BI – Unit 5 (Impact of Machine Learning in Business Intelligence Process) – END-SEM PYQ Answers

MAY-JUNE 2023

Q5a) What is Association Rule Mining? Explain Support, Confidence, and Lift. [5 marks]

Association rule mining is an unsupervised data mining technique used to discover interesting relationships, co-occurrences, and patterns among items in large transactional datasets. The classic application is market basket analysis — identifying which products are frequently purchased together so that supermarkets can design better store layouts, promotions, and recommendation systems.

An association rule takes the form: Antecedent → Consequent, e.g., {Bread, Butter} → {Milk}. This means: 'Customers who buy Bread and Butter also tend to buy Milk.'

1. Support

Support measures how frequently the itemset appears in the dataset. It tells you whether a pattern is statistically significant or just a rare coincidence.

Support(X → Y) = Count(transactions containing both X and Y) / Total transactions

Example: If {Bread, Milk} appears in 300 out of 1000 transactions, Support = 300/1000 = 30%. A minimum support threshold (e.g., 20%) filters out infrequent, irrelevant patterns. Only itemsets meeting this threshold are called 'frequent itemsets'.

2. Confidence

Confidence measures how often the rule is correct — given that a customer bought X, what is the probability they also bought Y?

Confidence(X → Y) = Support(X ∪ Y) / Support(X)

Example: If {Bread} appears in 500 transactions and {Bread, Milk} appears in 300, then Confidence({Bread} → {Milk}) = 300/500 = 60%. This means 60% of the time someone buys Bread, they also buy Milk. A minimum confidence threshold (e.g., 70%) ensures only strong rules are retained.

3. Lift

Lift measures whether the rule is genuinely informative or simply a result of the consequent being universally popular. It corrects for the base rate popularity of the consequent.

Lift(X → Y) = Confidence(X → Y) / Support(Y)

Interpretation: Lift = 1 means X and Y are independent (the rule is not useful). Lift > 1 means X and Y co-occur more often than chance (positive association — the rule is useful). Lift < 1 means purchasing X actually reduces the likelihood of Y (negative association).

Example: If Confidence = 60% but Support(Milk) = 70% (milk is bought in 70% of all transactions anyway), then Lift = 0.60/0.70 = 0.857 — meaning buying Bread does not actually make Milk purchase more likely than random. Lift < 1 makes this a poor rule despite the decent confidence.

Q5b) Difference between Hierarchical Clustering and Partitioning Method. [5 marks]

Aspect	Partitioning Methods (e.g., K-Means)	Hierarchical Methods
Number of clusters	Must specify K upfront	Determined from the

Aspect	Partitioning Methods (e.g., K-Means)	Hierarchical Methods
		dendrogram after the fact
Process	Iterative — assign points, update centroids, repeat until stable	Sequential — successively merge or split clusters
Output	A flat partition: K non-overlapping clusters	A tree (dendrogram) showing all possible cluster levels
Flexibility	Fixed K; must re-run for different K	Cut dendrogram at any level to get any number of clusters
Scalability	Efficient on large datasets (O(nKt))	Computationally expensive (O(n² log n) or worse)
Revision	Allows reassignment of points between clusters	No revision — once merged/split, cannot be undone
Cluster shape	Best for spherical, similarly sized clusters	Can handle various shapes and sizes
Example algorithms	K-Means, K-Medoids	Agglomerative (bottom-up), Divisive (top-down)

Q5c) Apriori Algorithm — Find Frequent Itemsets (min_support = 2, min_confidence = 60%). [8 marks]

The transaction database is:

TID	Items
T1	I1, I2, I5
T2	I2, I4
T3	I2, I3
T4	I1, I2, I4
T5	I1, I3
T6	I2, I3
T7	I1, I3
T8	I1, I2, I3, I5
T9	I1, I2, I3

Step 1: Generate C1 (1-item candidate sets) and L1 (frequent 1-itemsets)

Count each item across all 9 transactions:

Itemset	Count	Frequent? (min_support = 2)
{I1}	6	Yes
{I2}	7	Yes
{I3}	6	Yes
{I4}	2	Yes
{I5}	2	Yes

L1 = {{I1}, {I2}, {I3}, {I4}, {I5}} — all 5 items are frequent.

Step 2: Generate C2 (2-item candidates) and L2

Generate all 2-item combinations from L1 and count:

Itemset	Count	Frequent?
{I1, I2}	4	Yes
{I1, I3}	4	Yes
{I1, I4}	1	No
{I1, I5}	2	Yes
{I2, I3}	4	Yes
{I2, I4}	2	Yes
{I2, I5}	2	Yes
{I3, I4}	0	No
{I3, I5}	1	No
{I4, I5}	0	No

L2 = {{I1,I2}, {I1,I3}, {I1,I5}, {I2,I3}, {I2,I4}, {I2,I5}}

Step 3: Generate C3 and L3

Join L2 with itself to form 3-item candidates, then prune those with infrequent 2-item subsets:

Itemset	Count	Frequent?
{I1, I2, I3}	2	Yes
{I1, I2, I5}	2	Yes
{I1, I3, I5}	1	No
{I2, I3, I4}	0	No (I3,I4 infrequent — pruned)

L3 = {{I1,I2,I3}, {I1,I2,I5}}

No 4-item candidates can be generated from L3 (not enough qualifying 3-item sets). Process stops.

Step 4: Generate Association Rules (min_confidence = 60%)

For each frequent itemset, generate all possible rules and check confidence. Showing rules for the most informative itemsets:

Rule	Confidence Calculation	Confidence	Strong?
I1 ∧ I2 → I3	count({I1,I2,I3}) / count({I1,I2}) = 2/4	50%	No
I1 ∧ I3 → I2	count({I1,I2,I3}) / count({I1,I3}) = 2/4	50%	No
I2 ∧ I3 → I1	count({I1,I2,I3}) / count({I2,I3}) = 2/4	50%	No
I1 → I2 ∧ I3	count({I1,I2,I3}) / count({I1}) = 2/6	33%	No
I1 ∧ I2 → I5	count({I1,I2,I5}) / count({I1,I2}) = 2/4	50%	No
I1 ∧ I5 → I2	count({I1,I2,I5}) / count({I1,I5}) = 2/2	100%	Yes
I2 ∧ I5 → I1	count({I1,I2,I5}) / count({I2,I5}) = 2/2	100%	Yes
I5 → I1 ∧ I2	count({I1,I2,I5}) / count({I5}) = 2/2	100%	Yes
I2 → I4	count({I2,I4}) / count({I2}) = 2/7	29%	No
I4 → I2	count({I2,I4}) / count({I4}) = 2/2	100%	Yes
I1 → I5	count({I1,I5}) / count({I1}) = 2/6	33%	No
I5 → I1	count({I1,I5}) / count({I5}) = 2/2	100%	Yes

Strong Association Rules (confidence >= 60%): {I1,I5}→{I2} (100%), {I2,I5}→{I1} (100%), {I5}→{I1,I2} (100%), {I4}→{I2} (100%), {I5}→{I1} (100%).

> **Note:** *The Apriori principle states: if an itemset is infrequent, all its supersets are also infrequent. This allows early pruning of candidates, which is what makes Apriori efficient for large datasets.*

Q6a) Explain Bayes Theorem in Detail. [5 marks]

Bayes' Theorem is a fundamental result in probability theory that describes how to update our belief about a hypothesis when we receive new evidence. It forms the mathematical foundation of Naive Bayes classifiers — one of the most widely used and efficient classification algorithms in text classification and spam filtering.

Bayes' Theorem: P(H|E) = [P(E|H) × P(H)] / P(E)

- P(H|E) — Posterior probability: The probability that hypothesis H is true given that we have observed evidence E. This is what we want to calculate.
- P(H) — Prior probability: Our initial belief in hypothesis H before observing any evidence. Example: 10% of emails are spam, so P(spam) = 0.10.
- P(E|H) — Likelihood: The probability of observing evidence E if hypothesis H were true. Example: If an email is spam, the probability it contains the word 'win' is 0.80.
- P(E) — Evidence probability: The overall probability of observing evidence E across all hypotheses. Acts as a normalizing constant.

Example — Email Spam Detection: Suppose we receive an email containing the word 'win'. We want to know if it is spam. Given: P(spam) = 0.10, P(not spam) = 0.90, P('win'|spam) = 0.80, P('win'|not spam) = 0.05.

P(E) = P('win'|spam)×P(spam) + P('win'|not spam)×P(not spam) = 0.80×0.10 + 0.05×0.90 = 0.08 + 0.045 = 0.125

P(spam|'win') = (0.80 × 0.10) / 0.125 = 0.08/0.125 = 0.64 = 64%.

So despite the fact that only 10% of emails are spam (our prior), seeing the word 'win' updates our belief to 64% that this particular email is spam. Bayes' Theorem provides a principled, mathematically sound way to combine prior knowledge with new evidence.

> **Note:** *Naive Bayes extends this to multiple features by assuming conditional independence between features — 'naive' because this assumption is often violated in reality, yet the classifier performs surprisingly well in many text classification tasks.*

Q6b) Difference between Classification and Clustering. [5 marks]

Aspect	Classification	Clustering
Learning type	Supervised — uses labeled training data	Unsupervised — no labels required
Goal	Predict the class label of unseen instances	Discover natural groupings in unlabeled data
Output	Predefined classes (known categories)	Clusters (discovered categories)
Training	Learns from past labeled examples	No training labels — finds structure in raw data
Example algorithms	Decision Trees, SVM, Logistic Regression, Naive Bayes	K-Means, DBSCAN, Hierarchical Clustering
Evaluation	Accuracy, Precision, Recall, F1 Score	Silhouette Score, Davies-Bouldin Index
Use case example	Email spam/not-spam prediction	Customer segmentation for marketing
When to use	When historical labeled data is available	When exploring unknown patterns in new data

Q6c) Explain Logistic Regression with Example. [8 marks]

Logistic regression is a supervised classification algorithm used to predict the probability that a given input belongs to a particular class. Despite the word 'regression' in its name, it is a classification method — the name comes from the logistic (sigmoid) function it uses to bound output probabilities between 0 and 1.

The Core Idea

Linear regression predicts a continuous output: y = w0 + w1×x1 + w2×x2 + ... Linear outputs can range from -∞ to +∞, which is unsuitable for predicting probabilities that must lie in [0,1]. Logistic regression solves this by passing the linear combination through the sigmoid function.

Sigmoid function: σ(z) = 1 / (1 + e^(-z)), where z = w0 + w1×x1 + w2×x2 + ...

The sigmoid function maps any real number to the range (0,1). We interpret the output as: P(Y=1|X) = σ(z). If this probability exceeds 0.5, we classify the instance as class 1; otherwise as class 0.

Types of Logistic Regression

- Binary Logistic Regression: Two output classes (0 or 1, Yes or No, Spam or Not Spam). Uses a single sigmoid output unit. Example: Predicting whether a loan application will default (Yes/No).
- Multinomial Logistic Regression: Three or more classes with no natural ordering. Uses a softmax function instead of sigmoid to produce probabilities across all classes. Example: Classifying a news article into Sports, Politics, Business, or Technology.
- Ordinal Logistic Regression: Three or more ordered classes. Example: Predicting customer satisfaction as Low, Medium, or High.

Worked Example — Predicting Exam Pass/Fail

Suppose we want to predict whether a student passes (1) or fails (0) based on hours studied. After training, suppose the model learns: z = -4 + 1.5 × (Hours_Studied). For a student who studies 3 hours: z = -4 + 1.5×3 = -4 + 4.5 = 0.5. Then σ(0.5) = 1/(1+e^(-0.5)) ≈ 1/(1+0.607) ≈ 0.622. Since 0.622 > 0.5, the student is predicted to Pass.

For a student who studies 1 hour: z = -4 + 1.5×1 = -2.5. σ(-2.5) ≈ 0.076 — only 7.6% chance of passing, so predicted to Fail.

Model Training

Logistic regression is trained by minimizing the log-loss (cross-entropy loss) function using gradient descent or a similar optimization algorithm. The log-loss penalizes confident wrong predictions more heavily than uncertain ones, incentivizing calibrated probability estimates.

> **Note:** *Decision boundary: The model learns a linear boundary in the feature space that separates the two classes. Everything on one side is classified as class 1, everything on the other as class 0. For non-linearly separable data, polynomial features can be added.*

NOV-DEC 2023

Q5a) [REPEATED] Difference between Classification and Clustering with applications. [6 marks]

Q5b) [REPEATED] Short note on Logistic Regression. [6 marks]

Q5c) Apriori Algorithm — (min_support=2, min_confidence=70%). [6 marks]

The transaction database (same as given in the question):

TID	Items
T100 (1)	I1, I2, I5
T100 (2)	I2, I4
T100 (3)	I2, I3
T100 (4)	I1, I2, I4
T100 (5)	I1, I3
T100 (6)	I2, I3
T100 (7)	I1, I3
T100 (8)	I1, I2, I3, I5
T100 (9)	I1, I2, I3

This is the same dataset as May-June 2023, Q5c. The frequent itemsets are identical (L1, L2, L3 computed above). The only difference is the confidence threshold is now 70% instead of 60%.

Strong Rules at min_confidence = 70%

From the rules computed earlier, those with confidence >= 70% are:

- {I1,I5} → {I2} — Confidence 100% (Strong)
- {I2,I5} → {I1} — Confidence 100% (Strong)
- {I5} → {I1,I2} — Confidence 100% (Strong)
- {I4} → {I2} — Confidence 100% (Strong)
- {I5} → {I1} — Confidence 100% (Strong)

Rules with confidence between 60% and 70% (which were strong at 60% but not at 70%) are now excluded. In this dataset, all the strong rules at 60% already had 100% confidence, so the result is the same.

Q6a) What are Association Rules? How to evaluate using Support and Confidence? [6 marks]

Association rules are IF-THEN statements that describe co-occurrence patterns among items in a transactional dataset. They take the form Antecedent → Consequent and express the observation that whenever the antecedent set of items appears in a transaction, the consequent also tends to appear.

A classic example: {Diapers} → {Beer}. This famously discovered rule in retail data showed that young fathers (buying diapers) also frequently purchased beer, prompting stores to place beer near the baby aisle.

Evaluation metrics for association rules — Support and Confidence are defined in full under May-June 2023, Q5a above. To summarize: Support filters out statistically insignificant (too rare) patterns, while Confidence filters out weak or unreliable rules. Both minimum thresholds must be met for a rule to be considered useful.

Q6b) State different formulae for Evaluation of Classification Models. [6 marks]

After a classifier is trained, its performance must be rigorously evaluated on unseen test data. The evaluation starts with the Confusion Matrix, which tabulates the four possible outcomes of a binary classifier:

	Predicted Positive	Predicted Negative
Actual Positive	TP (True Positive)	FN (False Negative)
Actual Negative	FP (False Positive)	TN (True Negative)

The following metrics are derived from TP, FP, TN, FN:

- Accuracy = (TP + TN) / (TP + TN + FP + FN) — Overall proportion of correct predictions. Can be misleading for imbalanced classes (e.g., 99% of data is class 0 — a classifier that always predicts 0 gets 99% accuracy but detects no positives).
- Precision = TP / (TP + FP) — Of all instances predicted positive, what fraction were actually positive? High precision means few false alarms. Use when the cost of a false positive is high (e.g., innocent person flagged as criminal).
- Recall (Sensitivity) = TP / (TP + FN) — Of all actual positives, what fraction did we correctly identify? High recall means few missed cases. Use when the cost of missing a positive is high (e.g., failing to diagnose a disease).
- Specificity = TN / (TN + FP) — Of all actual negatives, what fraction did we correctly identify as negative?
- F1 Score = 2 × (Precision × Recall) / (Precision + Recall) — Harmonic mean of precision and recall. Provides a single balanced metric, especially useful for imbalanced datasets.
- ROC-AUC — Plots TPR (Recall) vs. FPR (1-Specificity) at various classification thresholds. AUC (Area Under Curve) = 1.0 is a perfect classifier; AUC = 0.5 is random guessing.

Q6c) K-Means Clustering — Website Visitors Clustering (K=2). [6 marks]

Dataset: Age values of 19 visitors: 16, 16, 17, 20, 20, 21, 21, 22, 23, 29, 36, 41, 42, 43, 44, 45, 61, 62, 66.

Step 1: Initialize Centroids

Choose K=2 random initial centroids. A common approach is to pick the first and last value (or the most spread-out points). Let: C1 = 16, C2 = 66.

Step 2: Assign Points to Nearest Centroid

For each data point, calculate the distance to each centroid and assign to the nearer one.

Midpoint between C1=16 and C2=66 is (16+66)/2 = 41. Points ≤ 41 → Cluster 1, points > 41 → Cluster 2.

- Cluster 1 (<= 41): 16, 16, 17, 20, 20, 21, 21, 22, 23, 29, 36, 41
- Cluster 2 (> 41): 42, 43, 44, 45, 61, 62, 66

Step 3: Recalculate Centroids

New C1 = Mean of Cluster 1 = (16+16+17+20+20+21+21+22+23+29+36+41)/12 = 282/12 = 23.5

New C2 = Mean of Cluster 2 = (42+43+44+45+61+62+66)/7 = 363/7 = 51.86

Step 4: Reassign with Updated Centroids

Midpoint = (23.5+51.86)/2 = 37.68. Points <= 37 → Cluster 1, points > 37 → Cluster 2.

- Cluster 1: 16, 16, 17, 20, 20, 21, 21, 22, 23, 29, 36 (removed 41 which is > 37.68... wait: 41 > 37.68 so it moves)
- Cluster 1 (revised): 16, 16, 17, 20, 20, 21, 21, 22, 23, 29, 36 (11 points)
- Cluster 2 (revised): 41, 42, 43, 44, 45, 61, 62, 66 (8 points)

Step 5: Recalculate Again

New C1 = (16+16+17+20+20+21+21+22+23+29+36)/11 = 241/11 = 21.9

New C2 = (41+42+43+44+45+61+62+66)/8 = 404/8 = 50.5

Midpoint = (21.9+50.5)/2 = 36.2. Check assignments: All points <= 36 → C1, points > 36 → C2. This gives the same assignment as before — convergence achieved.

Final Result: Cluster 1 (Young: mean ~22) = {16,16,17,20,20,21,21,22,23,29,36}. Cluster 2 (Older: mean ~50) = {41,42,43,44,45,61,62,66}.

MAY-JUNE 2024

Q5a) [REPEATED] What is Logistic Regression? Discuss the types. [6 marks]

Q5b) [REPEATED] How are Classification and Clustering different? Discuss use with example. [6 marks]

Q5c) What is a Decision Tree? Explain with a Case Study. [6 marks]

A decision tree is a supervised classification and regression algorithm that learns a flowchart-like tree structure from labeled training data. Each internal node represents a test on a feature attribute, each branch represents an outcome of that test, and each leaf node represents a class label or predicted value. The model makes predictions by traversing from the root node to a leaf by following the path of applicable conditions.

How a Decision Tree is Built

The tree is built top-down using a greedy algorithm. At each node, the algorithm selects the feature that best splits the data according to a purity criterion:

- Gini Impurity: Measures the probability that a randomly selected sample would be misclassified if randomly labeled according to the class distribution in the node. Lower Gini = purer node.
- Information Gain (Entropy-based): Chooses the feature that maximally reduces uncertainty (entropy) after the split. Used in the ID3 and C4.5 algorithms.
- Splitting continues until all leaf nodes are pure (contain only one class) or a stopping criterion is met (max depth, minimum samples per node).

Case Study — Loan Default Prediction

Suppose a bank has historical data of loan applicants with features: Credit Score (Good/Bad), Income Level (High/Low), Employment Status (Employed/Unemployed), and the target: Default (Yes/No).

The decision tree might learn the following structure: Root: Is Credit Score Good? → If No → Default = Yes (95% of bad credit borrowers defaulted). → If Yes → Is Income Level High? → If Yes → Default =

No (5% defaulted). → If No → Is Employment Status Employed? → If Yes → Default = No (15% defaulted). → If No → Default = Yes (70% defaulted).

A new applicant with Good Credit, Low Income, and Unemployed status would follow the path: Good Credit → Low Income → Unemployed → Predicted: Default = Yes.

Advantages and Disadvantages

- Advantages: Highly interpretable (the tree can be visualized and explained to non-technical stakeholders), handles both numerical and categorical features, requires no feature scaling.
- Disadvantages: Prone to overfitting if grown too deep (remedied by pruning or using Random Forests), unstable (small changes in data can produce a very different tree), not ideal for continuous smooth decision boundaries.

Note: *Random Forest (ensemble of many decision trees) resolves overfitting by averaging predictions across hundreds of trees grown on different random subsets of the data and features.*

Q6a) What is K-Means Clustering? Explain step-by-step working. [6 marks]

K-Means is a partitioning clustering algorithm that divides a dataset into K non-overlapping clusters by iteratively assigning each data point to its nearest cluster centroid and then updating the centroids. The algorithm minimizes the Within-Cluster Sum of Squares (WCSS) — the total squared distance between each point and its assigned centroid.

Step-by-Step Algorithm

- Step 1 — Choose K: Specify the desired number of clusters K. This is a user-defined hyperparameter. The Elbow Method can help select an appropriate K by plotting WCSS against different values of K and identifying the 'elbow point'.
- Step 2 — Initialize Centroids: Randomly select K data points from the dataset as initial centroids (or use K-Means++ initialization for better convergence).
- Step 3 — Assign Points to Nearest Centroid: For each data point, calculate the Euclidean distance to all K centroids and assign the point to the cluster of the nearest centroid. Distance = sqrt((x1-c1)² + (x2-c2)² + ...).
- Step 4 — Update Centroids: For each cluster, recalculate the centroid as the mean of all data points assigned to that cluster: New centroid = mean of all points in the cluster.
- Step 5 — Repeat: Repeat Steps 3 and 4 until convergence — i.e., until no data point changes its cluster assignment (centroids stabilize).
- Step 6 — Output: The final K clusters with their centroids are the result.

Limitations

- Requires K to be specified in advance — wrong K gives meaningless clusters.
- Sensitive to initial centroid placement — different runs may converge to different local minima. K-Means++ initialization mitigates this.
- Assumes spherical, similarly sized clusters — performs poorly on elongated or irregularly shaped clusters.
- Sensitive to outliers — a single extreme outlier can distort a centroid significantly. K-Medoids is a more robust alternative.

Q6b) What is the Apriori Algorithm? Discuss applications with example. [6 marks]

The Apriori algorithm (Agrawal & Srikant, 1994) is the foundational algorithm for mining frequent itemsets and generating association rules from transactional databases. Its name comes from the Latin 'a priori' (from before) — referring to the key property it exploits: prior knowledge about subsets.

The Apriori Principle (Anti-Monotonicity Property)

If an itemset is infrequent (below min_support), then every superset of that itemset is also infrequent and can be pruned without counting. This dramatically reduces the number of candidates that need to be evaluated.

Algorithm Overview

- Pass 1: Count all single items and retain those meeting min_support → L1.
- Pass k: Generate candidate k-itemsets (Ck) by joining (k-1)-itemsets in L(k-1), then prune candidates that have any infrequent (k-1)-itemset subset, then scan the database to count and retain frequent ones → Lk.
- Repeat until no new frequent itemsets are found.
- Post-mining: Generate association rules from each frequent itemset and retain those meeting min_confidence.

A complete worked example is provided under May-June 2023, Q5c above.

Applications

- Market Basket Analysis: Supermarkets discover which product combinations are frequently purchased together. E.g., {Bread, Peanut Butter} → {Jelly} → place them near each other to increase basket size.
- Medical Diagnosis: Find patterns in patient symptom/test result datasets. E.g., {Chest Pain, High Cholesterol} → {Heart Disease} — helps flag high-risk patients.
- Web Usage Mining: Discover which sequences of web pages users tend to visit together, informing navigation design and personalized content recommendations.
- Fraud Detection: Identify combinations of transaction characteristics that frequently co-occur with fraudulent activity — e.g., {Overseas transaction, Unusual hour, High amount} → {Fraud}.
- Telecommunications: Discover calling patterns and service combinations subscribed together to design bundled offerings.

Q6c) Define — Frequent Itemset, Minimum Support Count, Hierarchical Clustering, Regression. [6 marks]

- Frequent Itemset: A set of items that appears together in at least 'minimum support count' transactions in a database. Example: If min_support_count = 3 and {Milk, Bread} appears in 5 transactions, it is a frequent itemset. Frequent itemsets are the building blocks from which association rules are derived.
- Minimum Support Count: The user-defined threshold specifying the minimum number of transactions in which an itemset must appear for it to be considered frequent. It filters out rare, statistically insignificant patterns. Example: With 1000 transactions and a 20% support threshold, the minimum support count = 200.
- Hierarchical Clustering: A clustering method that builds a hierarchy of clusters represented as a tree structure called a dendrogram. In agglomerative (bottom-up) hierarchical clustering, each data point starts as its own cluster, and the two closest clusters are iteratively merged until one cluster remains. The user can cut the dendrogram at any height to obtain any desired number of

clusters. Unlike K-Means, the number of clusters need not be specified in advance. It is deterministic (same result every run) but computationally expensive for large datasets.

- Regression: A supervised statistical/machine learning technique used to predict a continuous numerical output variable based on one or more input features. Simple Linear Regression models the relationship as a straight line: y = w0 + w1×x. Multiple Linear Regression extends this to multiple features. Logistic Regression, despite its name, is used for classification (predicts class probabilities). Regression is evaluated using metrics like Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and R-squared (coefficient of determination).

Additional Concepts & Quick Reference

Classification Algorithms Summary

Algorithm	Type	Key Idea	Best Used When
Naive Bayes	Probabilistic	Applies Bayes theorem with feature independence assumption	Text classification, spam filtering, small datasets
Logistic Regression	Linear	Sigmoid function maps linear model to [0,1] probability	Binary classification, interpretability required
Decision Tree	Tree-based	Recursive feature splits to minimize impurity	Mixed feature types, interpretability required
K-Nearest Neighbors	Instance-based	Classify based on K nearest training examples	Small datasets, no training time needed
SVM	Kernel-based	Find maximum-margin hyperplane separating classes	High-dimensional data, clear margin of separation

Evaluation Metrics Quick Reference

Metric	Formula	Ideal Value	Priority When...
Accuracy	(TP+TN) / Total	1.0 (100%)	Balanced classes
Precision	TP / (TP+FP)	1.0	FP is costly (e.g., spam filter)
Recall	TP / (TP+FN)	1.0	FN is costly (e.g., disease diagnosis)
F1 Score	2×P×R / (P+R)	1.0	Imbalanced dataset
AUC-ROC	Area under TPR vs FPR curve	1.0	Comparing models at all thresholds

Clustering Comparison

Aspect	K-Means	Hierarchical (Agglomerative)	DBSCAN
Cluster shape	Spherical	Any (based on linkage)	Any (density-based)
K required?	Yes	No	No
Handles outliers	Poor	Moderate	Excellent
Scalability	Good O(nKt)	Poor O(n² log n)	Moderate O(n log n)
Result	Flat partition	Dendrogram	Core/border/noise points

NOV-DEC 2025 [REPEATED] All questions repeated.

MAY-JUN 2025 [REPEATED] All questions repeated.

Cross-Reference: All Unit 5 Questions Across All 5 Years

Topic	MJ-23	ND-23	MJ-24	ND-25	MJ-25
Association Rule (support, confidence, lift)	Q5a	Q6a	-	Q6b	Q5a
Apriori Algorithm (worked)	Q5c	Q5c	-	-	Q5c
Classification vs Clustering (diff)	Q6b	Q5a	Q5b*	-	Q6b
Logistic Regression (types + example)	Q6c	Q5b*	Q5a	Q5a	Q6c
Bayes Theorem	Q6a	-	-	-	Q6a
Hierarchical vs Partitioning Clustering	Q5b	-	-	Q5b	-
K-Means Clustering (worked)	-	Q6c	-	-	-
Decision Tree (case study)	-	-	Q5c	Q5c	-
Classification Evaluation Formulae	-	Q6b	-	-	-
Definitions (Frequent Itemset etc.)	-	-	Q6c	Q6a	-
Short note on Classification	-	-	-	-	Q5b

Logistic Regression has now appeared in all five examination sessions — it is the single most tested topic in Unit 5. Association Rules and the Apriori algorithm are also consistently high-frequency. Apriori has appeared with the exact same dataset in three different sessions (May-Jun 2023, Nov-Dec 2023, May-Jun 2025), which means mastering that one worked example is essential.