BUSINESS INTELLIGENCE AND DATA ANALYTICS

Open in Colab YouTube

Introduction to Clustering

Clustering is an *unsupervised machine learning* technique used to group similar data points together based on their features. Unlike classification, clustering doesn't rely on predefined labels—instead, it discovers inherent patterns or groupings in the data.

Each group formed is called a **cluster**, and the aim is to ensure that:

- Data points **within** the same cluster are more similar to each other.
- Data points **between** clusters are as different as possible.

Common Clustering Algorithms

- **K-Means**
- **Hierarchical Clustering**
- **DBSCAN**
- **Gaussian Mixture Models (GMM)**

Uses of Clustering in Business

Clustering is widely used in various business domains to segment data for strategic decision-making. Here are some key use cases:

1. **Customer Segmentation**

Goal: Group customers based on behavior to target marketing more effectively. **Example Dataset:** E-commerce data with features like purchase frequency, total spend, product categories, etc.

> A retail company might use K-Means clustering to segment customers into:
>
> - High-spenders
> - Occasional buyers
> - Deal seekers

These segments help in tailoring promotions and offers.

2. **Market Basket Analysis**

Goal: Identify patterns in buying behavior. **Example Dataset:** Transaction logs with items bought per customer.

> Clustering can group similar purchase patterns, such as customers who frequently buy electronics and accessories together, aiding in **product bundling** strategies.

3. **Anomaly Detection**

Goal: Detect unusual behavior, such as fraud or outlier transactions. **Example Dataset:** Banking transaction data including amount, time, location, and device used.

> If most customer behaviors fall into well-defined clusters, outliers can be flagged for investigation.

4. **Document or Content Categorization**

Goal: Automatically organize documents or articles by topic. **Example Dataset:** News articles or customer support tickets with text features.

> Companies can cluster customer support requests to identify common complaints or feedback themes.

5. **Location-Based Market Analysis**

Goal: Analyze store performance or customer distribution. **Example Dataset:** Geolocation data of customers or stores.

> A supermarket chain might use clustering to group stores by region and performance metrics for regional marketing strategies.

---

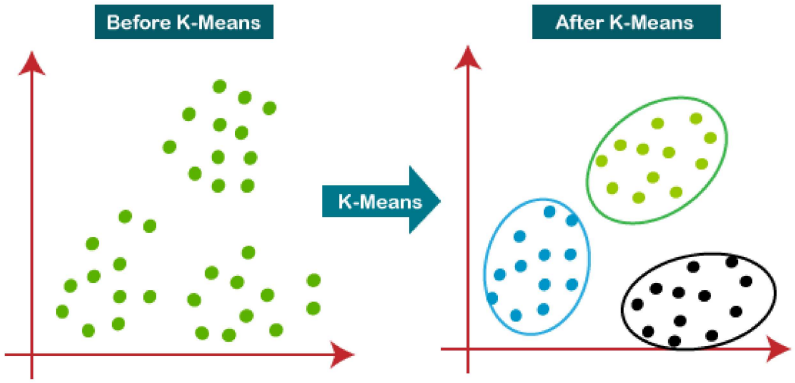
1. **Introduction to K-Means Clustering**

K-Means Clustering is an unsupervised learning algorithm used to group similar data points into a predefined number of clusters, K. The objective is to partition the dataset into K clusters, where each data point belongs to the cluster with the nearest mean, minimizing the

variance within each cluster.

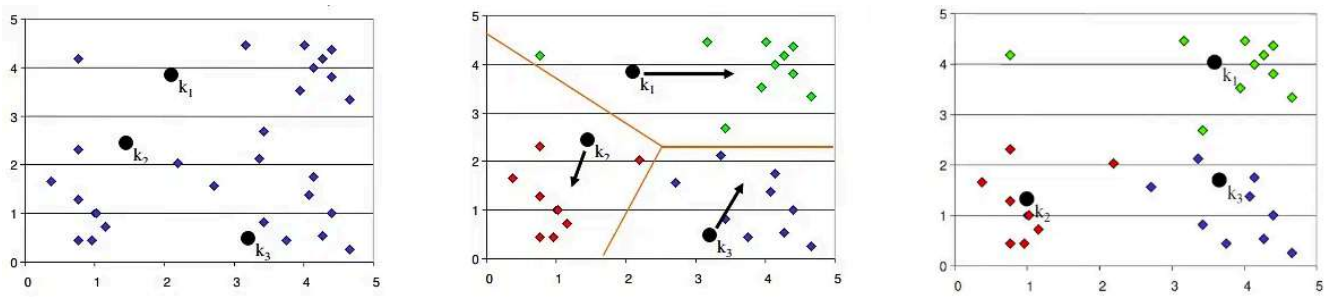
2. How K-Means Works

The K-Means algorithm works iteratively to partition a dataset into K clusters.



Here's how the algorithm proceeds:

- **Step 1**: Define the number K to decide the number of clusters.
- **Step 2**: Initialize K centroids **randomly**.
- **Step 3**: Assign each data point to the nearest centroid, forming K clusters.
- **Step 4**: Recalculate the centroids as the mean of all data points assigned to each cluster.
- **Step 5**: Repeat steps 2 and 3 until the centroids no longer change, or the algorithm converges.



[Reference: K-Means Clustering Tutorial]

---

3. Mathematical Explanation

Given a dataset X with n observations, the objective is to minimize the following **within-cluster sum of squares (WCSS)**:

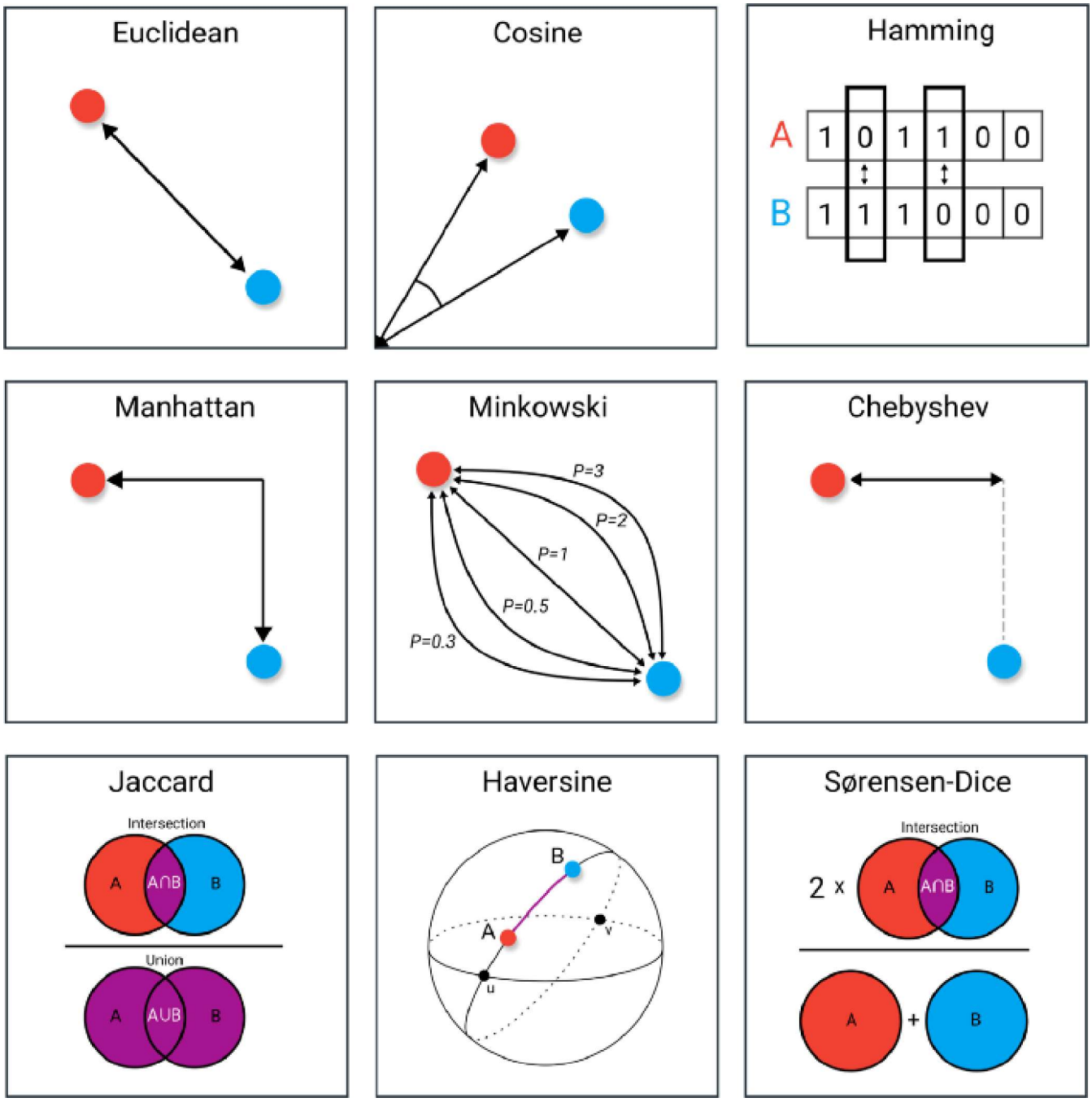
$$\sum_{i=1}^{K} \sum_{x \in C_i} \|x - \mu_i\|^2$$

Where:

- C_i is the set of points assigned to cluster i,
- μ_i is the centroid of cluster i,
- $\|x - \mu_i\|^2$ is the squared Euclidean distance between a point x and the centroid μ_i.

---

4. Common Distance measures



4.1. Euclidean Distance

- Definition: The straight-line distance between two points in Euclidean space.
- Formula:

$$\text{Euclidean Distance} = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$$

- Use Case: Commonly used in clustering (e.g., K-means) and nearest neighbor algorithms.

4.2. Cosine Similarity/Distance

- Definition: Measures the cosine of the angle between two vectors. It evaluates orientation rather than magnitude.
- Formula:

$$\cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|}$$

- Use Case: Used in text similarity and high-dimensional spaces (e.g., document similarity).

4.3. Hamming Distance

- Definition: Counts the number of positions at which corresponding symbols differ between two strings of equal length.
- Formula:
  Count mismatches between (A) and (B).
- Use Case: Applied in binary and categorical data, like error detection in coding.

4.4. Manhattan Distance

- Definition: The sum of absolute differences between the coordinates of two points.
- Formula:

$$\text{Manhattan Distance} = \sum_{i=1}^{n} |x_i - y_i|$$

- Use Case: Used when movement is restricted to horizontal and vertical paths (e.g., grid-based games or city blocks).

4.5. Minkowski Distance

- Definition: Generalized distance metric, including Euclidean ((p=2)) and Manhattan ((p=1)) distances as special cases.
- Formula:

$$\text{Minkowski Distance} = \left(\sum_{i=1}^{n} |x_i - y_i|^p \right)^{1/p}$$

- Use Case: Offers flexibility to adjust distance metrics based on parameter (p).

4.6. Chebyshev Distance

- Definition: The maximum absolute difference between two coordinates.
- Formula:

$$\text{Chebyshev Distance} = \max_i (|x_i - y_i|)$$

- Use Case: Used in chess for king's moves and grid-based pathfinding.

4.7. Jaccard Index/Distance

- Definition: Measures similarity or dissimilarity between two sets.
- Formula:

$$\text{Jaccard Index} = \frac{|A \cap B|}{|A \cup B|}$$

$$\text{Jaccard Distance} = 1 - \text{Jaccard Index}$$

- Use Case: Applied in text analysis, image segmentation, and set comparison.

4.8. Haversine Distance

- Definition: Calculates the shortest distance between two points on a sphere, considering curvature (e.g., Earth).
- Formula:

$$d = 2r \arcsin \sqrt{\sin^2\left(\frac{\Delta\phi}{2}\right) + \cos\phi_1 \cos\phi_2 \sin^2\left(\frac{\Delta\lambda}{2}\right)}$$

- Use Case: Used in geographic applications like GPS.

4.9. Sørensen-Dice Coefficient

- Definition: Measures the similarity between two sets.
- Formula:

$$\text{Sørensen-Dice Coefficient} = \frac{2|A \cap B|}{|A| + |B|}$$

- Use Case: Common in biology and text analysis.

5. **K-Means Clustering Step by Step Example**

Step 1: Euclidean Distance Formula

The Euclidean distance measures the straight-line distance between two points in a 2D space. The formula is:

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

We will use this formula to compute the distance between each point and the centroids.

Problem Setup

- **Data Points**: (x, y)
  - A1: (2, 10)
  - A2: (2, 5)
  - A3: (8, 4)
  - A4: (5, 8)
  - A5: (7, 5)
  - A6: (6, 4)

- A7: (1, 2)
- A8: (4, 9)

- **Initial Centroids**:
  - Mean 1: (2, 10)
  - Mean 2: (5, 8)
  - Mean 3: (1, 2)

We will calculate the distances between each point and the centroids and assign each point to the closest cluster.

Iteration 1

Steps:

1. **Compute Distances**: Use the Euclidean distance formula to calculate the distance from each point to all centroids.
2. **Assign Clusters**: Each point is assigned to the cluster of the nearest centroid.

Results:

- Distances and cluster assignments are shown in the table below.

Point	Coordinates	Dist Mean 1 (√((2−x)² + (10−y)²))	Dist Mean 2 (√((5−x)² + (8−y)²))	Dist Mean 3 (√((1−x)² + (2−y)²))	Cluster
A1	(2, 10)	√((2−2)² + (10−10)²) = 0.00	√((5−2)² + (8−10)²) = 3.61	√((1−2)² + (2−10)²) = 8.25	1
A2	(2, 5)	√((2−2)² + (10−5)²) = 5.00	√((5−2)² + (8−5)²) = 3.61	√((1−2)² + (2−5)²) = 3.16	3
A3	(8, 4)	√((2−8)² + (10−4)²) = 9.43	√((5−8)² + (8−4)²) = 5.00	√((1−8)² + (2−4)²) = 7.28	2
A4	(5, 8)	√((2−5)² + (10−8)²) = 3.61	√((5−5)² + (8−8)²) = 0.00	√((1−5)² + (2−8)²) = 6.71	2
A5	(7, 5)	√((2−7)² + (10−5)²) = 8.06	√((5−7)² + (8−5)²) = 3.61	√((1−7)² + (2−5)²) = 6.08	2
A6	(6, 4)	√((2−6)² + (10−4)²) = 8.94	√((5−6)² + (8−4)²) = 4.47	√((1−6)² + (2−4)²) = 5.39	2
A7	(1, 2)	√((2−1)² + (10−2)²) = 8.00	√((5−1)² + (8−2)²) = 7.62	√((1−1)² + (2−2)²) = 1.00	3
A8	(4, 9)	√((2−4)² + (10−9)²) = 2.24	√((5−4)² + (8−9)²) = 1.41	√((1−4)² + (2−9)²) = 7.07	2

Using the provided table, the **new centroids** are computed as follows:

Cluster Assignments

From the table:

- **Cluster 1**: A1 (Coordinates: $(2, 10)$)
- **Cluster 2**: A3, A4, A5, A6, A8 (Coordinates: $(8, 4), (5, 8), (7, 5), (6, 4), (4, 9)$)
- **Cluster 3**: A2, A7 (Coordinates: $(2, 5), (1, 2)$)

Centroid Calculations

1. **Centroid for Cluster 1**: Since Cluster 1 has only one point, the centroid remains at the coordinates of A1: $\text{Centroid}_1 = (2.0, 10.0)$
2. **Centroid for Cluster 2**: $\text{Centroid}_2 = \left(\frac{8+5+7+6+4}{5}, \frac{4+8+5+4+9}{5}\right)$ $\text{Centroid}_2 = (6.0, 6.0)$
3. **Centroid for Cluster 3**: $\text{Centroid}_3 = \left(\frac{2+1}{2}, \frac{5+2}{2}\right)$ $\text{Centroid}_3 = (1.5, 3.5)$

Updated Centroids for Iteration 1

- **Cluster 1**: $(2.0, 10.0)$
- **Cluster 2**: $(6.0, 6.0)$
- **Cluster 3**: $(1.5, 3.5)$

Summary

- The centroids **has changed** after the first iteration because they were updated based on the new cluster assignments.
- The changes are as follows:
  - Cluster 1: No change (only one point).
  - Cluster 2: Adjusted to $(6.0, 6.0)$.
  - Cluster 3: Adjusted to $(1.5, 3.5)$.

Iteration 2

Steps:

1. **Recalculate Centroids**: For each cluster, compute the new centroid as the mean of all points in that cluster.
2. **Reassign Clusters**: Recompute distances to the updated centroids and assign points to the nearest cluster.

Results:

- Distances and new cluster assignments are shown in the table below.
- **Used Centroids**: $(2.0, 10.0)$, $(6.0, 6.0)$, $(1.5, 3.5)$

Point	Coordinates	Dist Mean 1 (√((x1−x)² + (y1−y)²))	Dist Mean 2 (√((x2−x)² + (y2−y)²))	Dist Mean 3 (√((x3−x)² + (y3−y)²))	Cluster
A1	(2, 10)	√((2−2)² + (10−10)²) = 0.00	√((6−2)² + (6−10)²) = 5.39	√((1.5−2)² + (3.5−10)²) = 8.83	1
A2	(2, 5)	√((2−2)² + (10−5)²) = 5.00	√((6−2)² + (6−5)²) = 4.47	√((1.5−2)² + (3.5−5)²) = 2.50	3
A3	(8, 4)	√((2−8)² + (10−4)²) = 9.22	√((6−8)² + (6−4)²) = 3.61	√((1.5−8)² + (3.5−4)²) = 6.86	2
A4	(5, 8)	√((2−5)² + (10−8)²) = 3.61	√((6−5)² + (6−8)²) = 2.24	√((1.5−5)² + (3.5−8)²) = 6.92	2
A5	(7, 5)	√((2−7)² + (10−5)²) = 8.06	√((6−7)² + (6−5)²) = 2.24	√((1.5−7)² + (3.5−5)²) = 5.92	2
A6	(6, 4)	√((2−6)² + (10−4)²) = 8.94	√((6−6)² + (6−4)²) = 2.83	√((1.5−6)² + (3.5−4)²) = 5.22	2
A7	(1, 2)	√((2−1)² + (10−2)²) = 8.06	√((6−1)² + (6−2)²) = 7.21	√((1.5−1)² + (3.5−2)²) = 1.80	3
A8	(4, 9)	√((2−4)² + (10−9)²) = 2.24	√((6−4)² + (6−9)²) = 3.61	√((1.5−4)² + (3.5−9)²) = 6.80	1

Explanation

- **Dist Mean 1**: Calculated using the first centroid ((2, 10)).
- **Dist Mean 2**: Calculated using the second centroid ((6, 6)).
- **Dist Mean 3**: Calculated using the third centroid ((1.5, 3.5)).
- **Cluster**: Points are assigned to the cluster of the closest centroid (smallest distance).

Comments:

- Notice how centroids shift toward the center of their clusters.
- Some points might change clusters as centroids update.

Formula for New Centroids

For each cluster, the new centroid is calculated as the mean of the (x)- and (y)-coordinates of all points in the cluster:

$$\text{Centroid}_i = \left(\frac{\sum x_{\text{cluster}}}{\text{n}}, \frac{\sum y_{\text{cluster}}}{\text{n}}\right)$$

Cluster Assignments

Based on the given table:

- **Cluster 1**: A1, A8 (Coordinates: ((2, 10), (4, 9)))
- **Cluster 2**: A3, A4, A5, A6 (Coordinates: ((8, 4), (5, 8), (7, 5), (6, 4)))
- **Cluster 3**: A2, A7 (Coordinates: ((2, 5), (1, 2)))

Centroid Calculations

Cluster 1 (Centroid 1):

$$\text{Centroid}_1 = \left(\frac{2+4}{2}, \frac{10+9}{2}\right) = (3.0, 9.5)$$

Cluster 2 (Centroid 2):

$$\text{Centroid}_2 = \left(\frac{8+5+7+6}{4}, \frac{4+8+5+4}{4}\right) \text{Centroid}_2 = \left(\frac{26}{4}, \frac{21}{4}\right) = (6.5, 5.25)$$

Cluster 3 (Centroid 3):

$$\text{Centroid}_3 = \left(\frac{2+1}{2}, \frac{5+2}{2}\right) = (1.5, 3.5)$$

Updated Centroids

1. **Centroid 1**: ((3.0, 9.5))
2. **Centroid 2**: ((6.5, 5.25))
3. **Centroid 3**: ((1.5, 3.5))

These centroids are the updated positions for the next iteration of the K-Means algorithm.

Iteration 3

Steps:

1. **Recompute Centroids**: Repeat the process of recalculating centroids.
2. **Reassign Clusters**: Update cluster assignments based on new centroids.

Results:

- Distances and cluster assignments are shown in the table below.
- **Used Centroids**: $(3.0, 9.5)$, $(6.5, 5.25)$, $(1.5, 3.5)$

Point	Coordinates	Dist Mean 1	Dist Mean 2	Dist Mean 3	Cluster
A1	(2, 10)	1.12	5.94	8.83	1
A2	(2, 5)	4.53	4.18	2.50	3
A3	(8, 4)	8.86	3.00	6.86	2
A4	(5, 8)	2.55	2.28	6.92	2
A5	(7, 5)	7.90	2.05	5.92	2
A6	(6, 4)	8.55	2.65	5.22	2
A7	(1, 2)	7.55	7.07	1.80	3
A8	(4, 9)	1.80	3.74	6.80	1

Comments:

- Observe how centroids continue to adjust.
- Points begin to stabilize in their respective clusters.

Iteration 4

Steps:

1. **Recalculate Centroids**: Adjust centroids as the mean of their cluster points.
2. **Check for Convergence**: If centroids do not change, the algorithm stops.

Results:

- Distances and final cluster assignments are shown in the table below.
- **Final Centroids**: $(3.67, 9.0)$, $(7.0, 4.33)$, $(1.5, 3.5)$

Point	Coordinates	Dist Mean 1	Dist Mean 2	Dist Mean 3	Cluster
A1	(2, 10)	1.60	6.12	8.83	1
A2	(2, 5)	4.12	4.01	2.50	3
A3	(8, 4)	8.64	2.62	6.86	2
A4	(5, 8)	2.36	2.03	6.92	2
A5	(7, 5)	7.68	1.75	5.92	2
A6	(6, 4)	8.28	2.35	5.22	2
A7	(1, 2)	7.23	7.07	1.80	3
A8	(4, 9)	1.41	3.77	6.80	1

Comments:

- The algorithm converges as centroids stabilize and no further changes occur.

Final Results

- Each point has been assigned to its nearest cluster.
- The process has converged after 4 iterations.

Key Observations

1. **Centroid Movement**: Centroids progressively adjust to the cluster's center during each iteration.
2. **Convergence**: K-Means stops when the centroids no longer move.

---

6. Python Code Example

Here's how to implement K-Means clustering in Python using `scikit-learn` and Mall Customers Information
We will use a using [Mall Customers Dataset]() to demonstrate kmeans clustering using scikit Learn Library.

Index	CustomerID	Genre	Age	Annual Income (k$)	Spending Score (1-100)
0	1	Male	19	15	39
1	2	Male	21	15	81
2	3	Female	20	16	6
3	4	Female	23	16	77
4	5	Female	31	17	40
5	6	Female	22	17	76
6	7	Female	35	18	6

Index	CustomerID	Genre	Age	Annual Income (k$)	Spending Score (1-100)
7	8	Female	23	18	94
8	9	Male	64	19	3
9	10	Female	30	19	72
10	11	Male	67	19	14
11	12	Female	35	20	99
12	13	Male	58	20	15
13	14	Female	24	20	77
14	15	Male	22	20	79

Import Libraries and Dataset

```
# Import necessary libraries
import pandas as pd
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt

# Step 1: Get the data
df = pd.read_csv('https://raw.githubusercontent.com/msfasha/307304-Data-Mining/refs/heads/main/20242/datasets/mall_customers.csv')
df
```

	CustomerID	Gender	Age	Annual Income (k$)	Spending Score (1-100)
0	1	Male	19	15	39
1	2	Male	21	15	81
2	3	Female	20	16	6
3	4	Female	23	16	77
4	5	Female	31	17	40
...	...	...	...	...	...
195	196	Female	35	120	79
196	197	Female	45	126	28
197	198	Male	32	126	74
198	199	Male	32	137	18
199	200	Male	30	137	83

200 rows × 5 columns

Next steps: Generate code with df | View recommended plots | New interactive sheet

Apply K-Means with K = 3

```
# Step 2: Select relevant features (Annual Income and Spending Score) for clustering
features = df[["Annual Income (k$)", "Spending Score (1-100)"]]

# Step 3: Apply K-Means with K=3
kmeans = KMeans(n_clusters=3, random_state=42)
kmeans.fit(features)

# Step 4: Add the cluster labels to the DataFrame
df["Cluster"] = kmeans.labels_
```

Print the updated dataframe showing the assigned cluster for each record

```
df
```

	CustomerID	Gender	Age	Annual Income (k$)	Spending Score (1-100)	Cluster
0	1	Male	19	15	39	2
1	2	Male	21	15	81	2
2	3	Female	20	16	6	2
3	4	Female	23	16	77	2
4	5	Female	31	17	40	2
...	...	...	...	...	...	...
195	196	Female	35	120	79	1
196	197	Female	45	126	28	0
197	198	Male	32	126	74	1
198	199	Male	32	137	18	0
199	200	Male	30	137	83	1

200 rows × 6 columns

Next steps: Generate code with df | View recommended plots | New interactive sheet

Print out the centroids of the clusters

```
kmeans.cluster_centers_
```

```
array([[87.        , 18.63157895],
       [86.53846154, 82.12820513],
       [44.15447154, 49.82926829]])
```

Format centroids in a dataframe for better visualization

```
# Assuming features and KMeans object already defined
centroids = pd.DataFrame(kmeans.cluster_centers_, columns=features.columns)

# Display centroids for interpretation
print("Centroids for each cluster:")
centroids
```

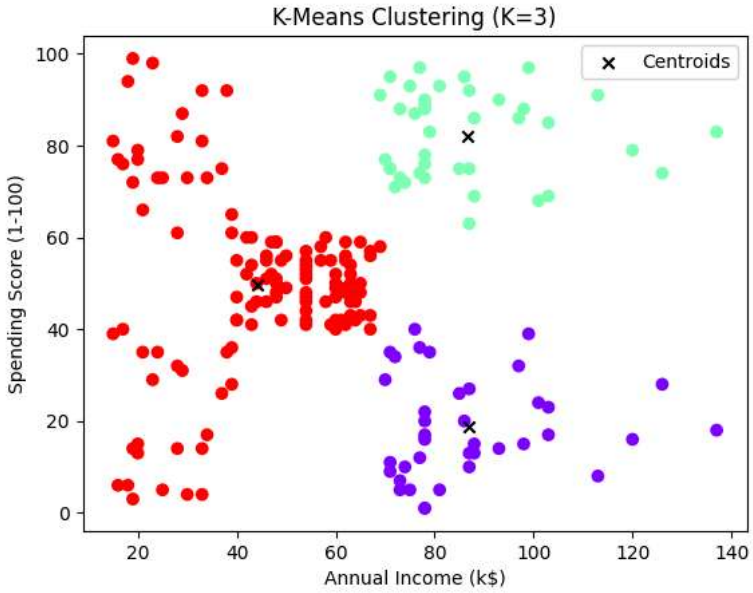
```
Centroids for each cluster:
```

	Annual Income (k$)	Spending Score (1-100)	Age
0	59.879032	35.427419	44.483871
1	88.731707	79.243902	32.975610
2	29.971429	68.514286	25.771429

Next steps: Generate code with centroids | View recommended plots | New interactive sheet

Display The Scatter Plot and the DataFrame

```
# Step 5: Plot the clusters and centroids
plt.scatter(features["Annual Income (k$)"], features["Spending Score (1-100)"], c=df["Cluster"], cmap="rainbow")
plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1], color="black", marker="x", label="Centroids")
plt.title("K-Means Clustering (K=3)")
plt.xlabel("Annual Income (k$)")
plt.ylabel("Spending Score (1-100)")
plt.legend()
plt.show()
```



Observations for (k=3):

1. **Cluster Formation**:
   - The data points are divided into **three distinct clusters**, differentiated by their spending scores and annual incomes.
   - The clusters reflect distinct patterns in customer behavior.
2. **Cluster Characteristics**:
   - **Cluster 1 (e.g., Red)**:
     - Represents customers with **low to moderate income** and a **wide range of spending scores**.
     - Includes customers who might be more price-conscious or have varied spending behaviors.
   - **Cluster 2 (e.g., Purple)**:
     - Comprises customers with **high income** but **low spending scores**.
     - Indicates potentially savings-oriented individuals, despite higher incomes.
   - **Cluster 3 (e.g., Green)**:
     - Includes customers with **high income** and **high spending scores.**
     - Likely represents high-value customers who are ideal for premium marketing campaigns.
3. **Centroid Placement**:
   - The centroids (black "X" markers) represent the mean position of each cluster.
   - The distance between centroids suggests clear separation between the groups.
4. **Insights**:
   - This segmentation aligns well with typical customer segmentation strategies in marketing:
     - **Cluster 1**: Target with discounts or budget-friendly options.
     - **Cluster 2**: Upsell higher-end products or encourage spending.
     - **Cluster 3**: Retain as loyal high-value customers with exclusive offers.
5. **Potential Next Steps**:
   - Validate the clusters with real-world customer behavior (e.g., demographic or shopping preferences).
   - Experiment with **k=4** to check for further sub-clusters.
   - Perform **3D clustering** by adding another feature (e.g., Age or Gender).

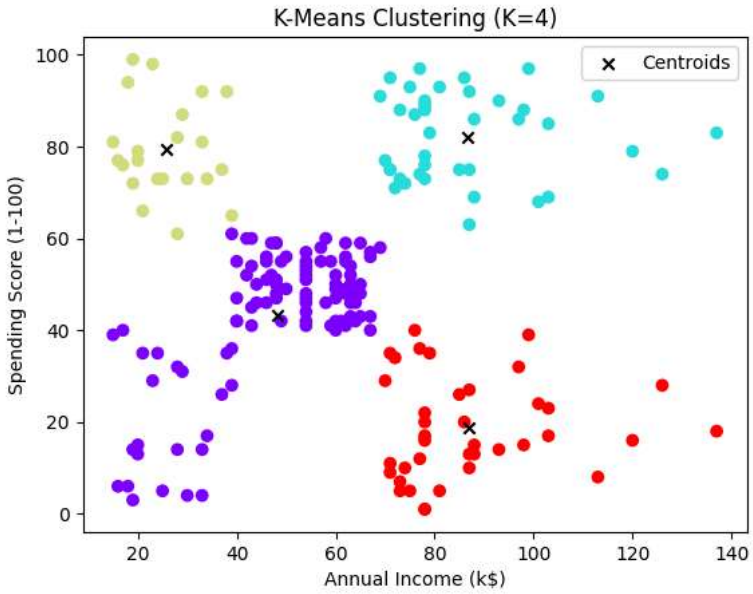
Apply K-Means with K = 4

```
# Select relevant features (Annual Income and Spending Score) for clustering
features = df[["Annual Income (k$)", "Spending Score (1-100)"]]

# Apply K-Means with K=4
kmeans = KMeans(n_clusters=4, random_state=42)
kmeans.fit(features)

# Add assigned clusters to each row
df["Cluster"] = kmeans.labels_
```

```
# Plot the clusters and centroids
plt.scatter(features["Annual Income (k$)"], features["Spending Score (1-100)"], c=df["Cluster"], cmap="rainbow")
plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1], color="black", marker="x", label="Centroids")
plt.title("K-Means Clustering (K=4)")
plt.xlabel("Annual Income (k$)")
plt.ylabel("Spending Score (1-100)")
plt.legend()
plt.show()
```



With (k=4) clusters, the customer segmentation becomes more refined. Here are the observations and insights:

Observations

1. **Cluster Formation**:
   - The data is divided into four distinct clusters, with clear separation along **Annual Income (k$)** and **Spending Score (1-100)**.
2. **Cluster Characteristics**:
   - **Cluster 1 (Light Green)**:
     - **Low Annual Income**, **High Spending Score**.
     - Represents customers who spend a large proportion of their income, possibly impulsive spenders or younger customers.
   - **Cluster 2 (Purple)**:
     - **Moderate Annual Income**, **Moderate Spending Score**.
     - Likely represents the "average" customer with balanced income and spending behavior.
   - **Cluster 3 (Light Blue)**:
     - **High Annual Income**, **High Spending Score**.
     - Represents premium, high-value customers with significant purchasing power.
   - **Cluster 4 (Purple)**:
     - **High Annual Income**, **Low Spending Score**.
     - Likely represents frugal or savings-oriented individuals who may prioritize saving over spending despite their high income.
3. **Centroids**:
   - The centroids (black "X" markers) represent the average characteristics of each cluster.
   - Clear separation of centroids confirms meaningful clusters.

Insights

1. **Marketing Strategies**:
   - **Cluster 1 (Light Green)**:
     - Target with discounts, promotions, or budget-friendly offers to maximize spending potential.
   - **Cluster 2 (Purple)**:
     - Maintain regular engagement, as they represent a balanced customer segment.

- **Cluster 3 (Light Blue)**:
  - Focus on premium products and services. These customers are ideal for exclusive campaigns.
- **Cluster 4 (Purple)**:
  - Consider introducing tailored promotions or incentivizing spending for this segment.

2. **Segmentation Insights**:
   - Including (k=4) provides finer segmentation, highlighting distinct behavioral groups like "impulsive spenders" and "savings-oriented high-income customers."
   - This segmentation can inform personalized marketing and product offerings.
3. **Potential for Further Analysis**:
   - **3D Clustering**: Add a third feature like **Age** to see if it further refines the clusters.
   - **Exploring Gender**: Overlay gender information to analyze behavioral patterns by gender.

---

8. Cluster Data Using 3 Features

```
# Step 2: Select relevant features (Annual Income and Spending Score) for clustering
features = df[["Annual Income (k$)", "Spending Score (1-100)", "Age"]]

# Step 3: Apply K-Means with K=3
kmeans = KMeans(n_clusters=3, random_state=42)
kmeans.fit(features)

# Step 4: Add the cluster labels to the DataFrame
df["Cluster"] = kmeans.labels_

df
```

	CustomerID	Gender	Age	Annual Income (k$)	Spending Score (1-100)	Cluster
0	1	Male	19	15	39	2
1	2	Male	21	15	81	2
2	3	Female	20	16	6	0
3	4	Female	23	16	77	2
4	5	Female	31	17	40	2
...	...	...	...	...	...	...
195	196	Female	35	120	79	1
196	197	Female	45	126	28	1
197	198	Male	32	126	74	1
198	199	Male	32	137	18	1
199	200	Male	30	137	83	1

200 rows × 6 columns

Next steps: Generate code with df | View recommended plots | New interactive sheet

Display Centroids

```
centroids = pd.DataFrame(kmeans.cluster_centers_, columns=features.columns)

# Display centroids for interpretation
print("Centroids for each cluster:")
print(centroids)
```

```
Centroids for each cluster:
   Annual Income (k$)  Spending Score (1-100)        Age
0           59.879032               35.427419  44.483871
1           88.731707               79.243902  32.975610
2           29.971429               68.514286  25.771429
```

3D Scatter Plot

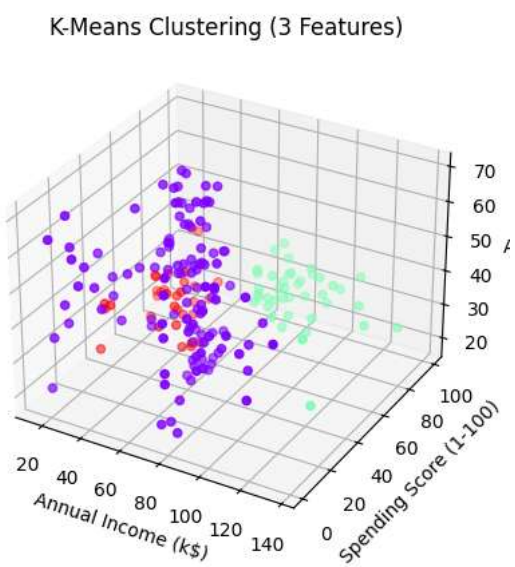
We can use a 3D scatter plot to visualize three features together.

```
from mpl_toolkits.mplot3d import Axes3D

fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')

# Plotting points
ax.scatter(
    df["Annual Income (k$)"],
    df["Spending Score (1-100)"],
    df["Age"],
    c=df["Cluster"], cmap="rainbow"
)

# Label axes
ax.set_title("K-Means Clustering (3 Features)")
ax.set_xlabel("Annual Income (k$)")
ax.set_ylabel("Spending Score (1-100)")
ax.set_zlabel("Age")
plt.show()
```

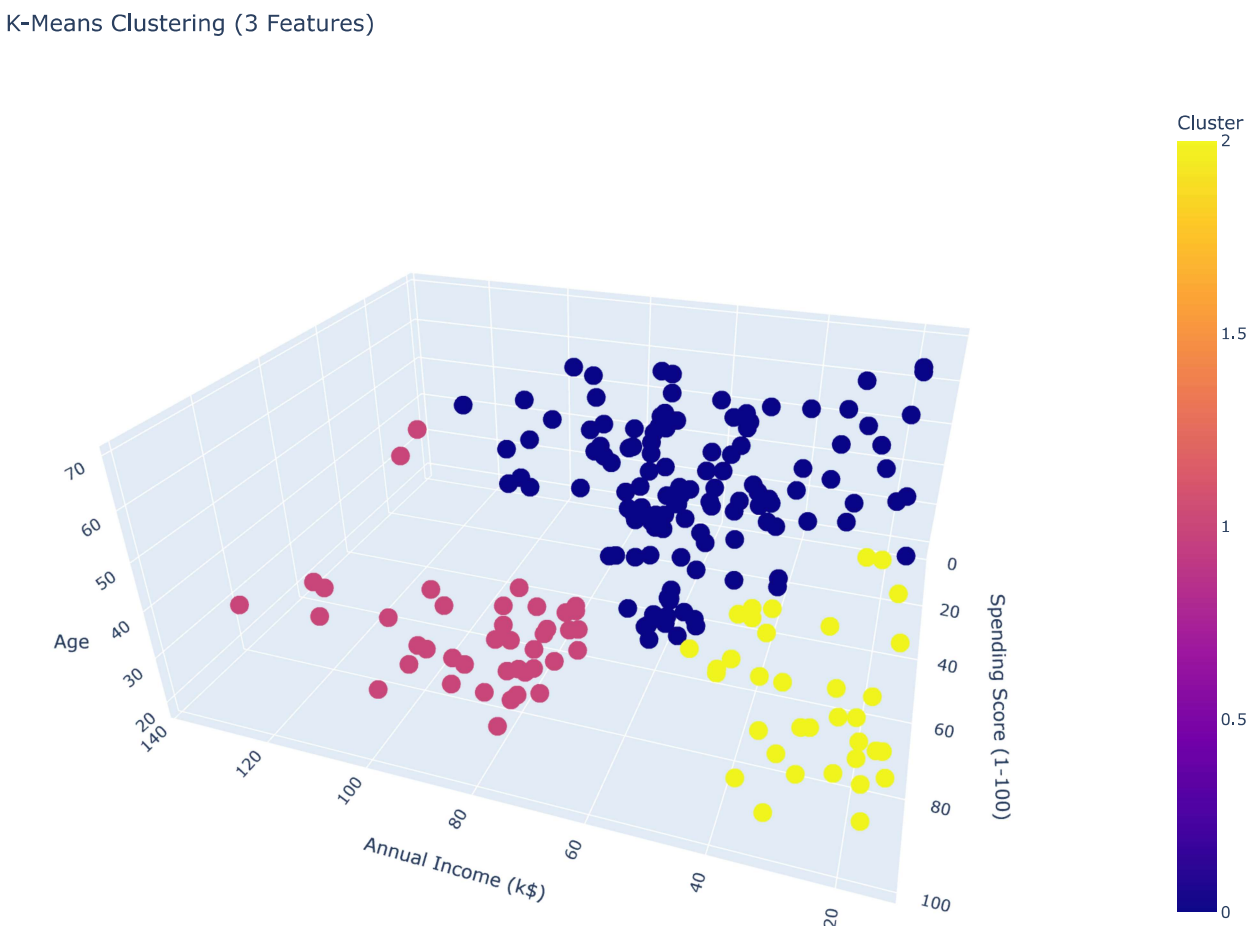


Use Plotly Package to Create Dynamic Plots

```
import plotly.express as px

# Create an interactive 3D scatter plot
fig = px.scatter_3d(
    df,
    x="Annual Income (k$)",
    y="Spending Score (1-100)",
    z="Age",
    color="Cluster",  # Color points by cluster
    title="K-Means Clustering (3 Features)",
    labels={
        "Annual Income (k$)": "Annual Income (k$)",
        "Spending Score (1-100)": "Spending Score (1-100)",
        "Age": "Age"
    },
    width=1000,  # Set the width of the plot
    height=800   # Set the height of the plot
)

# Show the interactive plot
fig.show()
```

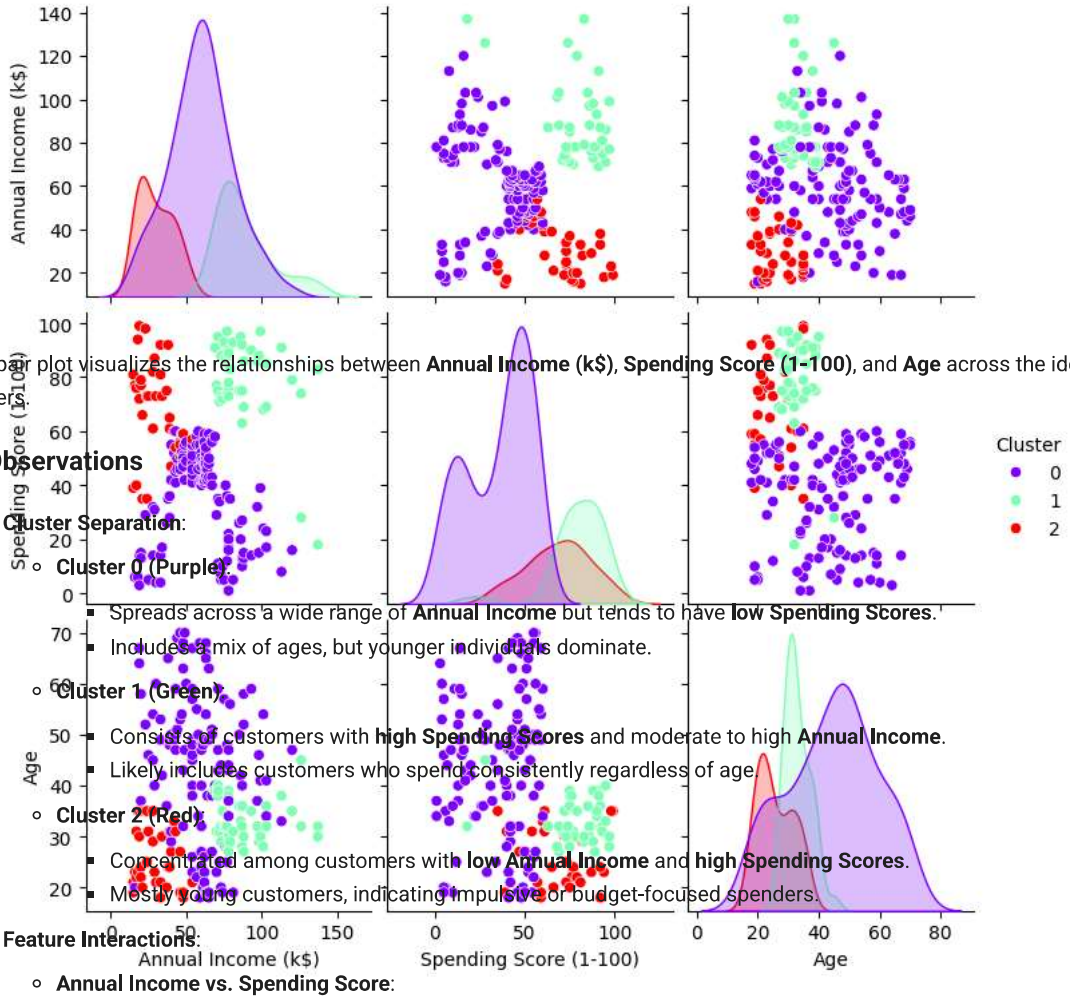


Pairwise Scatter Plots

We can create scatter plots for each pair of the three features, highlighting clusters using different colors.

```
import seaborn as sns

# Add Cluster information to the dataframe
sns.pairplot(df, vars=["Annual Income (k$)", "Spending Score (1-100)", "Age"], hue="Cluster", palette="rainbow")
plt.show()
```



This pair plot visualizes the relationships between **Annual Income (k$)**, **Spending Score (1-100)**, and **Age** across the identified customer clusters.

Observations

1. **Cluster Separation**:
   - **Cluster 0 (Purple)**:
     - Spreads across a wide range of **Annual Income** but tends to have **low Spending Scores**.
     - Includes a mix of ages, but younger individuals dominate.
   - **Cluster 1 (Green)**:
     - Consists of customers with **high Spending Scores** and moderate to high **Annual Income**.
     - Likely includes customers who spend consistently regardless of age.
   - **Cluster 2 (Red)**:
     - Concentrated among customers with **low Annual Income** and **high Spending Scores**.
     - Mostly young customers, indicating impulsive or budget-focused spenders.
2. **Feature Interactions**:
   - **Annual Income vs. Spending Score**:
     - Cluster 0 includes customers with higher income but lower spending.
     - Cluster 1 indicates high-spending customers across various income levels.
     - Cluster 2 reflects high spending at lower income levels, highlighting a younger demographic.
   - **Age**:
     - Younger customers dominate **Cluster 2**, while **Cluster 1** has a broader age range.
     - Older customers tend to appear in **Cluster 0**, indicating cautious spending habits.

Recommendations

1. **Targeted Marketing**:
   - **Cluster 0 (Low Spending, Higher Income)**:
     - Encourage spending with tailored promotions or loyalty programs.
     - Highlight premium products to appeal to their income level.
   - **Cluster 1 (High Spending, Various Income)**: