**K-Means Clustering – Complete Guide for Students**

**🔍 What is Clustering?**

**Clustering** is a type of **Unsupervised Learning** — that means:

* No labels or target values.
* The algorithm finds patterns or groups **on its own**.
* Goal: Group similar data points together based on their features.

**💡 What is K-Means Clustering?**

**K-Means** is the most popular clustering algorithm.

* It groups the data into **K clusters**.
* Each cluster has a **centroid** (center).
* Each data point is assigned to the **nearest centroid**.
* The centroids are updated until the grouping **stabilizes**.

**🛍️ Real-Life Example: Customer Segmentation**

Let’s say you own a **shopping mall**.

You want to group your customers based on:

* Age
* Spending score

🎯 **Goal**: Find different types of customers like:

* Budget buyers
* Medium spenders
* High-end spenders

But you don’t know these groups in advance. That’s where K-Means comes in — it will discover these natural groupings for you.

**🛠️ How K-Means Works (Step-by-Step)**

Let’s say you want to create **K = 3 clusters**.

**Step 1: Choose K**

You decide how many groups (K) you want.

**Step 2: Initialize Centroids**

Randomly select **K data points** as initial **centroids** (the center of each cluster).

**Step 3: Assign Points**

Assign each data point to the **nearest centroid** using a distance measure (like Euclidean distance).

**Step 4: Update Centroids**

Move the centroids to the **average position** of all the points in that cluster.

**Step 5: Repeat**

Repeat steps 3 and 4 until:

* Cluster assignments don’t change.
* Or a maximum number of iterations is reached.

**📐 Distance Formula: Euclidean Distance**

Distance=(x1−x2)2+(y1−y2)2\text{Distance} = \sqrt{(x\_1 - x\_2)^2 + (y\_1 - y\_2)^2}Distance=(x1​−x2​)2+(y1​−y2​)2​

Used to measure how “close” a point is to a centroid.

**📊 Python Example**

python

CopyEdit

import pandas as pd

from sklearn.cluster import KMeans

import matplotlib.pyplot as plt

# Sample data

data = {

'Age': [18, 22, 24, 27, 30, 35, 40, 45, 52, 60],

'Spending\_Score': [90, 86, 80, 77, 60, 45, 40, 30, 20, 15]

}

df = pd.DataFrame(data)

# Apply K-Means

kmeans = KMeans(n\_clusters=3, random\_state=0)

df['Cluster'] = kmeans.fit\_predict(df[['Age', 'Spending\_Score']])

# Plot clusters

plt.scatter(df['Age'], df['Spending\_Score'], c=df['Cluster'], cmap='viridis')

plt.scatter(kmeans.cluster\_centers\_[:, 0], kmeans.cluster\_centers\_[:, 1], color='red', marker='X', s=200)

plt.title("Customer Segments")

plt.xlabel("Age")

plt.ylabel("Spending Score")

plt.show()

**❓ How to Choose the Best K? (Elbow Method)**

1. Try K values from 1 to 10.
2. For each K, calculate **SSE** (Sum of Squared Errors).
3. Plot SSE vs K.
4. Look for the “**elbow**” — the point where the decrease in SSE slows down. That’s the optimal K.

**📘 Key Terms for Students**

| **Term** | **Meaning** |
| --- | --- |
| **Centroid** | Center point of a cluster |
| **Cluster** | A group of similar data points |
| **Inertia** | Sum of squared distances to centroids (SSE) |
| **Unsupervised** | No labels, algorithm finds structure itself |
| **Scalability** | Works well on large datasets |

**✅ When to Use K-Means**

Use it when:

* You want to find **natural groups** in data.
* The data is **numerical**.
* You need a **simple and fast** clustering algorithm.

**⚠️ Limitations of K-Means**

* You must decide K in advance.
* Doesn’t work well with **non-spherical clusters**.
* Sensitive to **outliers** and **initial centroid placement**.

**🧾 Summary Table**

| **Feature** | **K-Means Clustering** |
| --- | --- |
| Type | Unsupervised Learning |
| Input | Unlabeled numerical data |
| Output | Cluster assignments |
| Main Use Cases | Customer segmentation, document clustering, image compression |
| Strength | Simple, fast, scalable |
| Weakness | Fixed K, bad with complex shapes or outliers |

**Example Dataset: Customer Segmentation**

| **Customer\_ID** | **Age** | **Annual Income (k$)** | **Spending Score (1-100)** |
| --- | --- | --- | --- |
| 1 | 19 | 15 | 39 |
| 2 | 21 | 15 | 81 |
| 3 | 20 | 16 | 6 |
| 4 | 23 | 16 | 77 |
| 5 | 31 | 17 | 40 |
| 6 | 22 | 17 | 76 |
| 7 | 35 | 18 | 6 |
| 8 | 23 | 18 | 94 |
| 9 | 64 | 19 | 3 |
| 10 | 30 | 19 | 72 |
| 11 | 67 | 19 | 14 |
| 12 | 35 | 20 | 99 |
| 13 | 58 | 20 | 15 |
| 14 | 24 | 21 | 77 |
| 15 | 37 | 21 | 40 |

**Data Description:**

* **Customer\_ID**: Unique identifier for each customer.
* **Age**: The age of the customer.
* **Annual Income**: The annual income of the customer in thousands of dollars.
* **Spending Score**: A score assigned based on the spending behavior of the customer (on a scale of 1-100).