UNIT I

1. Define unsupervised learning. Explain any three applications of unsupervised learning in real-world domains.

Unsupervised learning: Machine learns patterns from data without labels (no right or wrong answer given).

Three applications of unsupervised learning

1. **Customer segmentation** → Grouping customers by shopping habits.
2. **Market basket analysis** → Finding which items are bought together.
3. **Anomaly detection** → Detecting fraud transactions or system failures.

2. Difference between clustering and association as an unsupervised machine learning task with suitable examples.

Aspect	Clustering	Association
Definition	Groups similar data points into clusters based on similarity or distance.	Finds relationships (rules) between variables/items in large datasets.
Goal	To discover natural groupings/patterns in data.	To identify strong if-then rules (associations) between items.
Output	Clusters (e.g., Cluster 1: young professionals, Cluster 2: students).	Association rules (e.g., *If customer buys bread → likely to buy butter*).
Techniques	K-Means, Hierarchical Clustering, DBSCAN, Gaussian Mixture Models.	Apriori Algorithm, FP-Growth, Eclat.

3. Differentiate between partitioning methods and hierarchical clustering.

• **Partitioning (like k-means):** Divide data into k groups directly. Needs number of clusters. Faster.
To choose the K, we use elbow method which uses within cluster sum of squares which defines the toal variation within a cluster for different clusters. Plot WCSS VS. K, the point at which it looks like an arm is considered the best point for K.

• Hierarchical**:** Build a tree (dendrogram) of clusters (merge or split step by step). Don't need the number of clusters. Slower. Computationally more expensive.

- Agglomerative: Agglomerative is a bottom-up approach, in which the algorithm starts with taking all data points as single clusters and merging them until one cluster is left.
- Divisive: Divisive algorithm is the reverse of the agglomerative algorithm as it is a top-down approach

Complete Linkage is farthest distance between the clusters data points .
Centroid Linkage is centroid distance between the clusters data points .

4. Explain density-based clustering. How does DBSCAN work?

- Clusters formed where data is dense.
- Points in sparse areas are noise.
- **DBSCAN (Density-based spatial clustering with noise) steps:**
  1. Pick a point → check neighbors within distance ε.
  2. If neighbors ≥ minPts → core point, form cluster.
  3. Expand cluster with connected core points.

5. Discuss advantages and limitations of clustering in machine learning.

☑ Advantages: Simple, finds hidden patterns, useful for large data.
✖ Limitations: Sensitive to noise, hard to choose k, struggles with high dimensions.

6. Explain the concept of biclustering.

Clusters **rows and columns** simultaneously. Fcous on local patterns across subset of features. Example: Clusters customers on some categories they buy.

7. What is spectral co-clustering? Give an application.

- Uses graph theory (eigenvalues of matrices).
- Application: Web search → group websites and users together.

8. What is an association rule? Provide an example.

• Rule of type: If X then Y. Example: {Milk} → {Bread}. Means people who buy milk often buy bread. Used in market based analysis. Algorithms are Apriori , Eclat, F-P Growth Algorithm

9. Define support, confidence, and lift in association rule mining.
   - **Support:** How often items appear together.
   - **Confidence:** How often Y appears when X occurs.
   - **Lift:** Strength of the rule compared to chance.

10. Explain the Apriori algorithm with its steps
   - Find frequent single items.
   - Form larger itemsets (pairs, triples).
   - Keep only those ≥ minimum support.
   - Generate rules with confidence ≥ threshold.

11. Discuss challenges in applying clustering to high-dimensional data.
   - Too many features → distance meaningless.
   - Curse of dimensionality.
   - Solution: Dimensionality reduction (PCA, t-SNE).

Numerical/Problem-Solving Questions

12. Apply k-means clustering (k=2) to the dataset: {(1,1), (2,1), (4,3), (5,4)}. Show steps.
   Choose initial centroids: C1 = (1,1), C2 = (5,4).
   Step 1 — assign points by distance
   - Distances to C1 and C2:
     - (1,1): to C1 = 0 → **C1**
     - (2,1): to C1 = 1, to C2 ≈ 4.243 → **C1**
     - (4,3): to C1 ≈ 3.606, to C2 ≈ 1.414 → **C2**
     - (5,4): to C1 = 5, to C2 = 0 → **C2**

   Clusters:
   - Cluster1 = {(1,1),(2,1)}
   - Cluster2 = {(4,3),(5,4)}

   Step 2 — update centroids
   - C1 = mean of Cluster1 = ((1+2)/2, (1+1)/2) = (1.5, 1)
   - C2 = mean of Cluster2 = ((4+5)/2, (3+4)/2) = (4.5, 3.5)

   Step 3 — reassign points
   Distances to new centroids keep the same cluster assignment (checked), so **converged**.
   Final answer:
   Cluster1 = {(1,1),(2,1)} with centroid (1.5,1)
   Cluster2 = {(4,3),(5,4)} with centroid (4.5,3.5)

13. Perform hierarchical clustering (single linkage) on data points: {1, 2, 6, 7, 9}.
   Single-linkage uses the **minimum** distance between clusters.
   Pairwise nearest neighbor distances (1D):
   - 1–2 = 1
   - 2–6 = 4
   - 6–7 = 1
   - 7–9 = 2

   Merging steps (ascending distances):
   1. Merge 1 & 2 at distance 1 → cluster {1,2}
   2. Merge 6 & 7 at distance 1 → cluster {6,7}
   3. Next smallest distance is 2: merge {6,7} with 9 → {6,7,9} at distance 2
   4. Finally merge {1,2} with {6,7,9} at distance = min distance between these clusters = 4 (between 2 and 6).

14. Using DBSCAN with parameters ε=2, minPts=2, classify the dataset: {1, 2, 2.5, 8, 9, 10}.
   For each point count neighbors within distance ≤ 2 (including itself).
   - For 1: neighbors {1,2,2.5} → count = 3 ≥ 2 → **core**
   - For 2: neighbors {1,2,2.5} → count = 3 → **core**
   - For 2.5: neighbors {1,2,2.5} → count = 3 → **core**

   These three are connected → **Cluster A = {1,2,2.5}**
   - For 8: neighbors {8,9,10} (10−8 = 2) → count = 3 → **core**
   - For 9: neighbors {8,9,10} → core
   - For 10: neighbors {8,9,10} → core

   These three form **Cluster B = {8,9,10}**

No noise points. Final clusters: {1,2,2.5} and {8,9,10}.

15. Calculate the support and confidence for the rule {Milk} → {Bread} using a transaction dataset.
    Total transactions = 4
    Transactions containing Milk = T1, T2, T4 → 3
    Transactions containing Milk and Bread together = T1, T2 → 2
    Support({Milk} → {Bread}) = occurrences (Milk ∪ Bread) / total = 2 / 4 = **0.5 (50%)**
    Confidence({Milk} → {Bread}) = support(Milk ∪ Bread) / support(Milk) = (2/4) / (3/4) = 2/3 = **0.6667 (66.67%)**

16. Given transactions: T1 = {Milk, Bread, Butter}
    T2 = {Milk, Bread}
    T3 = {Bread, Butter}
    T4 = {Milk, Butter}
    Find frequent itemsets with minimum support = 50%.
    Transactions:
    - T1 = {Milk, Bread, Butter}
    - T2 = {Milk, Bread}
    - T3 = {Bread, Butter}
    - T4 = {Milk, Butter}

    Total transactions = 4 → 50% means at least 2 transactions.
    Singletons (counts):
    - Milk: in T1,T2,T4 → 3 → 75% → **frequent**
    - Bread: in T1,T2,T3 → 3 → 75% → **frequent**
    - Butter: in T1,T3,T4 → 3 → 75% → **frequent**

    Pairs:
    - {Milk, Bread}: T1,T2 → 2 → 50% → **frequent**
    - {Milk, Butter}: T1,T4 → 2 → 50% → **frequent**
    - {Bread, Butter}: T1,T3 → 2 → 50% → **frequent**

    Triple:
    - {Milk,Bread,Butter}: only T1 → 1 → 25% → **not frequent**

    Final frequent itemsets (≥50%):
    All single items and all pairs listed above.

17. Apply Apriori algorithm step by step to generate association rules from the above dataset.
    Step 1 — L1 (frequent 1-itemsets): {Milk}, {Bread}, {Butter} (all 75%)
    Step 2 — Candidate 2-itemsets (C2): {Milk,Bread}, {Milk,Butter}, {Bread,Butter}
    All have support 50% → become L2.
    Step 3 — Candidate 3-itemsets (C3): {Milk,Bread,Butter} — support 25% → not frequent.
    Generate association rules from frequent itemsets (we can compute confidence for each pair):
    Example rules (and confidences computed earlier):
    - {Milk} → {Bread}: confidence = 2/3 = 66.67%
    - {Bread} → {Milk}: confidence = 2/3 = 66.67%
    - {Milk} → {Butter}: 2/3
    - {Butter} → {Milk}: 2/3
    - {Bread} → {Butter}: 2/3
    - {Butter} → {Bread}: 2/3

    (If you had a confidence threshold, pick only rules above it. Here each pair rule has 66.67% confidence.)
18. Use hierarchical clustering (complete linkage) to cluster {(1,1), (1.5,1.5), (5,5), (3,4)}.
    Pairwise distances (approx):
    - d(1,1)-(1.5,1.5) = $\sqrt{(0.5^2+0.5^2)} \approx 0.7071$
    - d(1,1)-(3,4) = $\sqrt{(2^2+3^2)} = \sqrt{13} \approx 3.6056$
    - d(1,1)-(5,5) = $\sqrt{(4^2+4^2)} = \sqrt{32} \approx 5.657$
    - d(1.5,1.5)-(3,4) = $\sqrt{(1.5^2+2.5^2)} = \sqrt{8.5} \approx 2.9155$
    - d(1.5,1.5)-(5,5) = $\sqrt{(3.5^2+3.5^2)} = \sqrt{24.5} \approx 4.9497$
    - d(3,4)-(5,5) = $\sqrt{(2^2+1^2)} = \sqrt{5} \approx 2.2361$

    Complete-linkage merges (use maximum distance between points of clusters):
    1. Merge (1,1) & (1.5,1.5) first (distance 0.7071) → cluster AB.
    2. Merge (5,5) & (3,4) next (distance 2.2361) → cluster CD.
    3. Distance between AB and CD under complete linkage = **maximum** distance among pairs across clusters = max{5.657, 3.6056, 4.9497, 2.9155} = **5.657**. So final merge at distance ≈ 5.657.

Clusters at intermediate steps:

- After first: { (1,1),(1.5,1.5) } and { (3,4) } and { (5,5) }
- After second: { (1,1),(1.5,1.5) } and { (3,4),(5,5) }
- Final merge at 5.657 gives one cluster.

19. Derive the time complexity of the Apriori algorithm.

- Apriori generates candidate k-itemsets from frequent (k−1)-itemsets and scans the database to count support.
- **Worst-case complexity is exponential** in the number of distinct items m: there can be up to 2^mitemsets.
- More practical expression: scanning cost ≈ $O(N \times C)$where N= number of transactions and C= number of candidates generated (which can be up to $O(2^m)$).
- **Summary:** Worst-case time ≈ **O(N · 2^m)** (exponential in item count), but actual cost is lower when many itemsets are pruned by support.

20. A dataset has 10,000 transactions. Rule {A} → {B} occurs 500 times, support(A)=0.1. Compute confidence and lift.

Total transactions = 10,000.
support(A) = 0.1 ⇒ A appears in $0.1 \times 10000 = 1000$transactions.
support(A ∪ B) = occurrences of rule / total = 500 / 10000 = 0.05.

Confidence({A} → {B}) = support(A ∪ B) / support(A) = 0.05 / 0.1 = **0.5 (50%)**

21. Demonstrate how biclustering can be applied in gene expression data.

- **Data layout:** rows = genes, columns = experimental conditions (e.g., time points). Entry = expression level.
- **Goal of biclustering:** find a subset of genes that behave similarly *only* across a subset of conditions (rows + columns together).
- **Example:** Genes G1, G2, G5 show high expression for conditions C2 and C4 but not others → bicluster {G1,G2,G5} × {C2,C4}.
- **Why useful:** genes may co-express only under specific conditions (disease state, treatment), which standard clustering (rows only) can miss.
- **Method outline:** pick a biclustering algorithm (e.g., Cheng & Church, spectral biclustering), compute biclusters, validate biologically (pathways, enrichment).

22. Design a real-world case study of clustering for customer segmentation.

Business: E-commerce store wants target marketing.
Data/features: recency (days since last purchase), frequency (purchases/month), monetary (avg spend), product categories liked, location, device type.
Steps:

1. **Preprocess:** clean data, normalize numeric features, encode categorical ones.
2. **Choose algorithm:** K-means or GMM for soft clusters; DBSCAN to find dense buyer groups & outliers.
3. **Run clustering (example k = 4):** you might get segments like:
   - High-value frequent buyers (target VIP offers)
   - Occasional big spenders (target with coupons)
   - Bargain hunters (promote deals)
   - New / cold leads (onboarding emails)
4. **Validate:** silhouette score, business-metric lift (conversion after targeted campaign).
5. **Action:** personalize emails, discounts, and product recommendations per cluster.

Outcome: better personalization → higher conversion & retention.

UNIT II

1. Define Gaussian Mixture Models (GMM).

A **Gaussian Mixture Model (GMM)** is a **probabilistic model** that assumes data is generated from a combination of several Gaussian (normal) distributions.
Each Gaussian represents a **cluster**, and every data point belongs to each cluster with a **certain probability**.
👉 Example: Grouping customers where each group's spending pattern follows a different Gaussian curve
Advantages of GMM over hard clustering

✅ Soft assignment — a point can belong to multiple clusters with probabilities.
✅ Can handle overlapping clusters.
✅ Works well with **elliptical** and **non-spherical** data.
✅ Provides a probabilistic framework for further statistical analysis.

2. Differentiate between Gaussian mixture and k-means clustering.

Feature	K-Means	Gaussian Mixture Model
Type	Hard clustering	Soft (probabilistic) clustering
Membership	Each point belongs to one cluster only	Each point has probability of belonging to multiple clusters
Shape of clusters	Spherical	Elliptical (can vary in shape/size)
Method	Minimizes distance to centroid	Maximizes likelihood under Gaussian mixture
Output	Centroids	Mean, variance, and weight for each Gaussian

3. Explain Variational Bayesian Gaussian mixture.
It is an **advanced version of GMM** that uses **Bayesian inference** to estimate parameters.
- Instead of fixing the number of clusters (k), it **automatically determines** how many are needed.
- It assumes priors on parameters and finds distributions over them using **variational methods** (approximation of probabilities).

👉 **Benefit:** Avoids overfitting and unnecessary clusters.

4. Define manifold learning with examples.
Manifold learning is a technique for **non-linear dimensionality reduction**, assuming data lies on a curved surface (manifold) within a higher-dimensional space.
👉 **Examples:**
- **Isomap** – preserves geodesic (curved) distances.
- **LLE (Locally Linear Embedding)** – preserves relationships among nearest neighbors.
- **t-SNE** – preserves local neighborhoods for visualization.

5. Explain the working of Isomap algorithm.
Isomap (Isometric Mapping) steps:
   1. **Compute nearest neighbors** for each data point.
   2. **Build a graph** connecting neighbors with edge distances.
   3. **Compute shortest paths (geodesic distances)** between all points using Dijkstra's or Floyd's algorithm.
   4. **Apply MDS (Multi-Dimensional Scaling)** on these geodesic distances to get low-dimensional coordinates.

☑ **Use:** Nonlinear data visualization like facial pose, motion, or image manifolds.

6. Discuss Locally Linear Embedding (LLE).
LLE finds a low-dimensional embedding that preserves **local neighborhood geometry**.
Steps:
   1. Find *k* nearest neighbors for each point.
   2. Compute weights to reconstruct each point from its neighbors.
   3. Find low-dimensional coordinates that preserve these weights.

☑ **Use:** Helpful in image, face, and text data where local relationships matter more than global ones.

7. What is Modified Locally Linear Embedding (MLLE)?
MLLE is an improved version of LLE that:
   - Reduces numerical instability.
   - Handles non-uniform sampling and noisy data better.
   - Uses **multiple weight vectors** instead of one for each point → captures structure more accurately.

☑ **Result:** More robust and stable embeddings.

8. Explain Spectral Embedding in manifold learning.
Spectral Embedding uses the **eigenvalues and eigenvectors** of a graph Laplacian (built from data points) to find low-dimensional structure.
Steps:
   1. Create a similarity graph between data points.
   2. Compute the Laplacian matrix.

3. Use the smallest eigenvectors to embed the data.

☑ **Use:** Nonlinear dimensionality reduction and clustering (basis of Spectral Clustering).

9. Define Multi-Dimensional Scaling (MDS).
   MDS converts pairwise distances between data points into a geometric map in fewer dimensions.
   Goal: Preserve distances as much as possible.
   👉 **Example:** If two customers are similar, they'll appear close together in the MDS plot.

10. Explain t-distributed Stochastic Neighbor Embedding (t-SNE).
    t-SNE is a powerful nonlinear dimensionality reduction and visualization technique.
    Steps:
    1. Compute probabilities of similarity between points in high dimension.
    2. Map them into a low-dimensional space (2D or 3D).
    3. Minimize difference between original and mapped similarities using a **Student-t distribution** (which prevents crowding).

    ☑ **Use:** Visualizing clusters in image, text, and biological data.

11. Compare PCA and t-SNE for dimensionality reduction.

Feature	PCA	t-SNE
Type	Linear	Nonlinear
Preserves	Global variance	Local neighborhood
Output	Directions (principal components)	2D/3D map for visualization
Speed	Fast	Slower
Use	Feature reduction	Data visualization

12. What is factor analysis? How is it different from PCA?
    Factor Analysis finds **latent (hidden) variables (factors)** that explain correlations among observed variables.

Aspect	PCA	Factor Analysis
Goal	Maximize total variance	Explain shared variance (common factors)
Assumes	No noise model	Explicit noise model
Type	Mathematical	Statistical (probabilistic)

☑ Example: Finding underlying factors like "intelligence" from correlated test scores.

13. Explain Kernel PCA and its advantage over standard PCA.

    Kernel PCA applies PCA in a **higher-dimensional space** using a **kernel function** (like RBF, polynomial) without explicitly computing that space (via the "kernel trick").
    ✅ **Advantage:** Captures **non-linear relationships** that standard (linear) PCA cannot.
    👉 Example: Separating data shaped like concentric circles — linear PCA fails, but kernel PCA succeeds.

14. Discuss Latent Dirichlet Allocation (LDA) with applications.
    LDA (Latent Dirichlet Allocation) is a **topic modeling algorithm** for text data.
    - Each **document** is a mixture of topics.
    - Each **topic** is a mixture of words.
    - Uses probabilistic modeling to discover hidden themes in large text collections.

    ☑ **Applications:**
    - Discovering topics in news articles or research papers
    - Recommendation systems
    - Organizing documents or social media posts by theme

Numerical/Problem-Solving Questions

15. Given data {1, 2, 5, 6}, fit a Gaussian mixture with 2 components. Show steps.
16. For a dataset {(2,3), (3,4), (4,5), (8,7)}, perform PCA and find first principal component.
17. Compute eigenvalues and eigenvectors of covariance matrix [[2,1],[1,2]] for PCA.
18. Perform truncated SVD on matrix [[1,0],[0,1],[1,1]].
19. Apply MDS on points {(0,0), (1,0), (0,1)} using Euclidean distance.
20. Use Isomap to reduce dataset {(0,0), (1,1), (2,2), (3,3)} to 1D.
21. Given a 3×3 covariance matrix, perform factor analysis (extract 2 factors).
22. Apply LLE on dataset {(0,0), (1,0), (0,1)} with k=2 neighbors.
23. Perform independent component analysis (ICA) on mixed signals X1 = S1 + S2, X2 = S1 - S2.
24. Decompose matrix [[4,2],[2,3]] using Non-negative Matrix Factorization (NMF).
25. A dataset has 1000 documents. Apply Latent Semantic Analysis (LSA) using SVD to extract 2 topics.
26. Perform kPCA using RBF kernel on data {(0,0), (1,0), (0,1)}.
27. Given covariance matrix [[3,1],[1,3]], reduce dimension using PCA to 1D.
28. Compute perplexity in t-SNE for dataset of 4 points with probabilities given.
29. Case study: Show how GMM can be used for speaker recognition.