UNIT–3 : Decision Tree and Instance-Based Learning (PYQ Solution)

AKTU · Machine Learning Techniques (BCDS062) · Solutions by **Chirag Jain**, Ycotes

SECTION A Answers (≈70 Words Each)

Q1. What is Decision Tree Learning? For which problems is it best suited? (May 2026, Jan 2024)

Decision tree learning builds a tree in which internal nodes test attributes, branches are test outcomes and leaves give class labels; it classifies by following a path from the root. It is best suited to problems with attribute-value instances, discrete target values, possibly noisy or missing data, and where disjunctive (if-then) rules are needed, such as loan approval and medical diagnosis.

Q2. What is entropy in decision tree learning? (Jan 2026)

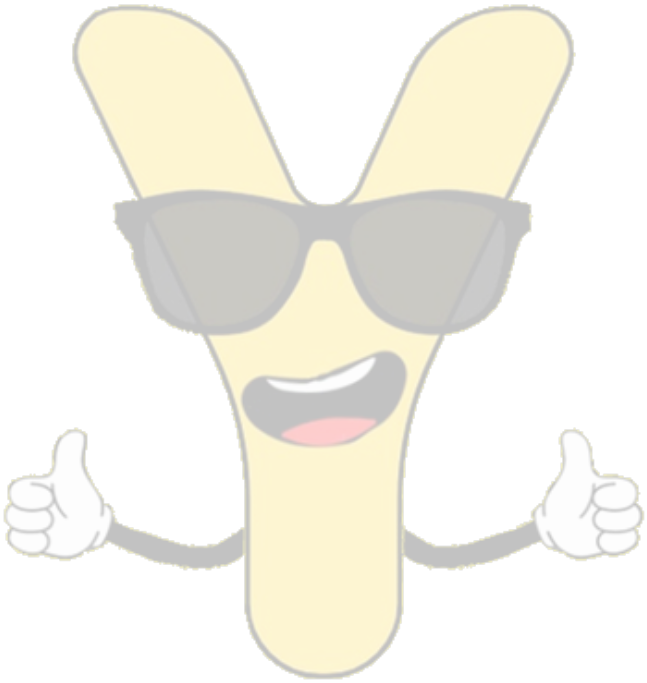
Entropy measures the impurity or uncertainty of a set S: Entropy(S) = −Σ pi $\log_2$ pi, where pi is the fraction of class i. It is 0 for a pure set and 1 for a 50/50 binary split. ID3 chooses the attribute that reduces entropy the most, i.e. gives the highest information gain.

Q3. Discuss overfitting and underfitting in decision trees. How can overfitting be avoided? (Jan 2025, Jan 2024)

Overfitting: the tree fits noise and training detail, giving high training but low test accuracy. Underfitting: the tree is too simple to capture the pattern and is poor on both. Overfitting is avoided by pre-pruning (stop growing early), post-pruning (reduced-error or rule post-pruning), limiting depth, requiring a minimum number of samples per leaf, and using a validation set or more data.

Q4. Explain Case-Based Learning. Differentiate between Lazy and Eager learning. (Jan 2024)

Case-based learning is an instance-based method that stores past cases with their solutions and solves a new problem by retrieving and adapting similar cases (retrieve, reuse, revise, retain). Lazy learning (k-NN, case-based) delays generalisation until a query arrives, so training is fast and prediction slow. Eager learning (decision trees, neural networks) builds a global model during training, so prediction is fast.



SECTION B Answers (≈250 Words Each)

Q1. What is information gain, and how does ID3 use it to build a decision tree? Illustrate the steps of the ID3 algorithm. (May 2026, Jan 2025)

ID3 (Iterative Dichotomiser 3) builds a decision tree top-down, greedily choosing at each node the attribute that best separates the examples.

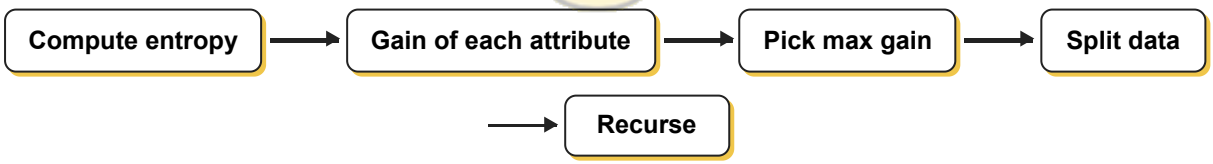
Entropy: Entropy(S) = $-p_+ \log_2 p_+ - p_- \log_2 p_-$ measures impurity.

Information gain: Gain(S, A) = Entropy(S) − Σ (|Sv| / |S|) × Entropy(Sv), the expected drop in entropy after splitting S on attribute A over its values v.

Steps of ID3:

1. Create a root node; if all examples have the same class, return a leaf with that class.
2. If no attributes remain, return a leaf with the majority class.
3. Compute Gain for every remaining attribute.
4. Select the attribute **A with maximum gain** and make it the node.
5. For each value of A create a branch and the subset of examples with that value.
6. If a subset is empty, add a leaf with the majority class; otherwise **recurse** on the subset with A removed.

Example: 14 examples with 9 Yes and 5 No: Entropy = $-(9/14)\log_2(9/14) - (5/14)\log_2(5/14) = 0.940$. If Wind splits into Weak (8: 6 Yes, 2 No, entropy 0.811) and Strong (6: 3 Yes, 3 No, entropy 1.0), Gain = 0.940 − (8/14)(0.811) − (6/14)(1.0) = **0.048**. The attribute with the highest gain becomes the root.



ID3 steps

Features and limits: it needs no backtracking, handles discrete attributes, but prefers many-valued attributes and overfits without pruning.

Q2. Explain the k-Nearest Neighbour (k-NN) algorithm and its limitations.

(Jan 2026)

k-NN is a simple **lazy, instance-based** learner: it stores all training examples and classifies a new point by the majority class among its **k closest** training points.

Algorithm:

1. Choose k (odd for binary classification).
2. Compute the distance (usually Euclidean, $d = \sqrt{\Sigma(x_i - y_i)^2}$) from the query to every training point.
3. Take the k nearest neighbours.
4. Classify by **majority vote** (or average for regression, optionally weighted by $1/d^2$).

Example: training points A(1, 1) class Red, B(2, 2) Red, C(5, 5) Blue, D(6, 5) Blue. Query (2, 1), k = 3. Distances: A 1.00, B 1.00, C 5.00, D 4.47. Nearest three are A, B, D, giving 2 Red and 1 Blue, so the query is **Red**.

Limitations:

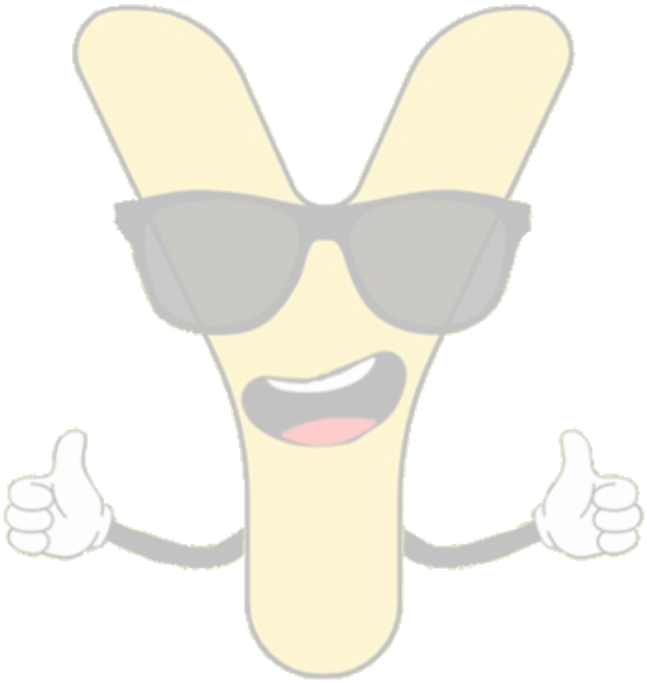
- **Slow prediction** and high memory, since all data is stored and scanned.
- Sensitive to **irrelevant features** and to feature scale (needs normalisation).
- **Curse of dimensionality:** distances lose meaning in high dimensions.
- Sensitive to noise and to the choice of k; poor on imbalanced classes.
- No explicit model, so low interpretability.

Q3. Compare Information Gain, Gain Ratio and Gini Index. (Jan 2024)

These are attribute-selection measures in decision trees.

- **Information Gain** (ID3): Gain = Entropy(S) − Σ(|Sv|/|S|) Entropy(Sv). It is biased towards attributes with many values (e.g. an ID column).
- **Gain Ratio** (C4.5): GainRatio = Gain / SplitInfo, where SplitInfo = $-\Sigma(|Sv|/|S|) \log_2(|Sv|/|S|)$. It penalises many-valued attributes.
- **Gini Index** (CART): Gini(S) = $1 - \Sigma\, p_i^2$. A weighted Gini of the children is minimised.

Basis	Information Gain	Gain Ratio	Gini Index
Algorithm	ID3	C4.5	CART
Based on	Entropy (log)	Gain normalised by split info	Probability squares
Bias	Many-valued attributes	Reduces that bias; may favour tiny splits	Favours larger partitions
Range (binary)	0 to 1	0 to 1	Gini 0 to 0.5
Cost	Log computations	Slightly higher	Cheaper (no log)
Goal	Maximise	Maximise	Minimise



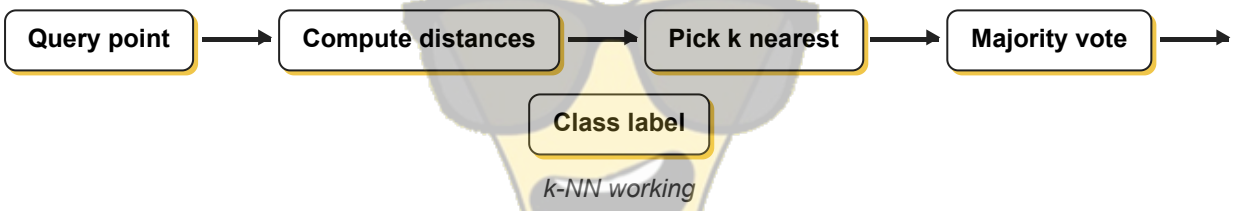
SECTION C Answers (≈500 Words Each)

Q1. What is Instance-Based Learning, and how does k-NN work? Demonstrate k-NN classification with an example. (May 2026, Jan 2024)

Instance-based learning (memory-based or lazy learning) stores the training examples and postpones generalisation until a query arrives. A new instance is classified by comparing it with the stored ones, using a local approximation for each query. Examples: k-NN, locally weighted regression, case-based reasoning.

k-NN algorithm:

1. Store all training data (features and labels).
2. For a query x, compute the distance to every training point: Euclidean $d = \sqrt{\Sigma(x_i - y_i)^2}$.
3. Select the k nearest points.
4. Assign the **majority class** (regression: mean of their values).



k-NN working

Example: classify a fruit by weight (g) and size (cm) with k = 3.

Fruit	Weight	Size	Class
1	150	7	Apple
2	160	8	Apple
3	120	5	Orange
4	110	4	Orange
Query: (140, 6). Distances: to 1 = √(100 + 1) = 10.05, to 2 = √(400 + 4) = 20.1, to 3 = √(400 + 1) = 20.02, to 4 = √(900 + 4) = 30.07. Nearest three: fruits 1, 3, 2 → Apple, Orange, Apple. Majority = **Apple**.			

Choosing k: small k is noisy and overfits; large k smooths too much. Use cross-validation; scale features first.

Advantages: simple, no training, adapts to new data. **Disadvantages:** slow prediction, memory heavy, affected by irrelevant features and high dimensions.

Conclusion: k-NN is an intuitive local method that is effective on small, low-dimensional data.

Q2. Explain Locally Weighted Regression, Radial Basis Function Networks and Case-Based Learning with examples. How does LWR differ from traditional regression, and what are its applications? (May 2026, Jan 2025)

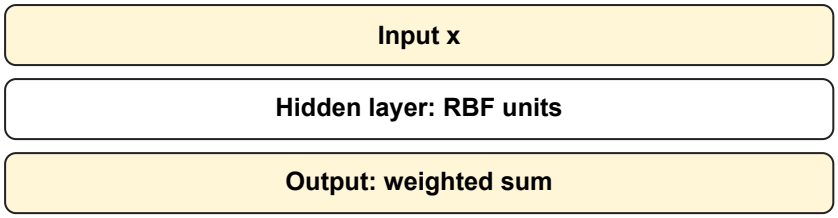
1. Locally Weighted Regression (LWR): a lazy method that fits a separate regression model for each query x_q using only nearby training points. Each training point gets a weight that falls with distance, e.g. $w_i = \exp(-(x_i - x_q)^2 / 2\tau^2)$ where τ is the bandwidth. The local model minimises $\Sigma\, w_i (y_i - \hat{y}_i)^2$ and predicts at x_q only. Example: predicting house price near a given area by fitting a line through similar-sized houses.

Difference from traditional regression:

Basis	Traditional (global)	LWR
Model	One fixed model for all data	New model per query
Training	Parameters learned once	Lazy; fitted at prediction time
Shape	Fixed form (line)	Fits non-linear curves locally
Prediction cost	Low	High
Accuracy	Poor on non-linear data	Good on non-linear data

Applications: robot control, time-series and sensor smoothing, price prediction, any non-linear data.

2. Radial Basis Function Networks: a two-layer network. Hidden units use radial functions, usually Gaussian $\varphi_j(x) = \exp(-\|x - c_j\|^2 / 2\sigma^2)$, centred at c_j; the output is a linear combination $f(x) = w0 + \Sigma\, w_j\, \varphi_j(x)$. Centres are chosen by clustering or from training points, and output weights by least squares. It acts as a local approximator. Example: function approximation and pattern classification (e.g. speech recognition).



RBF network

3. Case-Based Learning (CBR): instances are rich cases (problem plus solution). Cycle: **Retrieve** similar case, **Reuse** its solution, **Revise** if needed, **Retain** the new case. Example: a help-desk system that retrieves an earlier fault ticket and adapts its fix to the new complaint; also used in legal and medical reasoning.

Conclusion: all three approximate the target function locally; they differ in how instances are represented and how the local model is built.

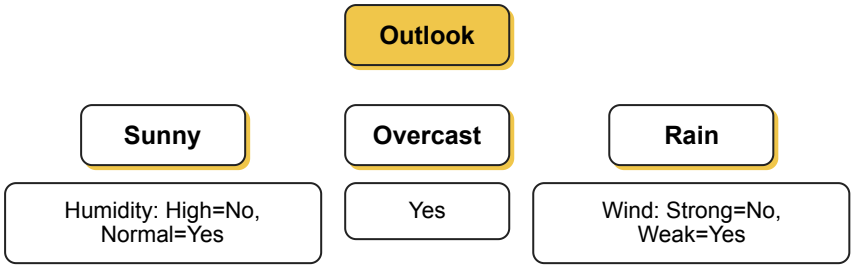
Q3. Explain the ID3 algorithm for decision tree learning. Discuss a Decision Tree and its working. (Jan 2026, Jan 2024)

A **decision tree** is a flowchart-like classifier: **internal nodes** test an attribute, **branches** are outcomes and **leaf nodes** hold a class label. To classify an instance, start at the root, follow the branch matching the attribute value, and continue until a leaf is reached. A tree can be read as a set of if-then rules (a disjunction of conjunctions).

ID3 algorithm (greedy, top-down):

1. If all examples are of one class, return a leaf of that class.
2. If no attributes remain, return a leaf with the majority class.
3. For each attribute compute **Information Gain** = Entropy(S) − Σ (|Sv|/|S|) Entropy(Sv), where Entropy(S) = −Σ pi $\log_2$ pi.
4. Choose the attribute with the highest gain as the decision node.
5. Create a branch per value, partition the data, and repeat on each partition with that attribute removed.

Example (PlayTennis, 14 cases, 9 Yes / 5 No): Entropy = 0.940. Gain(Outlook) = 0.246, Gain(Humidity) = 0.151, Gain(Wind) = 0.048, Gain(Temperature) = 0.029. **Outlook** becomes the root. Its Overcast branch is pure (Yes); Sunny splits on Humidity; Rain splits on Wind.



PlayTennis decision tree

Strengths: easy to interpret, handles categorical data, fast. **Weaknesses:** overfits (use pruning), prefers many-valued attributes, no backtracking so may reach a local optimum.

Conclusion: ID3 recursively picks the most informative attribute to build a compact tree.

Q4. Discuss inductive bias and inductive inference in decision trees. (Jan 2026)

Inductive inference is reasoning from specific training examples to a general rule, so that unseen instances can be classified. A decision tree learner performs inductive inference when it generalises from the training set to a tree.

Since many trees fit the same data, the learner must prefer one. This preference is the **inductive bias**: the set of assumptions that lets the learner classify unseen instances. Without bias a learner can only memorise (it cannot generalise).

Inductive bias of ID3:

- It searches the hypothesis space of **all finite discrete-valued functions**, so it is a complete space with no restriction bias.
- It has a **preference (search) bias**: it prefers **shorter trees** over longer ones, and trees that put **high-information-gain attributes close to the root**.
- This follows **Occam's razor**: the simplest hypothesis consistent with the data is preferred.

Restriction vs preference bias: candidate elimination has a restriction bias (only conjunctive hypotheses are considered), while ID3 has a preference bias (a full space, but search order favours short trees). Preference bias is generally better because the target function is never wrongly excluded.

Why prefer short trees? There are fewer short hypotheses than long ones, so a short hypothesis that fits the data is less likely to be a coincidence. Counter-

argument: the definition of "short" depends on representation.

Example: two trees fit the same data: a 3-node tree on Outlook and Humidity and a 9-node tree using Temperature too. ID3 returns the 3-node tree.

Conclusion: inductive bias is what makes generalisation possible; for ID3 it is the preference for short trees with informative attributes near the root.

Q5. For the dataset A = (1, 2, 1, 2, 1, 2), B = (2, 3, 3, 2, 2, 3), T = (Yes, No, Yes, No, Yes, No), calculate the entropy before and after splitting on A. Which attribute gives the highest information gain? (Jan 2025)

Entropy before split: T has 3 Yes and 3 No out of 6.
$E(T) = -(3/6)\log_2(3/6) - (3/6)\log_2(3/6) = 0.5 + 0.5 =$ **1.0**.

Split on A:

- A = 1: rows 1, 3, 5 → Yes, Yes, Yes (3 Yes, 0 No). Entropy = 0.
- A = 2: rows 2, 4, 6 → No, No, No. Entropy = 0.
- Weighted entropy after split = (3/6)(0) + (3/6)(0) = **0**.
- **Gain(A) = 1.0 − 0 = 1.0.**

Split on B:

- B = 2: rows 1, 4, 5 → Yes, No, Yes (2 Yes, 1 No). Entropy = $-(2/3)\log_2(2/3) - (1/3)\log_2(1/3) = 0.390 + 0.528 = 0.918$.
- B = 3: rows 2, 3, 6 → No, Yes, No (1 Yes, 2 No). Entropy = 0.918.
- Weighted entropy = (3/6)(0.918) + (3/6)(0.918) = 0.918.
- **Gain(B) = 1.0 − 0.918 = 0.082.**

Attribute	Entropy after split	Information gain
A	0	1.0
B	0.918	0.082

Conclusion: Attribute A gives the highest information gain (1.0) because it perfectly separates Yes and No (A = 1 → Yes, A = 2 → No), so it is chosen as the root and the tree needs no further splits.
