Unsupervised Machine Learning

Unsupervised learning, also known as unsupervised machine learning, uses machine learning (ML) algorithms to analyze and cluster unlabeled data sets. These algorithms discover hidden patterns or data groupings without the need for human intervention.

Unsupervised learning is a type of machine learning where the algorithm is trained on unlabeled data—meaning the data doesn't come with predefined outputs or categories.

The goal is for the model to find patterns, structures, or relationships within the data on its own.

Key Characteristics:

- **No labeled output**: Unlike supervised learning, there's no correct answer provided.
- **Pattern discovery**: The model identifies hidden patterns without explicit instruction

Common unsupervised learning approaches

- Clustering
- Association Rules
- Dimensionality reduction

Clustering

Clustering is the most common unsupervised learning method and helps you understand the natural grouping or inherent structure of a data set.

It is used for exploratory data analysis, pattern recognition, anomaly detection, image segmentation, and more.

Clustering algorithms, such as *k*-means or hierarchical clustering, group data points such that data points in the same group (or cluster) are more similar to each other than to data points in other groups.

For example, if a cell phone company wants to optimize the locations where it builds cell phone towers, it can use machine learning to estimate the number of clusters of people relying on its towers.

A phone can only talk to one tower at a time, so the team uses clustering algorithms to design the best placement of cell towers to optimize signal reception for groups, or clusters, of customers.

Objective:

Find a structure or pattern in a collection of unlabeled data without prior knowledge of class labels

Definition

Given a dataset X={x1,x2,...,xn} where each data point xi∈ R, the task is to partition the data into k clusters C={C1,C2,...,Ck} such that:

$$\bigcup_{i=1}^{k} C_i = X \quad \text{and} \quad C_i \cap C_j = \emptyset \quad \text{for } i \neq j$$

Types of Clustering

1. Partitioning Clustering
   Divides the data into a fixed number **k** of non-overlapping clusters.

   K-Means Clustering

K-Means is a way to group similar things together — and it does it automatically.

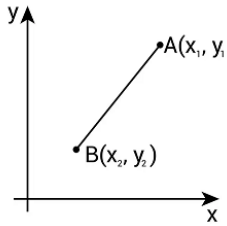
Customer Segmentation

No	Age	Amount
C1	20	500
C2	40	1000
C3	30	800
C4	18	300
C5	28	1200
C6	35	1400
C7	45	1800

now devide cluster based on spending hablit

initiate centroid of each cluster

Distance Formula



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

Minimum value reach particular cluster and become an updated new centroid.

CREATIVE INSTITUTE DATA SCIENCE

Imagine you run a **shopping mall**, and you have data about your customers:

- Age
- Income
- Spending habit

You want to divide customers into **3 types** (K = 3):

1. Low spenders
2. Medium spenders
3. High spenders

You don’t know who belongs where — you want the **algorithm to find out**!

Step 1: Choose K (number of groups)

Let's say K = 3 (you want 3 groups)

Step 2: Pick 3 random points to start (centers of groups)

Step 3: Put each person into the group whose center is nearest

Step 4: Find the average position of each group — that becomes the new center

Step 5: Move people to new closest centers

Step 6: Repeat steps 4 and 5 until nobody changes groups

Suppose these are your customers:

You own a **shopping mall**, and you want to understand your **customers** based on:

- Their **Annual Income**
- Their **Spending Score** (How much they usually spend)

Your goal is to **group customers into 3 categories**:

- Low spenders
- Average spenders
- High spenders

You don't know these groups already — you want the **K-Means algorithm to find them**.

CustomerID	Income (₹ in 1000s)	Spending Score
1	15	39
2	16	81
3	17	6
4	18	77
5	19	40
6	25	6
7	35	50
8	45	95
9	50	7
10	54	60

```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
```

```
data = {
    'CustomerID': [1,2,3,4,5,6,7,8,9,10],
    'Income': [15,16,17,18,19,25,35,45,50,54],
    'SpendingScore': [39,81,6,77,40,6,50,95,7,60]
}
```

```
df = pd.DataFrame(data)
```

```
X = df[['Income', 'SpendingScore']]
```

```
kmeans = KMeans(n_clusters=3, random_state=0)
```

```
df['Cluster'] = kmeans.fit_predict(X)
```

```
plt.figure(figsize=(8,6))
colors = ['red', 'blue', 'green']
for cluster in range(3):
```

```
    cluster_data = df[df['Cluster'] == cluster]
    plt.scatter(cluster_data['Income'], cluster_data['SpendingScore'],
                label=f'Cluster {cluster+1}', color=colors[cluster])
```

```
centroids = kmeans.cluster_centers_

plt.scatter(centroids[:, 0], centroids[:, 1], s=200, c='black', marker='X',
label='Centroids')

plt.title('Customer Segments by Income and Spending Score')
plt.xlabel('Annual Income (₹1000s)')
plt.ylabel('Spending Score')
plt.legend()
plt.grid(True)
plt.show()
```

- 3 clusters of customers in different colors.
- A black "X" at the center of each group (cluster centroid).
- Each group represents a different type of customer based on spending behavior and income.

- One group might be **low income, low spending** → Budget shoppers
- Another might be **high income, high spending** → Premium customers
- One might be **average income, moderate spending** → Casual buyers

Use of Customer Segmentation

You can **target each group differently**:

Cluster	Behavior	Marketing Strategy
1	Low spenders	Offer discounts, promotions
2	High spenders	Recommend premium products, loyalty rewards
3	Occasional buyers	Send reminders, engagement offers

This helps you **increase sales** without wasting budget.