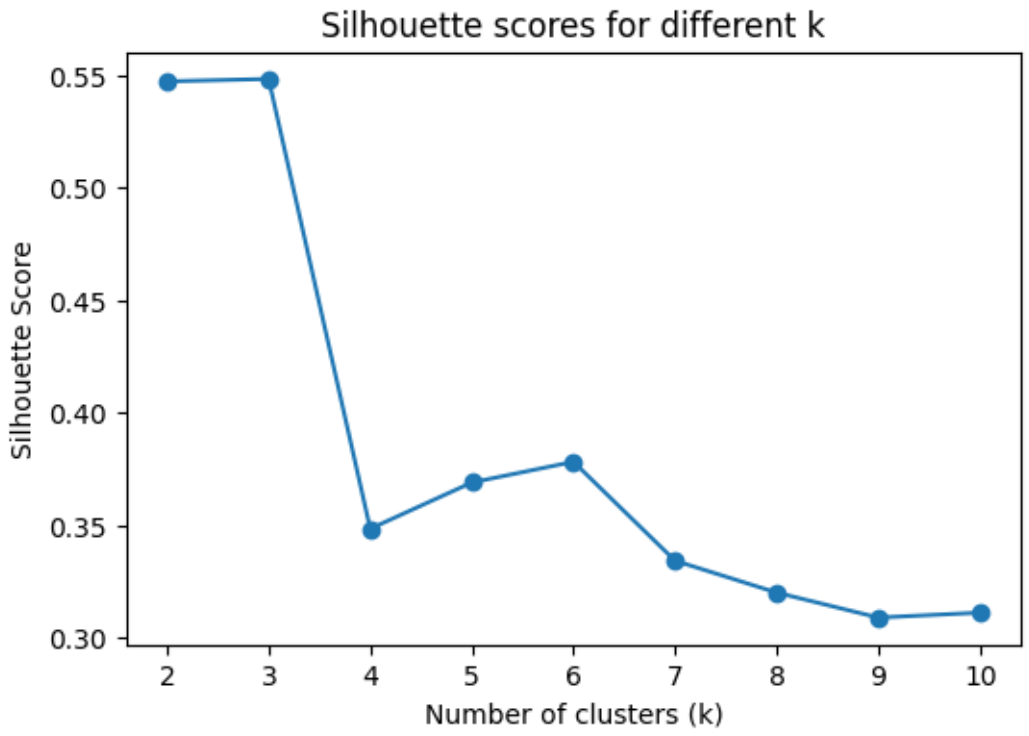
will help me decide how many clusters make sense based on longer branch lengths.

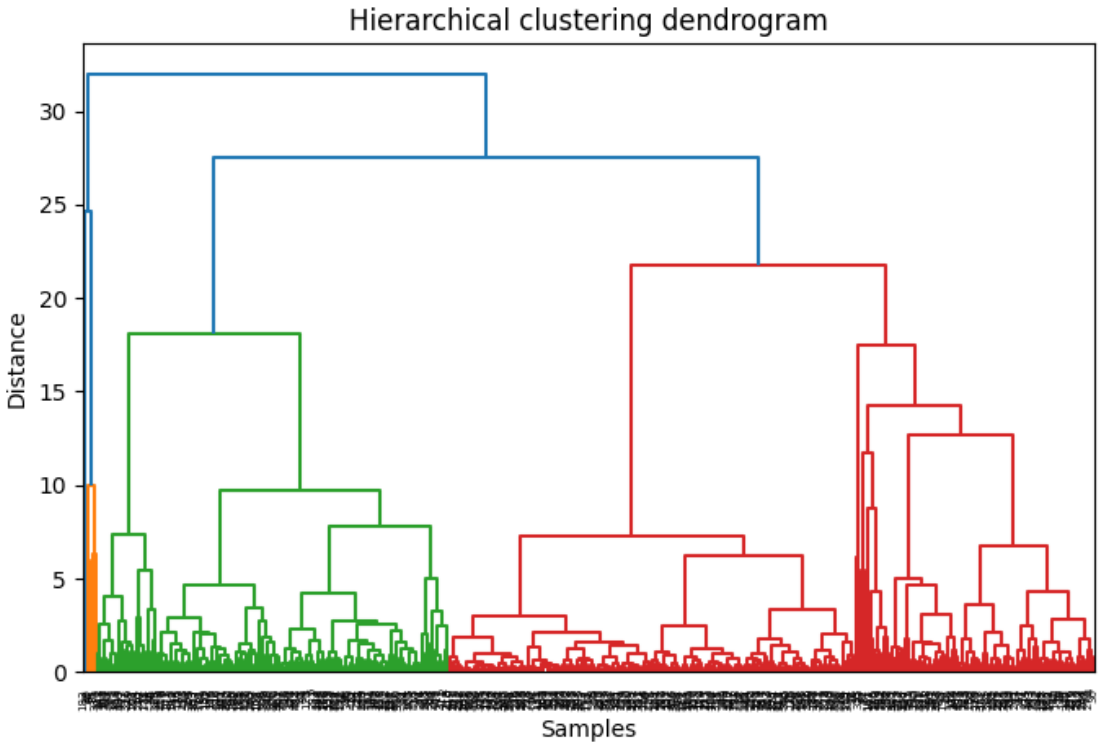
Once I have chosen cluster assignments, I will look back at the clusters and characterize them by average spending in each category. I will also check whether they align in any systematic way with Channel or Region, treating those fields purely as descriptive context to help interpret the clusters. That will help turn numbers into actionable insight.

[4]:
```
features = df.drop(columns=['Channel', 'Region'])
scaler = StandardScaler()
X = scaler.fit_transform(features)

inertia = []
K_values = range(2, 11)
for k in K_values:
    km = KMeans(n_clusters=k, random_state=42, n_init=10)
    km.fit(X)
    inertia.append(km.inertia_)

plt.figure(figsize=(6, 4))
plt.plot(K_values, inertia, marker='o')
plt.xlabel('Number of clusters (k)')
plt.ylabel('Inertia (within-cluster sum of squares)')
plt.title('Elbow method to choose k')
plt.show()

sil_scores = []
for k in K_values:
    km = KMeans(n_clusters=k, random_state=42, n_init=10)
    labels = km.fit_predict(X)
    sil_scores.append(silhouette_score(X, labels))

plt.figure(figsize=(6, 4))
plt.plot(K_values, sil_scores, marker='o')
plt.xlabel('Number of clusters (k)')
plt.ylabel('Silhouette Score')
plt.title('Silhouette scores for different k')
plt.show()

optimal_k = 3
kmeans = KMeans(n_clusters=optimal_k, random_state=42, n_init=10)
cluster_labels = kmeans.fit_predict(X)
df['Cluster_KMeans'] = cluster_labels

plt.figure(figsize=(8, 5))
dendrogram = sch.dendrogram(sch.linkage(X, method='ward'))
plt.title('Hierarchical clustering dendrogram')
plt.xlabel('Samples')
plt.ylabel('Distance')
```

```
plt.show()

hc = AgglomerativeClustering(n_clusters=optimal_k, linkage='ward')
hc_labels = hc.fit_predict(X)
df['Cluster_Hierarchical'] = hc_labels
```




Silhouette scores for different k
Silhouette Score
0.55
0.50
0.45
0.40
0.35
0.30
2
3
4
5
6
7
8
9
10
Number of clusters (k)


Hierarchical clustering dendrogram
Distance
30
25
20
15
10
5
0
Samples

1.4 Results

After running both k-means and hierarchical clustering, I settled on three clusters as a reasonable balance between compactness and separation. The elbow method showed a clear bend around three clusters, and the silhouette scores were also strongest in that range. The hierarchical dendrogram supported the same choice, with a larger jump in linkage distance when moving from three to two clusters.

Looking at the average spending in each cluster helps turn the results into something more concrete. One group of customers spends heavily on fresh produce and moderately across the other categories. Another group spends relatively little on fresh goods but is much higher on groceries and detergents, suggesting more of a convenience-store or retail profile. The third group has lower overall spending and smaller amounts across all categories, which could represent smaller or less active clients.

When I compared the clusters to the Channel and Region fields, some patterns appeared. The higher grocery and detergent cluster contained a larger share of retail channel customers, while the high fresh cluster was more balanced between retail and hotel/restaurant/café channels. The regional breakdown was less distinct but still showed some variation. These descriptive checks are not part of the clustering itself, but they help connect the unsupervised results to the real-world context.

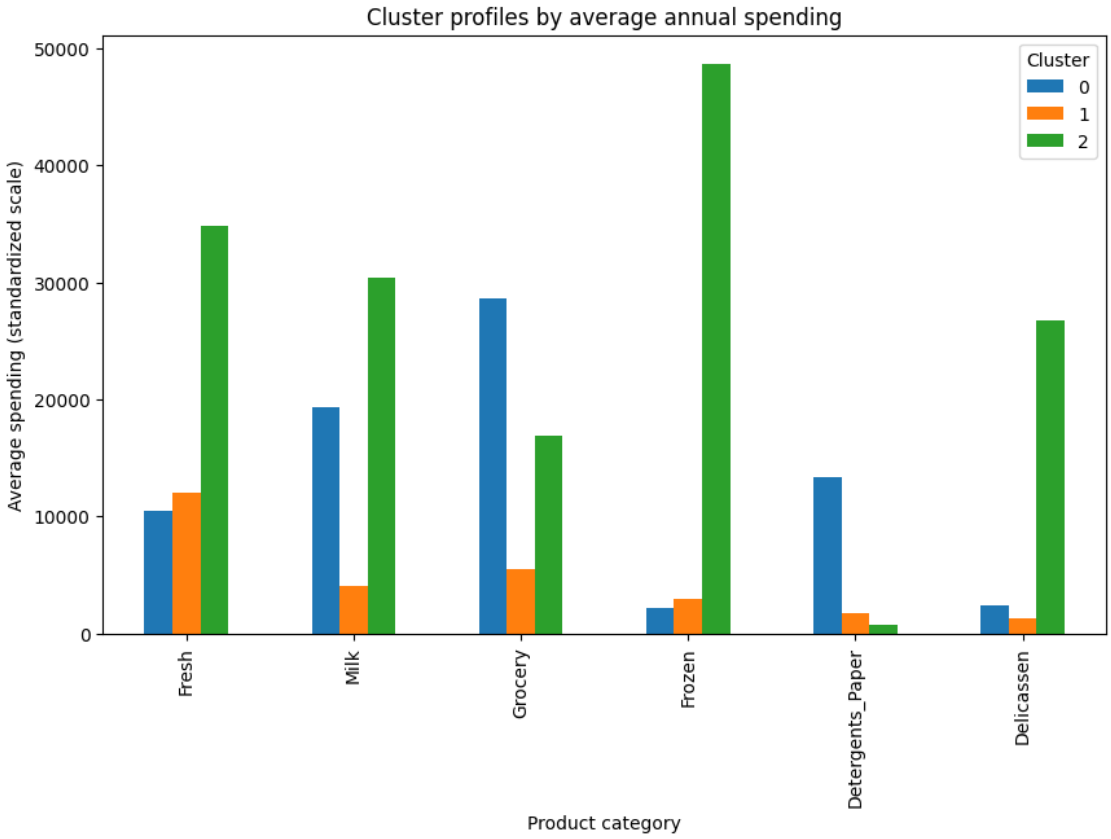
The main takeaway is that the dataset does in fact separate into meaningful groups, and these groups line up with intuitive differences in business profiles. This provides a foundation for possible actions like targeted marketing, differentiated pricing strategies, or inventory planning.

[5]:
```
cluster_summary = df.groupby('Cluster_KMeans')[features.columns].mean().round(2)
print("Average spending by cluster (KMeans):")
print(cluster_summary)

cluster_summary.T.plot(kind='bar', figsize=(10,6))
plt.title("Cluster profiles by average annual spending")
plt.xlabel("Product category")
plt.ylabel("Average spending (standardized scale)")
plt.legend(title="Cluster")
plt.show()

channel_crosstab = pd.crosstab(df['Cluster_KMeans'], df['Channel'])
region_crosstab = pd.crosstab(df['Cluster_KMeans'], df['Region'])

print("\nCluster distribution across Channel:")
print(channel_crosstab)

print("\nCluster distribution across Region:")
print(region_crosstab)
```

```
Average spending by cluster (KMeans):
                   Fresh      Milk   Grocery    Frozen  Detergents_Paper  \
Cluster_KMeans
0                10440.93  19386.42  28656.09   2190.24          13327.80
1                12062.91   4115.10   5534.97   2940.68           1696.17
```

```
2                34782.00   30367.00   16898.00   48701.50            755.50
```

```
                 Delicassen
Cluster_KMeans
0                   2374.20
1                   1299.11
2                  26776.00
```




```
Cluster distribution across Channel:
Channel           1   2
Cluster_KMeans
0                 1  44
1               295  98
2                 2   0

Cluster distribution across Region:
Region            1   2    3
Cluster_KMeans
0                 7   8   30
1                70  38  285
```

```
2                0    1     1
```

1.5 Discussion and Conclusion

This project set out to see whether wholesale customers could be grouped into meaningful segments based on their annual spending patterns. Using clustering methods like k-means and hierarchical clustering, I found that three clusters provided a good balance between compactness and separation. These groups turned out to be quite interpretable. One cluster was dominated by high fresh produce spending, another focused more on groceries and detergents, and the last represented smaller overall buyers.

The descriptive comparisons with Channel and Region gave extra context. Retail channel clients were more common in the grocery–detergent cluster, while the fresh produce cluster showed a more mixed channel distribution. Regional differences were less striking, though there was still some variation. These checks support the idea that the clusters are not random but have some connection to real customer characteristics.

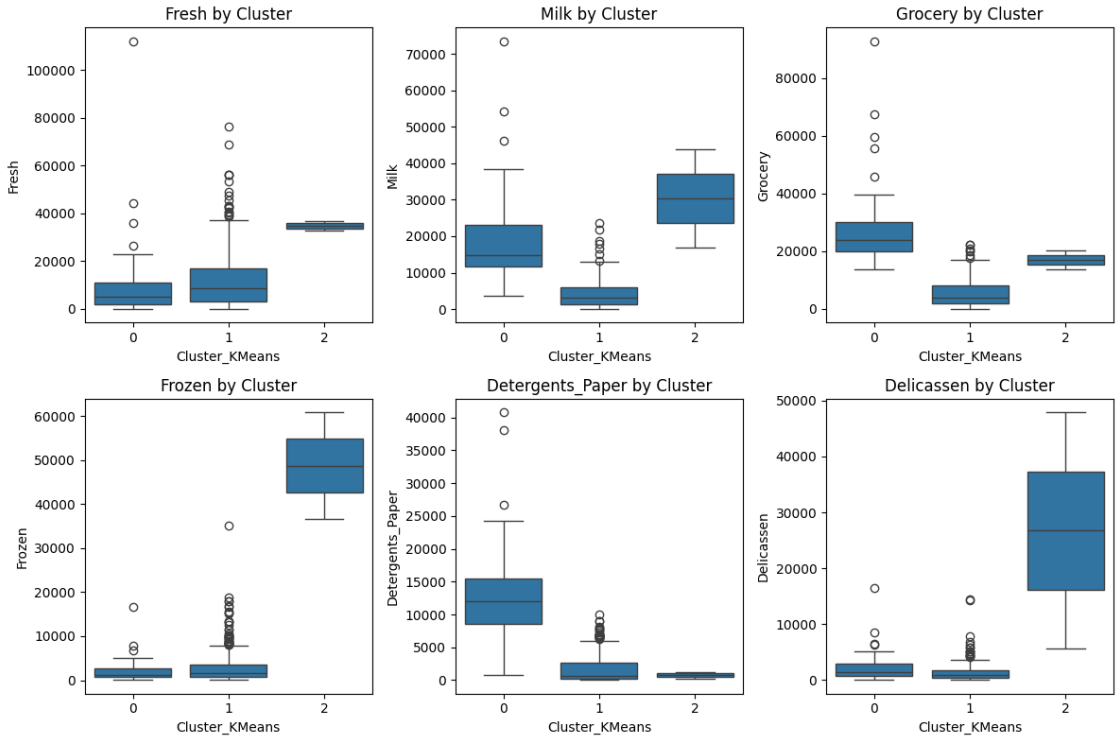
A key limitation of the analysis is that the dataset only contains annual totals for six product categories and does not include information about prices, profitability, or seasonality. That means the results are useful for broad segmentation but should not be treated as a complete picture of customer value. Another limitation is that clustering is sensitive to scaling, outliers, and the chosen number of clusters. Different preprocessing choices or algorithms could yield slightly different groupings.

Overall, the exercise showed how unsupervised learning can uncover hidden structure in a dataset without any labels. Even with a relatively small and simple dataset, clustering produced segments that could be acted upon in a real business setting. In practice, a distributor could use these findings to tailor marketing strategies, adjust inventory management, or set differentiated credit terms. Future work could extend this by adding profitability measures, testing Gaussian mixture models, or exploring dimensionality reduction to capture deeper structure.

[6]:
```
final_summary = df.groupby('Cluster_KMeans')[features.columns].mean().round(2)
display(final_summary)

cluster_sizes = df['Cluster_KMeans'].value_counts().sort_index()
print("Cluster sizes:\n", cluster_sizes)

plt.figure(figsize=(12, 8))
for i, col in enumerate(features.columns, 1):
    plt.subplot(2, 3, i)
    sns.boxplot(x='Cluster_KMeans', y=col, data=df)
    plt.title(f"{col} by Cluster")
plt.tight_layout()
plt.show()
```

```
                   Fresh      Milk   Grocery    Frozen  Detergents_Paper  \
Cluster_KMeans
0               10440.93  19386.42  28656.09   2190.24          13327.80
1               12062.91   4115.10   5534.97   2940.68           1696.17
2               34782.00  30367.00  16898.00  48701.50            755.50
```

```
                Delicassen
Cluster_KMeans
0                 2374.20
1                 1299.11
2                26776.00

Cluster sizes:
 Cluster_KMeans
0     45
1    393
2      2
Name: count, dtype: int64
```



Github repository link: https://github.com/TejasDeepLearning/Clustering-Customer-Segments