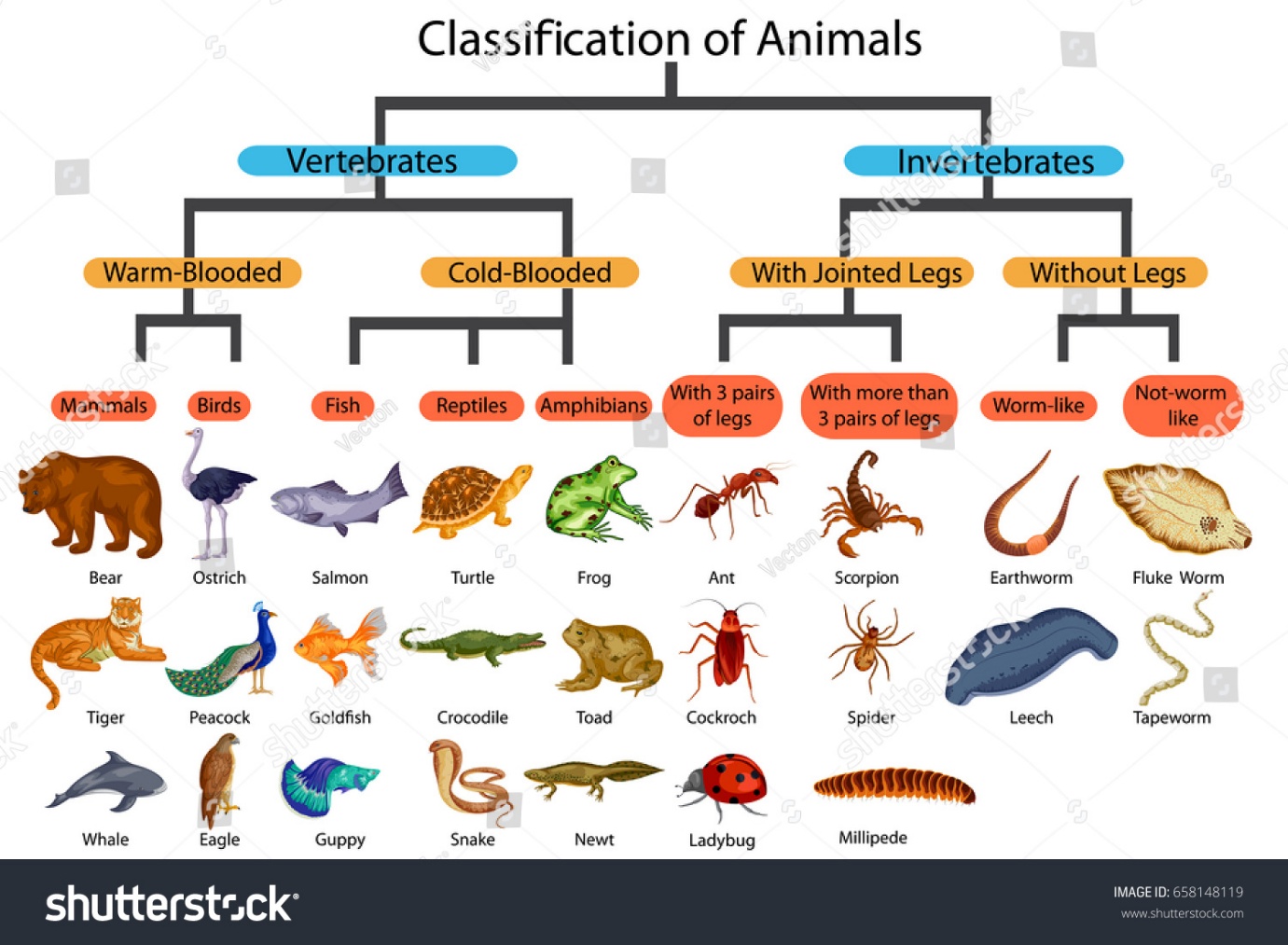
# What is Hierarchical Clustering?

**Hierarchical Clustering** builds clusters **step by step** — either by **merging** smaller clusters (bottom-up) or **splitting** larger ones (top-down).  
It creates a **hierarchy or tree structure** of clusters called a **dendrogram**.

Two main types:

1. **Agglomerative (bottom-up)** → Start with each point as its own cluster and keep merging.
2. **Divisive (top-down)** → Start with one cluster and split it recursively.





# What is DBSCAN?

**DBSCAN (Density-Based Spatial Clustering of Applications with Noise)** is an **unsupervised clustering algorithm** that groups together points that are **closely packed (dense regions)** and marks **outliers (noise)** as separate.

Unlike K-Means:

* It **does not need to know the number of clusters (k)** in advance.
* It can find **arbitrarily shaped clusters** (not just circular or spherical ones).
* It can **detect noise/outliers** automatically.

**2. Key Parameters**

1. **eps (ε)** – radius of the neighborhood.  
   → If the distance between two points is less than eps, they are considered neighbors.
2. **min\_samples** – minimum number of points required to form a dense region (a cluster).

**3. How DBSCAN Works Step-by-Step**

1. Pick an unvisited point.
2. Find all points within its eps radius → **neighbors**.
3. If neighbors ≥ min\_samples → it’s a **core point** → form a cluster.
4. Expand the cluster by including all reachable points (recursively).
5. If a point is not reachable from any cluster → it’s **noise**.

# CODE

# Step 1: Import libraries

import numpy as np

import matplotlib.pyplot as plt

from sklearn.datasets import make\_moons

from sklearn.cluster import DBSCAN

# Step 2: Create dataset

X, y = make\_moons(n\_samples=300, noise=0.05, random\_state=42)

# Step 3: Apply DBSCAN

dbscan = DBSCAN(eps=0.2, min\_samples=5)

y\_pred = dbscan.fit\_predict(X)

# Step 4: Visualize results

plt.figure(figsize=(8, 5))

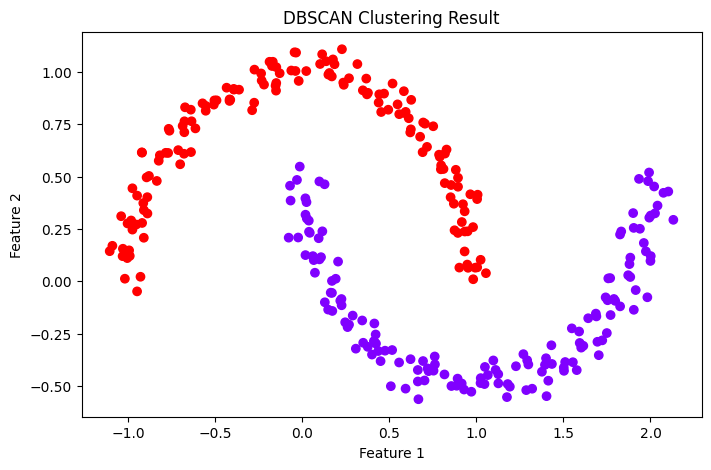
plt.scatter(X[:, 0], X[:, 1], c=y\_pred, cmap='rainbow')

plt.title("DBSCAN Clustering Result")

plt.xlabel("Feature 1")

plt.ylabel("Feature 2")

plt.show()



# DBSCAN vs K-Means: Key Differences

|  |  |  |
| --- | --- | --- |
| **Feature** | **DBSCAN** | **K-Means** |
| **Number of clusters** | Determined automatically | Must be given as k |
| **Cluster shape** | Can find irregular shapes | Works best with spherical clusters |
| **Noise/Outliers** | Detects and labels as -1 | Forces every point into a cluster |
| **Scalability** | Slower for large datasets | Faster on large datasets |
| **Parameter sensitivity** | Depends on eps and min\_samples | Depends on k and initialization |
| **Cluster density** | Works with variable densities | Fails if densities differ |