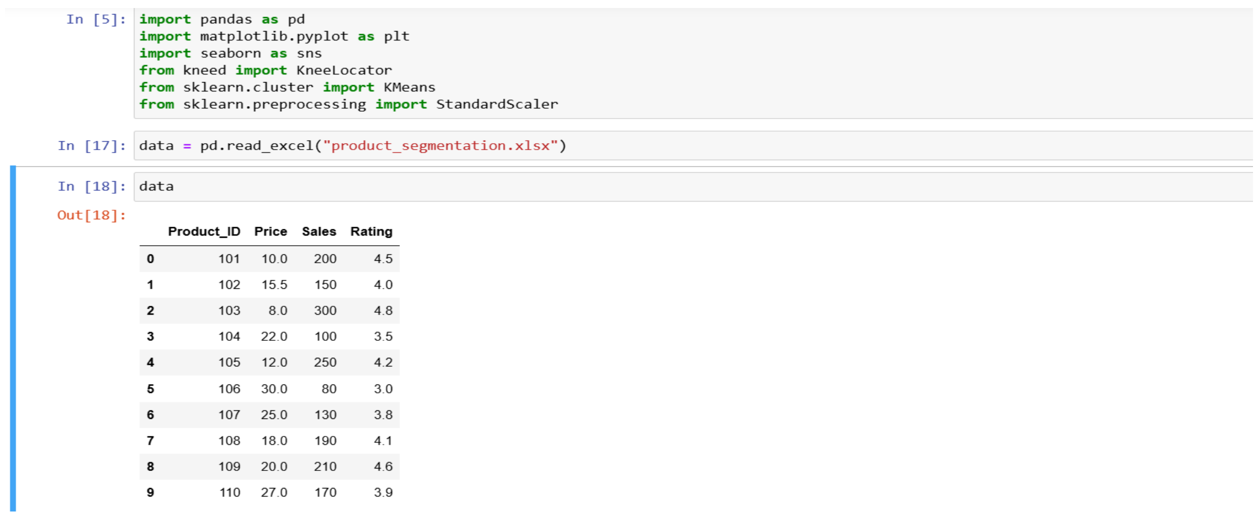
K-means Clustering:

K-means clustering is a method used to partition a set of data points into groups (clusters) based on similarity. The algorithm aims to minimize the variance within each cluster by iteratively assigning each data point to the cluster with the nearest mean (centroid), recalculating centroids based on the current assignments, and repeating until unification. This process results in clusters where data points within the same cluster are more like each other than to those in other clusters, making it useful for data analysis and pattern recognition tasks.

For implementing k-means clustering, we firstly import libraries and we perform clustering for product_segmentation dataset.



In [5]:
```python
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from kneed import KneeLocator
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
```

In [17]:
```python
data = pd.read_excel("product_segmentation.xlsx")
```

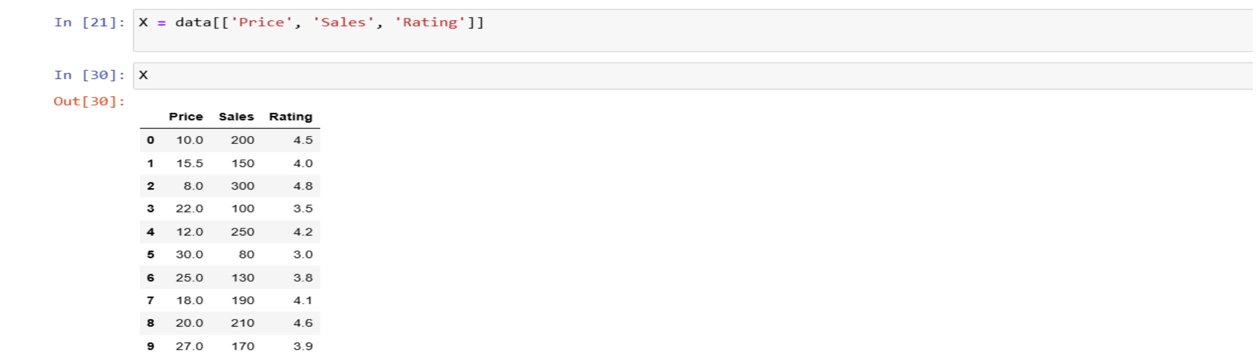
In [18]:
```python
data
```

Out[18]:

	Product_ID	Price	Sales	Rating
0	101	10.0	200	4.5
1	102	15.5	150	4.0
2	103	8.0	300	4.8
3	104	22.0	100	3.5
4	105	12.0	250	4.2
5	106	30.0	80	3.0
6	107	25.0	130	3.8
7	108	18.0	190	4.1
8	109	20.0	210	4.6
9	110	27.0	170	3.9

Selecting Specific Data Columns for Analysis

We select and create a new DataFrame X that includes only the columns 'Price', 'Sales', and 'Rating' from the original dataset data. This helps focus on these particular aspects of the data for analysis or other tasks like making predictions or finding patterns.



In [21]:
```python
X = data[['Price', 'Sales', 'Rating']]
```

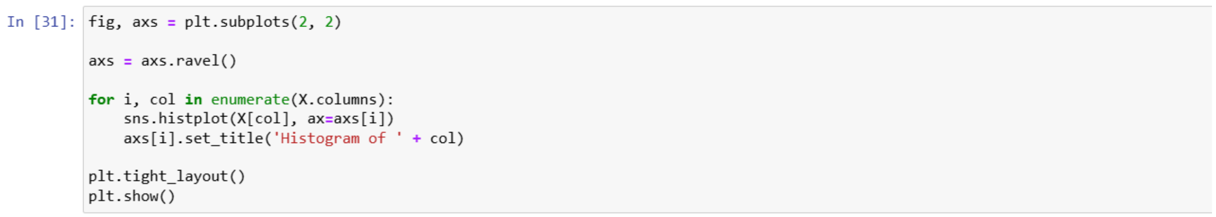
In [30]:
```python
X
```

Out[30]:

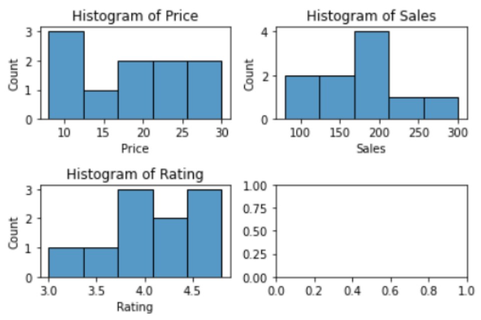
	Price	Sales	Rating
0	10.0	200	4.5
1	15.5	150	4.0
2	8.0	300	4.8
3	22.0	100	3.5
4	12.0	250	4.2
5	30.0	80	3.0
6	25.0	130	3.8
7	18.0	190	4.1
8	20.0	210	4.6
9	27.0	170	3.9

Visualizing Data Distributions with Histograms

This code segment creates a grid of four subplots using Matplotlib (fig, axs = plt.subplots(2, 2)), where each subplot will display a histogram (sns.histplot) of a specific column from the DataFrame X. The for loop iterates through each column in X, plotting its histogram on the corresponding subplot (axs[i]). Each histogram visualizes the distribution of values within the respective column ('Price', 'Sales', 'Rating'). The set_title method assigns a title to each subplot indicating the column name being visualized. Finally, plt.tight_layout() ensures proper spacing between subplots, and plt.show() displays the complete figure with all histograms. This visualization helps in understanding how values are distributed across different features in the dataset.



```
In [31]: fig, axs = plt.subplots(2, 2)

         axs = axs.ravel()

         for i, col in enumerate(X.columns):
             sns.histplot(X[col], ax=axs[i])
             axs[i].set_title('Histogram of ' + col)

         plt.tight_layout()
         plt.show()
```

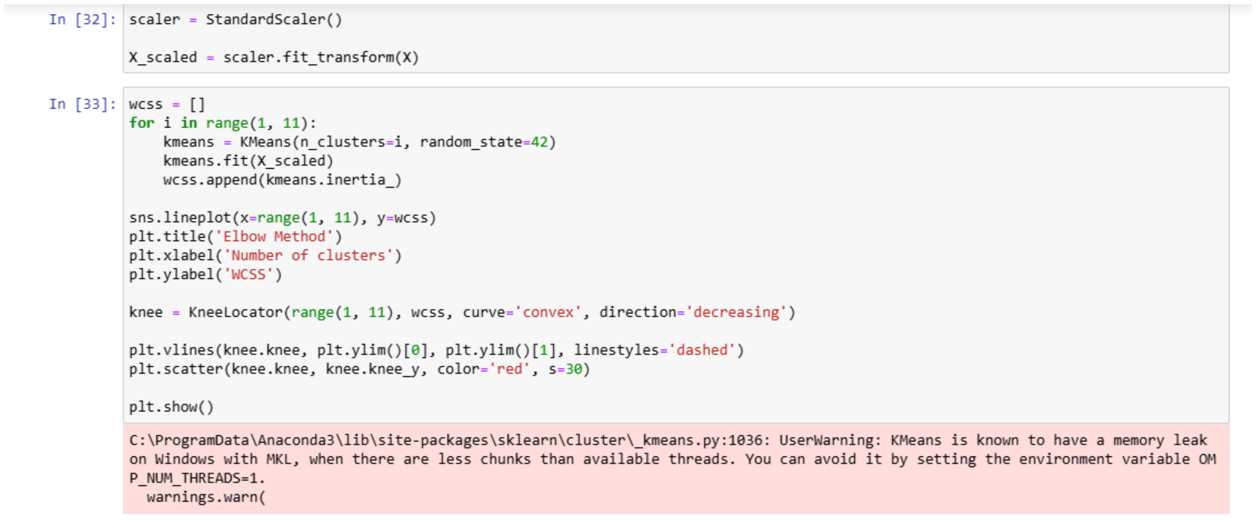


This code makes a grid of four plots. Each plot shows a histogram of different data columns like 'Price', 'Sales', and 'Rating'. It helps visualize how values are spread out in these columns, giving insights into their distributions and patterns in the dataset.

Determining Optimal Number of Clusters with the Elbow Method

This code segment performs an analysis to find the optimal number of clusters for K-means clustering using the Elbow Method. First, it scales the data (X_scaled) using StandardScaler() to normalize the values. Then, it iterates through a range of cluster numbers from 1 to 10 (for i in range(1, 11)), fitting K-means models (KMeans) to the scaled data and calculating the within-cluster sum of squares (WCSS) for each model (wcss.append(kmeans.inertia_)).

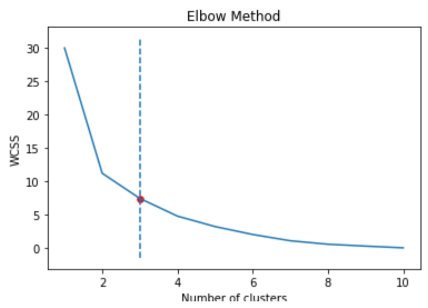
Then, it plots a line graph (sns.lineplot) where the x-axis represents the number of clusters and the y-axis represents the corresponding WCSS values. The Elbow Method aims to identify a point on this graph where the decrease in WCSS slows down, resembling an elbow shape. This point indicates the optimal number of clusters, suggesting the most significant gain in clustering performance without overfitting. The code also visualizes this point (knee.knee) with a dashed vertical line and a red dot, marking the recommended number of clusters for further analysis or segmentation of the data.



```
In [32]: scaler = StandardScaler()

         X_scaled = scaler.fit_transform(X)
```

```
In [33]: wcss = []
         for i in range(1, 11):
             kmeans = KMeans(n_clusters=i, random_state=42)
             kmeans.fit(X_scaled)
             wcss.append(kmeans.inertia_)

         sns.lineplot(x=range(1, 11), y=wcss)
         plt.title('Elbow Method')
         plt.xlabel('Number of clusters')
         plt.ylabel('WCSS')

         knee = KneeLocator(range(1, 11), wcss, curve='convex', direction='decreasing')

         plt.vlines(knee.knee, plt.ylim()[0], plt.ylim()[1], linestyles='dashed')
         plt.scatter(knee.knee, knee.knee_y, color='red', s=30)

         plt.show()
```

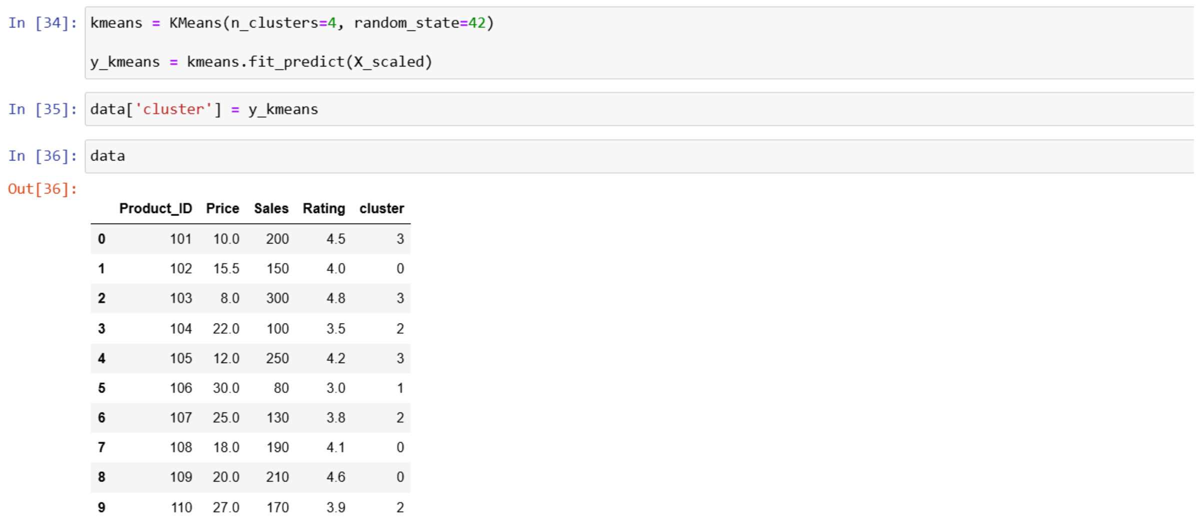
```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:1036: UserWarning: KMeans is known to have a memory leak
on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the environment variable OM
P_NUM_THREADS=1.
  warnings.warn(
```



This code uses the Elbow Method to decide how many clusters are best for K-means clustering. It scales the data, calculates WCSS for cluster numbers from 1 to 10, and plots these values. The elbow point on the graph indicates the optimal cluster count, balancing model accuracy and simplicity for effective data grouping and analysis.

Grouping Data into Clusters

In this code, we're using a method called K-means clustering to group our data into 4 clusters. Each cluster represents a group of data points that are similar to each other based on their values for attributes like 'Price', 'Sales', and 'Rating'. After applying this method, each data point is labeled with a cluster number, which is added as a new column ('cluster') in the original dataset (data). This helps us organize and analyze our data by grouping similar items together.



```
In [34]: kmeans = KMeans(n_clusters=4, random_state=42)

         y_kmeans = kmeans.fit_predict(X_scaled)

In [35]: data['cluster'] = y_kmeans

In [36]: data

Out[36]:
```

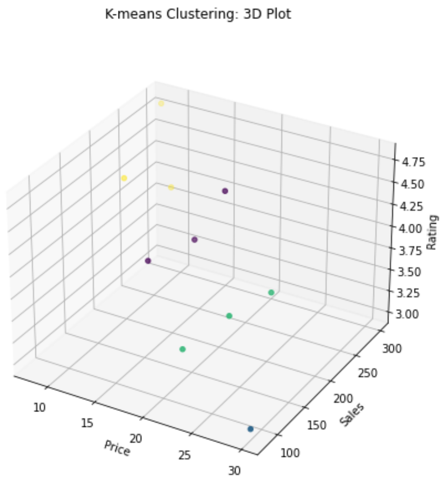
	Product_ID	Price	Sales	Rating	cluster
0	101	10.0	200	4.5	3
1	102	15.5	150	4.0	0
2	103	8.0	300	4.8	3
3	104	22.0	100	3.5	2
4	105	12.0	250	4.2	3
5	106	30.0	80	3.0	1
6	107	25.0	130	3.8	2
7	108	18.0	190	4.1	0
8	109	20.0	210	4.6	0
9	110	27.0	170	3.9	2

This code uses K-means clustering to group data points into four clusters based on attributes like 'Price', 'Sales', and 'Rating'. It helps organize and analyze the data by grouping similar items together, making it easier to spot patterns and make informed decisions about the dataset.

Visualizing Clusters in 3D Space

This code creates a 3D scatter plot to show how K-means clustering groups data based on 'Price', 'Sales', and 'Rating'. Each point in the plot represents an item from the dataset, positioned by its values for these three attributes. Points are colored by their cluster assignment to show distinct groups. This visualization helps to see how items with similar price ranges, sales figures, and ratings are grouped together, offering insights into patterns and relationships within the dataset.

In [37]:
```
fig = plt.figure(figsize=(10, 8))
ax = fig.add_subplot(111, projection='3d')

ax.scatter3D(data['Price'], data['Sales'], data['Rating'], c=data['cluster'], cmap='viridis')

ax.set_xlabel('Price')
ax.set_ylabel('Sales')
ax.set_zlabel('Rating')

plt.title('K-means Clustering: 3D Plot')

ax.set_box_aspect(aspect=None, zoom=0.9)
plt.show()
```



Conclusion

In K-means clustering, we used a method to group items based on their similarities in attributes like price, sales, and rating. First, we identified the optimal number of clusters by analyzing how data points' variance changes with different cluster counts. Once grouped, we visualized these clusters in 3D to see how items with similar characteristics are grouped together. This method helps to organize and understand data better, revealing patterns that can guide decisions like product pricing strategies or customer segmentation for targeted marketing. Therefore, K-means clustering is a useful tool for finding meaningful groups within data without needing predefined categories, making it valuable for various analytical tasks in business and research.