

AI & ML Algorithm (315330)

Model Question Paper — Set A · Model Answers

Q.1

a) Applications of AI.

2 M

- **Healthcare** — disease diagnosis from X-ray / MRI images, drug discovery.
- **Self-driving cars and robotics** — perception, path planning, obstacle avoidance.
- **Natural language processing** — chatbots, virtual assistants (Siri, Alexa), machine translation.
- **Finance / banking** — fraud detection, credit scoring, algorithmic trading.
- **E-commerce** — product recommendation; **Gaming** — intelligent opponents (chess).

(Any four applications, ½ M each)

b) Reasoning; types.

2 M

Reasoning is the mental (logical) process of deriving new conclusions, facts or decisions from the existing knowledge, facts and beliefs. In AI, the inference engine performs reasoning on the knowledge base.

Types (any two): Deductive, Inductive, Abductive, Common-sense, Monotonic and Non-monotonic reasoning.

(Definition 1 M, two types 1 M)

c) Data Analytics vs Data Science.

2 M

Data Analytics	Data Science
Examines existing data to answer specific, known questions (what happened and why).	Broad field that builds models and algorithms to find new questions and predict the future.
Uses statistics, SQL, Excel, BI tools (Power BI, Tableau).	Uses ML, programming (Python/R), big-data tools.
Narrow scope — part of data science.	Wide scope — includes analytics, ML, data engineering.

(Any two points, 1 M each)

d) Machine learning life cycle (diagram).

2 M

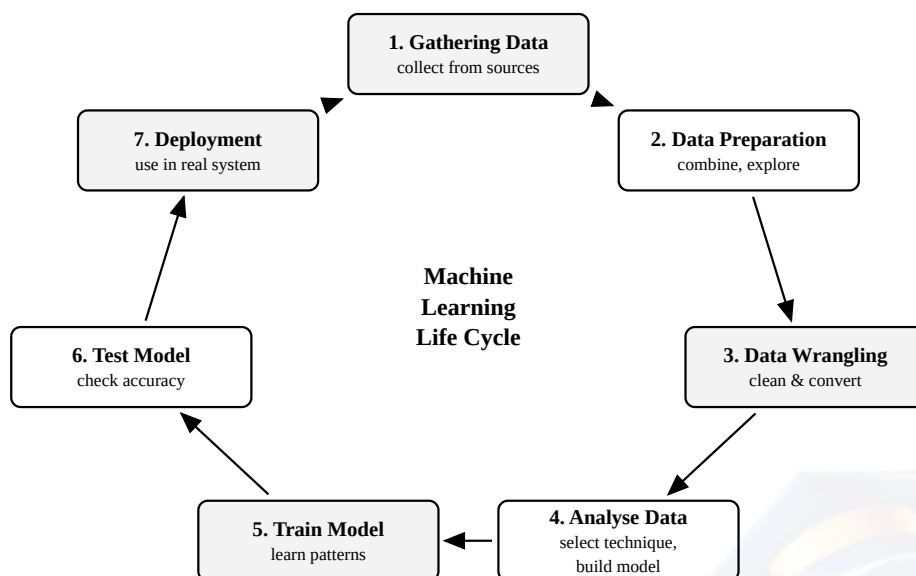


Fig.: Machine Learning life cycle (seven steps)

(Correct labelled diagram 2 M)

e) Unsupervised learning.

2 M

Unsupervised learning is a type of machine learning in which the model is trained on **unlabelled data** (only inputs, no output labels). The algorithm itself finds hidden patterns, groups or structure in the data.

Example: Customer segmentation — grouping customers of a shop into clusters according to their purchase behaviour using K-means clustering.

(Definition 1 M, example 1 M)

f) Precision and Recall.

2 M

Precision — out of all samples predicted as positive, the fraction that are actually positive:

Precision = $TP / (TP + FP)$.

Recall (Sensitivity, TPR) — out of all actual positive samples, the fraction correctly predicted as positive: **Recall = $TP / (TP + FN)$.**

(Each definition with formula 1 M)

g) Need of search algorithms.

2 M

- Most AI problems (route finding, puzzles, games, planning) are solved by **searching a state space** from the initial state to the goal state.
- Search algorithms give a **systematic way** to explore possible states/actions and find a **solution path** (sequence of actions).
- Good (heuristic) search finds the **optimal / least-cost** solution while saving **time and memory**.
- They are the basis of problem-solving agents, game playing and planning.

(Any two valid points, 1 M each)

Q.2

a) A* algorithm — trace.

4 M

A* is an informed (best-first) search that expands the node with the lowest $f(n) = g(n) + h(n)$, where $g(n)$ = actual cost from start to n and $h(n)$ = estimated cost from n to goal. It uses an OPEN list (nodes to be expanded, sorted on f) and a CLOSED list (expanded nodes). If $h(n)$ is admissible (never over-estimates), A* gives the optimal path.

Algorithm: 1) Put start S in OPEN with $f = h(S)$. 2) Remove the node n with smallest f from OPEN, put it in CLOSED. 3) If n is the goal, stop and trace back the path. 4) Otherwise generate successors; for each compute g , h , f ; add to OPEN (or update if a cheaper g is found). 5) Repeat from step 2.

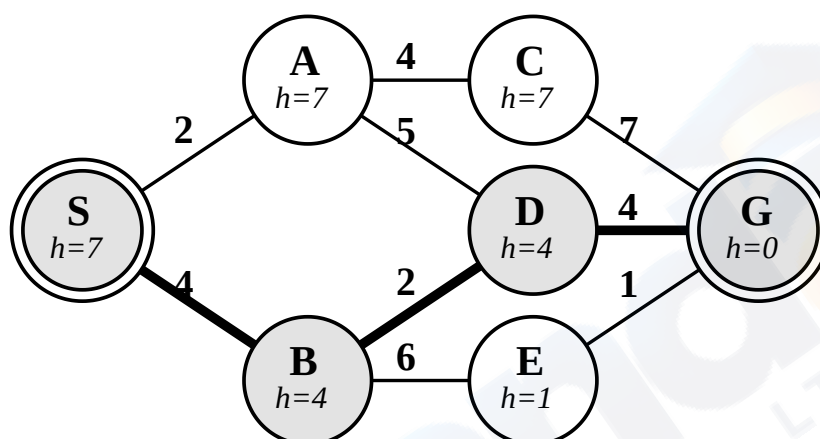


Fig.: Given graph (edge cost on edges, h inside nodes); bold = path found by A*

Step	Node selected (removed from Open)	Open list node($f = g + h$), sorted on f	Closed list
1	S ($g=0, h=7, f=7$)	B(8), A(9)	S
2	B ($g=4, h=4, f=8$)	A(9), D(10), E(11)	S, B
3	A ($g=2, h=7, f=9$)	D(10), E(11), C(13)	S, B, A
4	D ($g=6, h=4, f=10$)	G(10), E(11), C(13)	S, B, A, D
5	G ($g=10, h=0, f=10$)	Goal reached — stop	S, B, A, D, G

Path: $S \rightarrow B \rightarrow D \rightarrow G$; **cost** = $4 + 2 + 4 = 10$ (optimal). (Note: a greedy search using only h would go $S \rightarrow B \rightarrow E \rightarrow G$ with cost 11.)

(Explanation 1 M, trace 2 M, path and cost 1 M)

b) Bayes' theorem — numerical.

4 M

Bayes' theorem: For events A and B with $P(B) > 0$,

$$P(A|B) = P(B|A) \cdot P(A) / P(B)$$

$P(A)$ = prior probability, $P(B|A)$ = likelihood, $P(B)$ = evidence (total probability), $P(A|B)$ = posterior probability.

Given: D = has disease, $+$ = test positive. $P(D) = 0.02$, $P(\neg D) = 0.98$, $P(+|D) = 0.95$, $P(+|\neg D) = 0.08$.

Total probability of positive test: $P(+) = P(+|D)P(D) + P(+|\neg D)P(\neg D) = 0.95 \times 0.02 + 0.08 \times 0.98 = 0.0190 + 0.0784 = \mathbf{0.0974}$

Posterior: $P(D|+) = P(+|D)P(D) / P(+) = 0.0190 / 0.0974 = \mathbf{0.1951 \approx 19.5\%}$

So even after a positive test, the chance of really having the disease is only about 20%, because the disease is rare (low prior).

(Statement 1 M, P(+) 1½ M, posterior 1½ M)

c) Training dataset vs testing dataset.

4 M

Training dataset	Testing dataset
Used to train (fit) the ML model — the model learns patterns and parameters from it.	Used to evaluate the trained model on unseen data.
Larger part of data — usually 70–80%.	Smaller part — usually 20–30%.
Model sees the target labels during learning.	Labels are hidden from the model; used only to compare with predictions.
Accuracy on it shows how well the model fits (can be misleadingly high if overfitting).	Accuracy on it shows how well the model generalises .
Used in the training phase of the ML life cycle.	Used in the testing / evaluation phase.

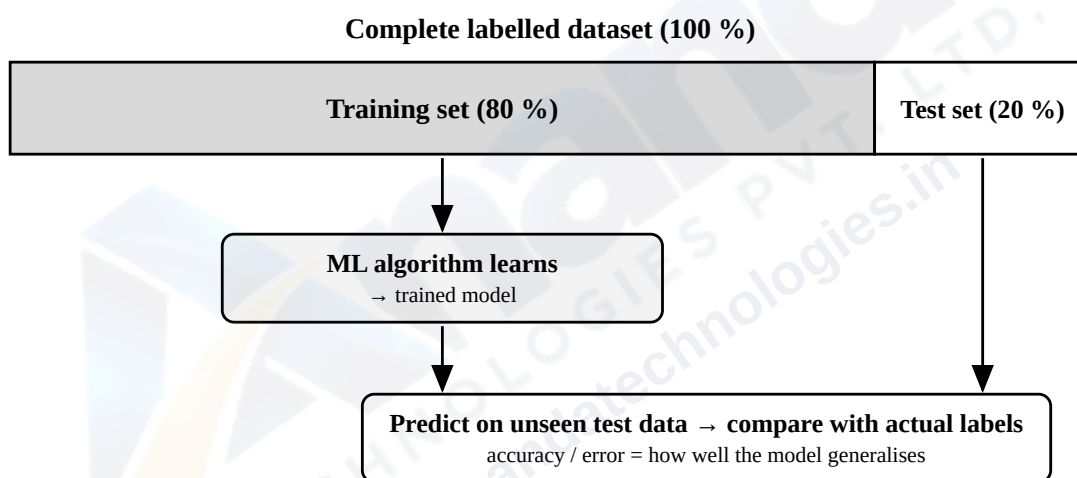


Fig.: Splitting a dataset into training (80%) and testing (20%) sets

(Any four points, 1 M each)

d) k-fold cross-validation; positive and negative class.

4 M

Cross-validation is a model-evaluation technique in which the data is divided into parts; the model is trained on some parts and tested on the remaining part, repeatedly, so that every record is used for both training and testing.

k-fold cross-validation: 1) Shuffle and divide data into k equal folds. 2) In each iteration use one fold as test set and remaining k–1 folds as training set. 3) Repeat k times so that each fold is tested once. 4) Final performance = average of k scores.

	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	
Iter 1	Test	Train	Train	Train	Train	→ score 0.80
Iter 2	Train	Test	Train	Train	Train	→ score 0.85
Iter 3	Train	Train	Test	Train	Train	→ score 0.90
Iter 4	Train	Train	Train	Test	Train	→ score 0.75
Iter 5	Train	Train	Train	Train	Test	→ score 0.85

Final score = average of the k scores

Fig.: 5-fold cross-validation; average score = $(0.80+0.85+0.90+0.75+0.85)/5 = 0.83$

Positive class — the class of interest that the model tries to detect (label 1), e.g. 'spam', 'disease present', 'fraud'. **Negative class** — the other class (label 0), e.g. 'not spam', 'healthy'. In cross-validation of a classifier, results of every fold are counted as TP, TN, FP, FN with respect to the positive class; **stratified k-fold** keeps the same ratio of positive and negative samples in every fold.

(Explanation 1 M, diagram 1 M, steps 1 M, positive/negative class 1 M)

Q.3

a) Greedy Best-First search — trace.

4 M

Greedy Best-First Search is an informed search that always expands the node that **appears closest to the goal**, i.e. the node with the smallest heuristic value: $f(n) = h(n)$. It uses a priority queue (OPEN list) ordered on $h(n)$. It is fast but **not optimal** and not complete in general (can get stuck in loops without a CLOSED list).

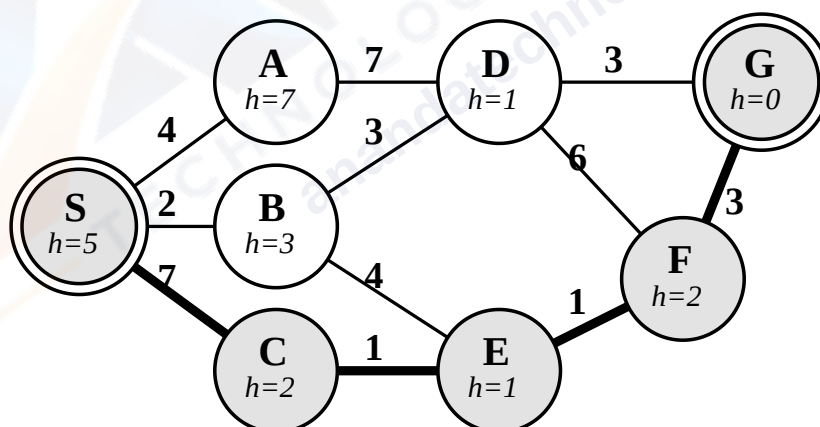


Fig.: Given graph; bold = path chosen by greedy best-first search

Step	Node selected (removed from Open)	Open list node(h), sorted on h	Closed list
1	S (h=5)	C(2), B(3), A(7)	S
2	C (h=2)	E(1), B(3), A(7)	S, C
3	E (h=1)	F(2), B(3), A(7)	S, C, E
4	F (h=2)	G(0), D(1), B(3), A(7)	S, C, E, F

5	G (h=0)	Goal reached — stop	S, C, E, F, G
---	---------	---------------------	---------------

Path found: $S \rightarrow C \rightarrow E \rightarrow F \rightarrow G$, cost = 12.

Comment: The optimal path is $S \rightarrow B \rightarrow D \rightarrow G$ with cost 8 (found by A*). Greedy search ignored the cost already travelled $g(n)$, so it reached the goal quickly with few expansions but the path is **not optimal**.

(Explanation 1 M, trace 2 M, comment 1 M)

b) Architecture of a knowledge-based agent.

4 M

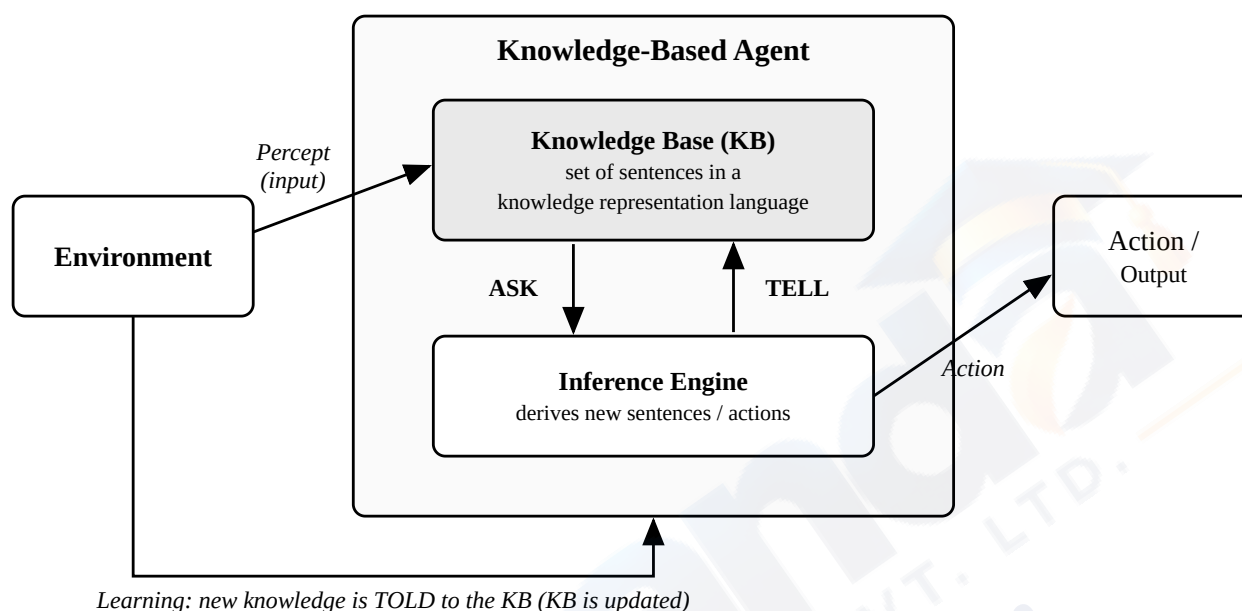


Fig.: Architecture of a knowledge-based agent

- **Knowledge Base (KB):** central component; a set of sentences (facts and rules) written in a knowledge-representation language (e.g. propositional / first-order logic). It holds what the agent knows about the world.
- **Inference engine:** applies inference rules on the KB to derive new facts and decide actions (forward / backward chaining).
- **TELL:** the agent tells the KB what it perceives (adds new sentences). **ASK:** the agent asks the KB what action to perform; the answer must follow from the KB.
- **Working cycle:** $\text{TELL}(\text{KB}, \text{percept}) \rightarrow \text{action} = \text{ASK}(\text{KB}, \text{what to do}) \rightarrow \text{TELL}(\text{KB}, \text{action taken})$. Learning updates the KB with new knowledge.

(Diagram 2 M, explanation 2 M)

c) Forms of data.

4 M

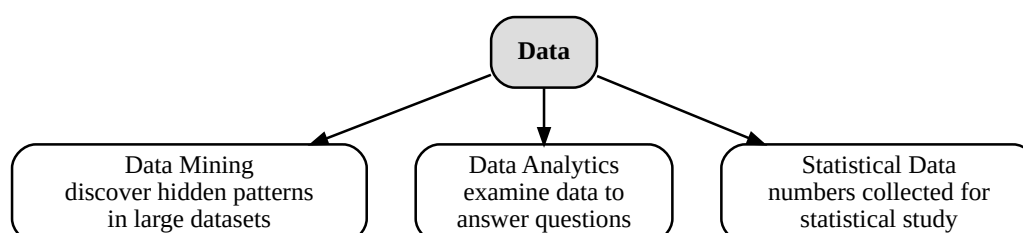


Fig.: Different forms of data used in ML

- **Data mining:** process of discovering hidden, previously unknown and useful patterns (associations, clusters, anomalies) from large datasets using ML, statistics and database techniques. Example: market-basket analysis — customers who buy bread also buy butter.
- **Data analytics:** examining, cleaning, transforming and modelling data to answer specific questions and support decisions (descriptive, diagnostic, predictive, prescriptive). Example: analysing monthly sales to find why sales dropped.
- **Statistical data:** numerical / categorical data collected, organised and summarised using statistical methods (mean, median, variance, probability, hypothesis testing). It may be qualitative or quantitative (discrete / continuous). Example: marks of students, census data.

ML uses all three: statistics to understand data, data mining to find patterns, and analytics to turn results into decisions.

(Each form with example 1 M + diagram/summary 1 M)

d) MAE, MSE and RMSE — numerical.

4 M

i	Actual y	Predicted $\hat{y}$	Error $e = y - \hat{y}$	e	e^2
1	12	10	2	2	4
2	15	16	-1	1	1
3	20	18	2	2	4
4	22	23	-1	1	1
5	25	27	-2	2	4
6	30	28	2	2	4
Σ				10	18

$$n = 6$$

$$\text{MAE} = \Sigma|y - \hat{y}| / n = 10 / 6 = \mathbf{1.6667}$$

$$\text{MSE} = \Sigma(y - \hat{y})^2 / n = 18 / 6 = \mathbf{3.0000}$$

$$\text{RMSE} = \sqrt{\text{MSE}} = \sqrt{3.0000} = \mathbf{1.7321}$$

RMSE $\geq$ MAE because squaring gives more weight to larger errors.

(Table 1 M, MAE 1 M, MSE 1 M, RMSE 1 M)

Q.4

a) Agglomerative hierarchical clustering — numerical.

4 M

Hierarchical clustering is an unsupervised algorithm that builds a hierarchy (tree) of clusters, shown as a **dendrogram**. **Agglomerative (bottom-up):** 1) start with each point as a separate cluster; 2) compute the distance matrix; 3) merge the two closest clusters; 4) update distances (single linkage = minimum distance between members); 5) repeat until one cluster remains. (Divisive clustering works top-down.)

Distances (Euclidean): $d(A,B) = \sqrt{[(1 - 2)^2 + (1 - 1)^2]} = 1.00$; $d(A,C) = \sqrt{[(1 - 4)^2 + (1 - 5)^2]} = 5.00$; $d(A,D) = \sqrt{[(1 - 6)^2 + (1 - 5)^2]} = 6.40$; $d(A,E) = \sqrt{[(1 - 7)^2 + (1 - 7)^2]} = 8.49$; $d(B,C) = \sqrt{[(2 - 4)^2 + (1 - 5)^2]} = 4.47$; $d(B,D) = \sqrt{[(2 - 6)^2 + (1 - 5)^2]} = 5.66$; $d(B,E) = \sqrt{[(2 - 7)^2 + (1 - 7)^2]} = 7.81$;

$$d(C,D) = \sqrt{[(4 - 6)^2 + (5 - 5)^2]} = 2.00; d(C,E) = \sqrt{[(4 - 7)^2 + (5 - 7)^2]} = 3.61; d(D,E) = \sqrt{[(6 - 7)^2 + (5 - 7)^2]} = 2.24$$

Step 1: minimum distance = 1.00 → merge **A** and **B**.

	A	B	C	D	E
A	0				
B	1.00	0			
C	5.00	4.47	0		
D	6.40	5.66	2.00	0	
E	8.49	7.81	3.61	2.24	0

Step 2: minimum distance = 2.00 → merge **C** and **D**.

	AB	C	D	E
AB	0			
C	4.47	0		
D	5.66	2.00	0	
E	7.81	3.61	2.24	0

Step 3: minimum distance = 2.24 → merge **CD** and **E**.

	AB	CD	E
AB	0		
CD	4.47	0	
E	7.81	2.24	0

Step 4: minimum distance = 4.47 → merge **AB** and **CDE**.

	AB	CDE
AB	0	
CDE	4.47	0

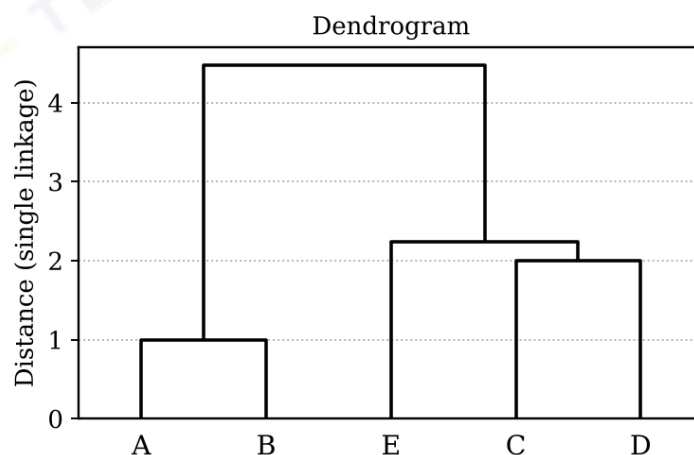


Fig.: Dendrogram (height of each join = merge distance)

(Explanation 1 M, merging steps 2 M, dendrogram 1 M)

b) Forward chaining vs backward chaining.

4 M

Forward chaining	Backward chaining
Starts from known facts and applies rules to reach the goal.	Starts from the goal and works backward to find supporting facts.
Data-driven , bottom-up approach.	Goal-driven , top-down approach.
Uses breadth-first strategy; may derive many unnecessary facts.	Uses depth-first strategy; only rules related to the goal are used.
Used in planning, monitoring, control, design (e.g. CLIPS).	Used in diagnosis, debugging, proving (e.g. PROLOG, MYCIN).

Example: Rules: R1: IF A AND B THEN C; R2: IF C THEN D. Facts: A, B. Goal: D.

Forward: A, B true → R1 gives C → R2 gives D → goal reached.

Backward: Goal D? → R2 needs C → R1 needs A and B → both are facts → D is proved.

(Any four points 3 M, example 1 M)

c) Overfitting and underfitting.

4 M

Overfitting: the model learns the training data too closely, including its noise, so it gives very high training accuracy but poor accuracy on new (test) data — **low bias, high variance**. Causes: too complex model, too many features, too little training data, training for too long. Remedies: more data, cross-validation, regularisation, pruning, early stopping, removing features.

Underfitting: the model is too simple to capture the pattern, so it performs badly on both training and test data — **high bias, low variance**. Causes: too simple model, too few features, too little training. Remedies: more complex model, add features, train longer, reduce regularisation.

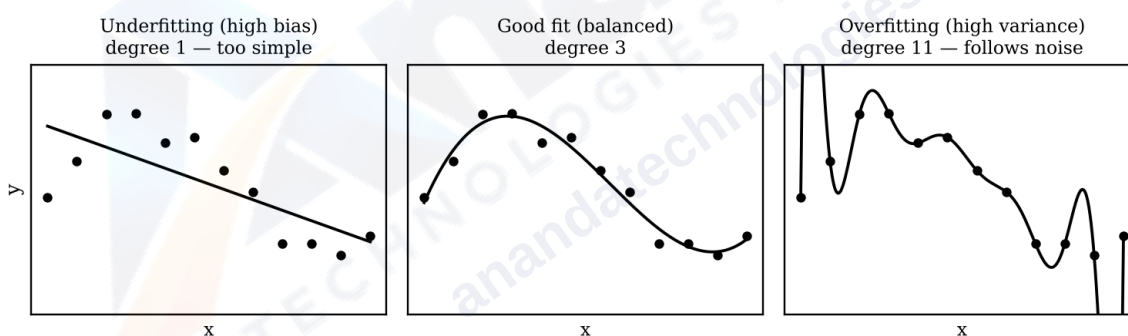


Fig.: Underfitting, good fit and overfitting on the same data

(Overfitting 1½ M, underfitting 1½ M, diagram 1 M)

d) K-means clustering — numerical.

4 M

Distance used: Euclidean $d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$. Initial centroids: C1 = P1(2, 3), C2 = P3(6, 8).

Iteration 1: C1 = (2, 3), C2 = (6, 8)

Point	d(P, C1)	d(P, C2)	Cluster
P1(2, 3)	0.00	6.40	C1
P2(3, 3)	1.00	5.83	C1

P3(6, 8)	6.40	0.00	C2
P4(7, 7)	6.40	1.41	C2
P5(2, 2)	1.00	7.21	C1
P6(8, 8)	7.81	2.00	C2
P7(3, 4)	1.41	5.00	C1

Cluster 1 = {P1, P2, P5, P7} → new C1 = $((2 + 3 + 2 + 3)/4, (3 + 3 + 2 + 4)/4) = (2.50, 3)$

Cluster 2 = {P3, P4, P6} → new C2 = $((6 + 7 + 8)/3, (8 + 7 + 8)/3) = (7, 7.67)$

Iteration 2: C1 = (2.50, 3), C2 = (7, 7.67)

Point	d(P, C1)	d(P, C2)	Cluster
P1(2, 3)	0.50	6.84	C1
P2(3, 3)	0.50	6.15	C1
P3(6, 8)	6.10	1.05	C2
P4(7, 7)	6.02	0.67	C2
P5(2, 2)	1.12	7.56	C1
P6(8, 8)	7.43	1.05	C2
P7(3, 4)	1.12	5.43	C1

Cluster 1 = {P1, P2, P5, P7} → new C1 = $((2 + 3 + 2 + 3)/4, (3 + 3 + 2 + 4)/4) = (2.50, 3)$

Cluster 2 = {P3, P4, P6} → new C2 = $((6 + 7 + 8)/3, (8 + 7 + 8)/3) = (7, 7.67)$

In iteration 2 the assignment is the same as before, so the centroids do not change → **stop**.

Final clusters: K1 = {P1, P2, P5, P7}, K2 = {P3, P4, P6}.

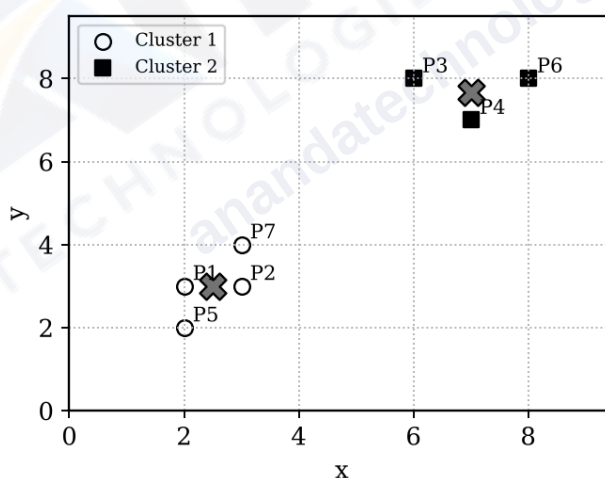


Fig.: Final clusters (X = centroid)

(Iteration 1 2 M, iteration 2 / convergence 1 M, final clusters 1 M)

e) Binary and multiclass classification.

4 M

Classification is supervised learning in which the output is a category (class label).

Binary classification: there are only **two classes** (positive / negative, 1 / 0). Examples: spam / not spam, pass / fail, fraud / genuine. Algorithms: logistic regression, SVM, decision tree. A threshold

(e.g. 0.5 on probability) decides the class.

Multiclass classification: there are **more than two classes** and each sample belongs to exactly one. Examples: digit recognition (0–9), classifying fruits as apple / mango / banana, iris species. Algorithms such as decision tree, k-NN, Naive Bayes and random forest handle many classes directly; binary classifiers are extended using **One-vs-Rest** (one classifier per class) or **One-vs-One** (one classifier per pair of classes).

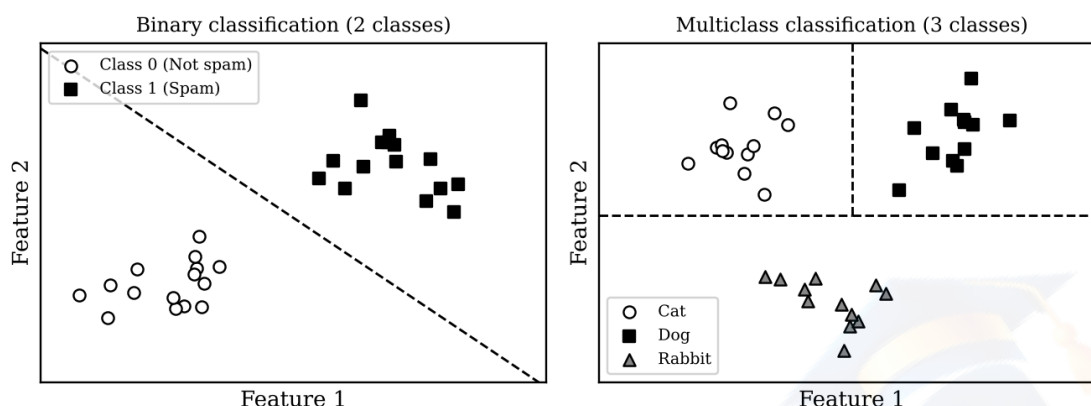


Fig.: Binary vs multiclass classification (dashed lines = decision boundaries)

(Binary 2 M, multiclass 2 M)

Q.5

a) Intelligent agent — structure and Turing test.

6 M

An **agent** is anything that perceives its **environment** through **sensors** and acts upon that environment through **actuators**. An **intelligent (rational) agent** selects the action that is expected to maximise its performance measure on the basis of its percepts and built-in knowledge. Examples: robot (camera, motors), software agent (keyboard input, screen output), self-driving car.

(i) Structure of an agent: Agent = Architecture + Agent program.

- **Architecture** — the machinery (hardware: sensors, actuators, computer) on which the program runs.
- **Agent function** — maps any percept sequence to an action: $f : P^* \rightarrow A$.
- **Agent program** — the implementation of the agent function that runs on the architecture.
- **Percept** — input received at an instant; **percept sequence** — complete history of percepts.

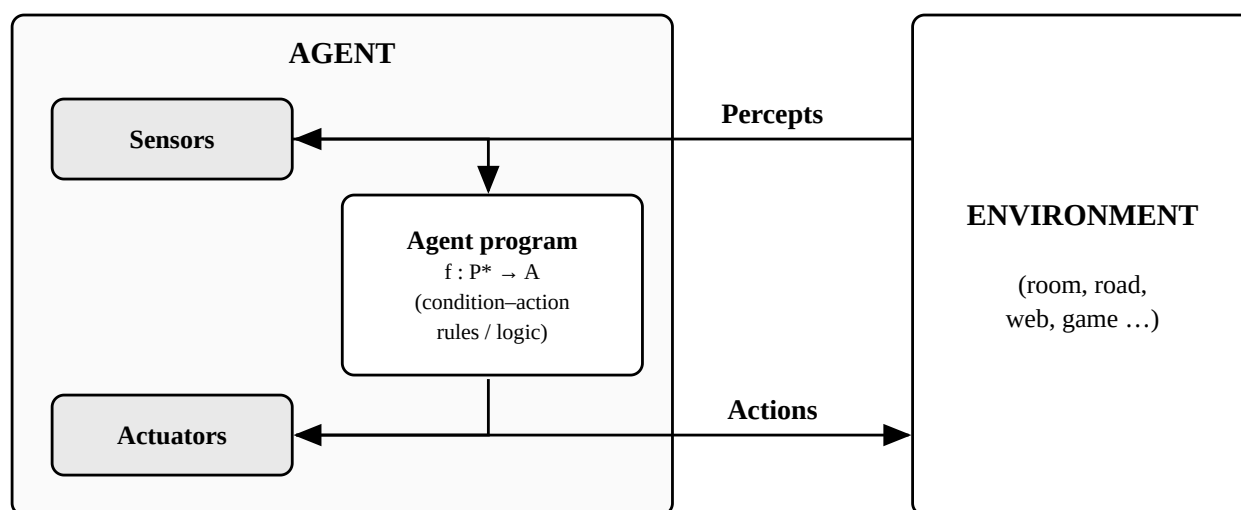


Fig.: Structure of an intelligent agent interacting with its environment

(ii) **Turing test** (Alan Turing, 1950) checks whether a machine can show human-like intelligence. An interrogator (C) communicates by typed text with a human (B) and a machine (A), both hidden in another room. C asks questions to both and must decide which is the machine. If C cannot reliably tell the machine from the human, the machine is said to **pass the test** (it can 'think'). To pass, the machine needs natural language processing, knowledge representation, automated reasoning and machine learning.

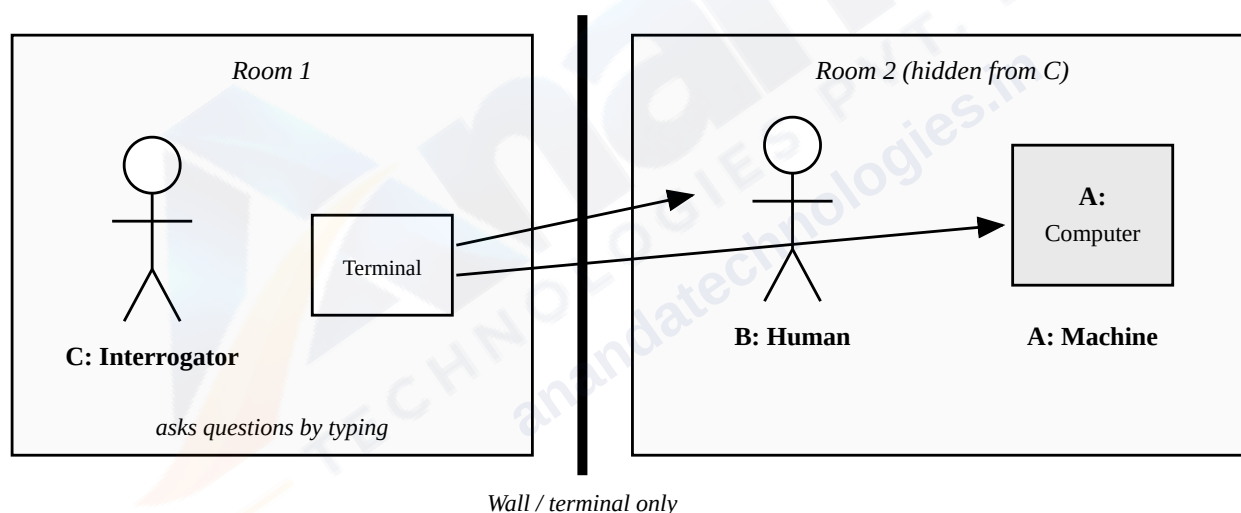


Fig.: Turing test — the interrogator (C) must find which one is the machine

(Agent definition 1 M, structure with diagram 2½ M, Turing test with diagram 2½ M)

b) Confusion matrix and metrics — numerical.

6 M

Confusion matrix (positive class = Spam):

		Predicted	
		Spam	Not spam
Actual	Spam	TP = 45	FN = 5
	Not spam	FP = 10	TN = 40

Total = TP + FN + FP + TN = 45 + 5 + 10 + 40 = 100

Accuracy = (TP + TN) / Total = (45 + 40) / 100 = **0.85** (85%)

Precision = TP / (TP + FP) = 45 / (45 + 10) = 45/55 = **0.8182**

Recall = TP / (TP + FN) = 45 / (45 + 5) = 45/50 = **0.9000**

F1-score = $2 \times \text{Precision} \times \text{Recall} / (\text{Precision} + \text{Recall}) = 2 \times 0.8182 \times 0.9000 / (0.8182 + 0.9000) = \mathbf{0.8571}$

Importance of confusion matrix:

- Shows not only how many predictions are correct but also the **type of errors** (FP = false alarm, FN = missed case).
- All classification metrics — accuracy, precision, recall, specificity, F1 — are calculated from it.
- Useful for **imbalanced data**, where accuracy alone is misleading.
- Helps to choose / tune the model and threshold for the problem (e.g. high recall for disease detection).

(Matrix 1 M, four metrics 4 M, importance 1 M)

c) Multiple linear regression in Python.

6 M

Multiple linear regression models a dependent variable y using two or more independent variables: $y = b_0 + b_1x_1 + b_2x_2 + \dots + b_nx_n$. The coefficients are found by the least-squares method (minimising the sum of squared errors).

```
# Multiple linear regression: price = b0 + b1*area + b2*bedrooms
import pandas as pd
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score

data = pd.DataFrame({
    'area':      [800, 1000, 1200, 1500, 1800, 2000, 2300, 2600],
    'bedrooms':  [1,   2,   2,   3,   3,   4,   4,   5],
    'price':     [32,  40,  45,  58,  66,  76,  83,  95]}) # Rs lakh
X = data[['area', 'bedrooms']] # independent variables
y = data['price']             # dependent variable

model = LinearRegression()
model.fit(X, y)               # train the model

print("Intercept (b0):", round(model.intercept_, 4))
print("Coefficients (b1, b2):", model.coef_.round(4))
print("R2 score:", round(r2_score(y, model.predict(X)), 4))

new = pd.DataFrame({'area': [1600], 'bedrooms': [3]})
print("Predicted price for 1600 sq ft, 3 BHK: Rs", round(model.predict(new)[0],
2), "lakh")
```

Output:

```
Intercept (b0): 5.8467
Coefficients (b1, b2): [0.0268 3.9403]
R2 score: 0.9993
Predicted price for 1600 sq ft, 3 BHK: Rs 60.54 lakh
```

The program builds the equation $\text{price} = b_0 + b_1 \cdot \text{area} + b_2 \cdot \text{bedrooms}$ from the data and uses it to predict the price of a new house; R^2 close to 1 means a good fit.

(Concept 1 M, program 4 M, output 1 M)

Q.6

a) Supervised vs unsupervised learning.

6 M

Point	Supervised learning	Unsupervised learning
Data	Trained on labelled data (input + correct output).	Trained on unlabelled data (input only).
Goal	Learn mapping input $\rightarrow$ output to predict labels of new data.	Find hidden patterns, groups or structure.
Supervision	Needs a "teacher" (labels) to guide learning.	No supervision; the model learns by itself.
Types of problems	Classification and regression.	Clustering, association, dimensionality reduction.
Algorithms	Linear / logistic regression, decision tree, k-NN, SVM, random forest.	K-means, hierarchical clustering, Apriori, PCA.
Accuracy / evaluation	Accuracy can be measured directly by comparing with known labels.	Harder to evaluate; no correct answers to compare.
Complexity	Simpler; but labelling data is costly and time consuming.	Computationally more complex; data easy to collect.
Example	Spam detection, house-price prediction.	Customer segmentation, market-basket analysis.

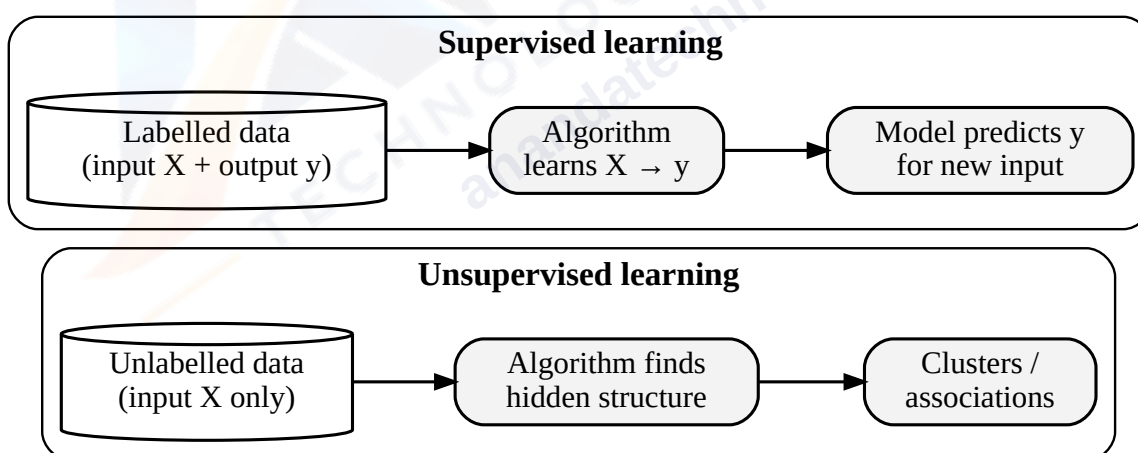


Fig.: Supervised vs unsupervised learning

(Any six points, 1 M each)

b) Decision tree using ID3 — numerical.

6 M

Formulas: Entropy $E(S) = -\sum p_i \log_2 p_i$; Gain(S, A) = $E(S) - \sum (|S_v|/|S|) E(S_v)$.

Step 1 — Entropy of whole data: Yes = 7, No = 3. $E(S) = -(7/10)\log_2(7/10) - (3/10)\log_2(3/10) = 0.8813$

Step 2 — Information gain of each attribute (Yes = 7, No = 3, $E = 0.8813$)

Attribute	Value	Yes	No	Entropy $E(S_v)$	Weighted E	Gain
Weather	Sunny	2	2	$-(2/4)\log_2(2/4) - (2/4)\log_2(2/4) = 1.0000$	0.6755	0.2058
	Cloudy	3	0	0 (pure node)		
	Rainy	2	1	$-(2/3)\log_2(2/3) - (1/3)\log_2(1/3) = 0.9183$		
Temperature	Hot	3	2	$-(3/5)\log_2(3/5) - (2/5)\log_2(2/5) = 0.9710$	0.8464	0.0349
	Mild	4	1	$-(4/5)\log_2(4/5) - (1/5)\log_2(1/5) = 0.7219$		
Wind	Weak	4	1	$-(4/5)\log_2(4/5) - (1/5)\log_2(1/5) = 0.7219$	0.8464	0.0349
	Strong	3	2	$-(3/5)\log_2(3/5) - (2/5)\log_2(2/5) = 0.9710$		

Highest gain → **Weather** is the root node.

Step 3 — Branch Weather = Sunny (Yes = 2, No = 2, $E = 1.0000$)

Attribute	Value	Yes	No	Entropy $E(S_v)$	Weighted E	Gain
Temperature	Hot	0	2	0 (pure node)	0.0000	1.0000
	Mild	2	0	0 (pure node)		
Wind	Weak	1	1	$-(1/2)\log_2(1/2) - (1/2)\log_2(1/2) = 1.0000$	1.0000	0.0000
	Strong	1	1	$-(1/2)\log_2(1/2) - (1/2)\log_2(1/2) = 1.0000$		

→ split this branch on **Temperature**; each child is pure.

Branch Weather = Cloudy: all records are 'Yes' → leaf node **Yes**.

Step 4 — Branch Weather = Rainy (Yes = 2, No = 1, $E = 0.9183$)

Attribute	Value	Yes	No	Entropy $E(S_v)$	Weighted E	Gain
Temperature	Mild	1	1	$-(1/2)\log_2(1/2) - (1/2)\log_2(1/2) = 1.0000$	0.6667	0.2516
	Hot	1	0	0 (pure node)		
Wind	Weak	2	0	0 (pure node)	0.0000	0.9183
	Strong	0	1	0 (pure node)		

→ split this branch on **Wind**; each child is pure.

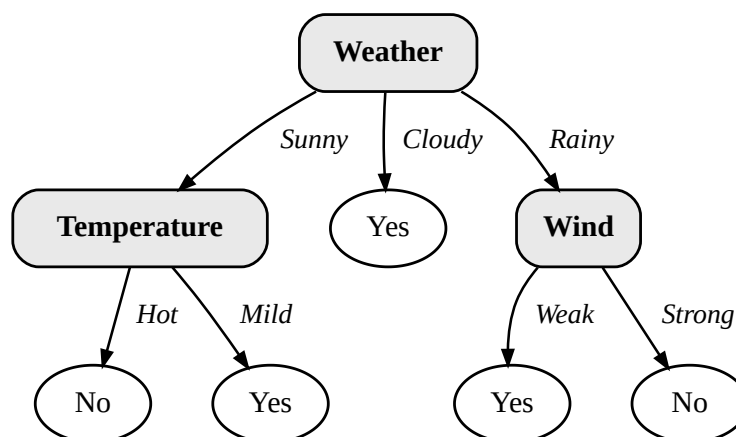


Fig.: Final decision tree (ID3)

(Entropy 1 M, gains 2 M, sub-tree 1 M, final tree 2 M)

c) Data cleaning with Python (pandas).

6 M

Data cleaning (data cleansing) is the process of detecting and correcting (or removing) incorrect, incomplete, duplicate, inconsistent or irrelevant data so that the dataset is accurate and ready for model training. 'Garbage in, garbage out' — dirty data gives a poor model.



Fig.: Data cleaning process

- **Missing data:** detect with `isnull()`; remove rows/columns (`dropna()`) or impute with mean / median / mode or a constant (`fillna()`).
- **Outliers:** values far from others (data-entry error or rare case); detect by IQR (outside $Q1 - 1.5 \cdot IQR$... $Q3 + 1.5 \cdot IQR$), Z-score or box plot; remove, cap or transform them.
- **Duplicates and structural errors:** `drop_duplicates()`; fix spelling, case, units and data types.

```

import pandas as pd
import numpy as np

df = pd.DataFrame({'Name': ['Amit', 'Riya', 'Sahil', 'Neha', 'Om', 'Pooja', 'Raj', 'Isha'],
                  'Age': [21, 22, np.nan, 20, 23, 21, np.nan, 22],
                  'Marks': [68, 72, 75, 70, 300, 66, 74, np.nan]})

print("Missing values in each column:")
print(df.isnull().sum())

# 1. Handle missing data: fill Age with mean, Marks with median
df['Age'] = df['Age'].fillna(df['Age'].mean())
df['Marks'] = df['Marks'].fillna(df['Marks'].median())

# 2. Detect outliers in Marks using IQR method
Q1 = df['Marks'].quantile(0.25)
Q3 = df['Marks'].quantile(0.75)
IQR = Q3 - Q1
low, high = Q1 - 1.5 * IQR, Q3 + 1.5 * IQR
print("\nQ1 =", Q1, " Q3 =", Q3, " IQR =", IQR, " limits:", low, "to", high)
  
```

```
print("Outliers:")
print(df[(df['Marks'] < low) | (df['Marks'] > high)])

# 3. Remove outlier rows
clean = df[(df['Marks'] >= low) & (df['Marks'] <= high)]
print("\nCleaned data:")
print(clean.round(2).to_string(index=False))
```

Output:

Missing values in each column:

```
Name      0
Age        2
Marks      1
dtype: int64
```

Q1 = 69.5 Q3 = 74.25 IQR = 4.75 limits: 62.375 to 81.375

Outliers:

	Name	Age	Marks
4	Om	23.0	300.0

Cleaned data:

	Name	Age	Marks
	Amit	21.0	68.0
	Riya	22.0	72.0
	Sahil	21.5	75.0
	Neha	20.0	70.0
	Pooja	21.0	66.0
	Raj	21.5	74.0
	Isha	22.0	72.0

(Explanation 2 M, program 3 M, output 1 M)

Note: This is a **model question paper prepared by Ananda Technologies** only for practice and exam preparation, based on the pattern of previous MSBTE question papers and the K-scheme syllabus. It is **not an official MSBTE paper**, and it does **not** mean that these questions will come in the exam. Answers follow standard textbook points; marks shown are indicative. Free for students — anandatechnologies.in