Machine Learning Exam Answer Key

Machine Learning for IT Application Development

Question Bank (Q1 to Q63) with questions, answers and diagrams, plus previous year questions (PYQs) not covered by the bank

Part	Contents
Part A	Question bank: Unit 4 Classification (Q1 to Q21), Unit 5 Optimization (Q22 to Q42), Unit 6 Semi-supervised and Reinforcement Learning (Q43 to Q63)
Part B	PYQs not in the bank (B1 to B28), with a map showing where the other PYQs are answered

Contents

Contents 2

Part A: Question Bank 3

Unit 4: Classification Techniques (Q1 to Q21) . . . 3

Unit 5: Optimization Techniques (Q22 to Q42) . . . 10

Unit 6: Semi-Supervised and Reinforcement Learning (Q43 to Q63) . . . 16

Part B: PYQs Not Covered in the Question Bank 22

Where the already-covered PYQs are answered . . . 22

Unit 1 and 2: Introduction and Data Handling . . . 22

Unit 3, 4 and Other PYQ Topics . . . 26

Part A: Question Bank

Unit 4: Classification Techniques (Q1 to Q21)

Q1 **Define a Support Vector Machine (SVM) and explain its primary purpose in machine learning.** **[4]**

SVM is a supervised algorithm that finds the best separating **hyperplane** between classes with the **maximum margin**. **Purpose:** accurate classification (and regression, as SVR), especially for high-dimensional data.

- **Hyperplane:** the decision boundary $w \cdot x + b = 0$ (a line in 2D, a plane in 3D).
- **Support vectors:** the training points closest to the hyperplane; only they decide its position.

Q2 **Define the k-Nearest Neighbours (KNN) algorithm and explain its fundamental principle.** **[4]**

k-NN is an instance-based, non-parametric algorithm. **Principle:** similar points lie close together. To classify a new point: compute its distance (usually Euclidean) to all training points, pick the k nearest, and take a majority vote (classification) or the average (regression). A worked example is in Q12.

Q3 **Define the CART algorithm and explain its primary purpose in machine learning.** **[4]**

Classification And Regression Trees build a **binary** tree. At every node it picks the split that gives the best purity (**Gini index** for classification, **MSE reduction** for regression). Purpose: an interpretable model made of simple if-else rules.

Q4 **Define Bagging as an ensemble method in machine learning.** **[4]**

Bootstrap Aggregating: train many models in parallel, each on a random **bootstrap sample** (sampling with replacement), then combine by majority vote (classification) or averaging (regression). It **reduces variance**. Example: Random Forest.

Q5 **State the issues in decision tree learning.** **[4]**

- **Overfitting** on deep trees (fix: pruning).
- Handling continuous attributes (needs thresholds) and missing values.
- Information gain is biased towards attributes with many values (fix: gain ratio).
- **Instability:** a small change in data can give a very different tree.
- Greedy construction, so the tree is only locally optimal.
- Biased towards the majority class on imbalanced data.

Pruning (also asked as a PYQ): **pre-pruning** stops growth early (max depth, minimum samples per leaf); **post-pruning** grows the full tree and then removes branches that do not improve validation accuracy (e.g. cost-complexity pruning).

Q6 **Justify the statement "KNN is called a lazy learner algorithm".** **[4]**

k-NN has **no training phase**. It only stores the training data and builds no model. All computation (distance calculation and voting) happens at **prediction time**. Hence it is lazy: training is instant, prediction is slow and memory use is high.

Q7 **Define Boosting as an ensemble method in machine learning.** **[4]**

A **sequential** ensemble. Each new weak learner concentrates on the examples the previous learners got wrong (their weights are increased). The final prediction is a **weighted vote**. It **reduces bias**. Examples: AdaBoost, Gradient Boosting, XGBoost.

Q8 **Explain the core idea behind Bagging and how it differs from Boosting in terms of how base models are combined.** **[4]**

- **Bagging:** independent models trained in parallel on bootstrap samples; combined by an **equal-weight** vote or average; reduces variance.
- **Boosting:** models trained one after another, each fixing its predecessor's errors; combined by a **weighted** vote (better models get more weight); reduces bias.

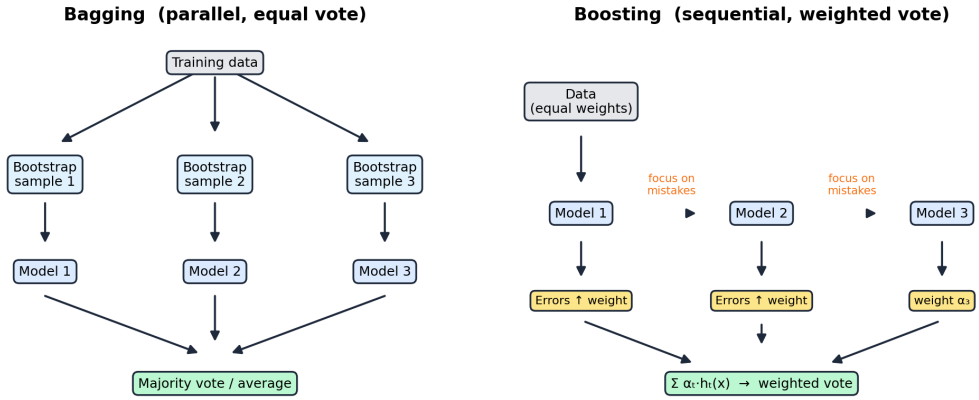

Figure: Bagging (parallel) vs Boosting (sequential)

Q9 **Describe the concept of a margin in SVMs. How does the margin relate to the separation of data points in SVM classification?** **[4]**

Margin = the distance between the hyperplane and the nearest points of each class (the support vectors). Its width is **2/‖w‖**.

- A wider margin means better separation, a more confident classifier and **better generalisation**.
- SVM maximises the margin by minimising ½‖w‖² subject to $y_i(w \cdot x_i + b) \geq 1$.
- Soft margin (parameter C) tolerates a few misclassifications for noisy data.

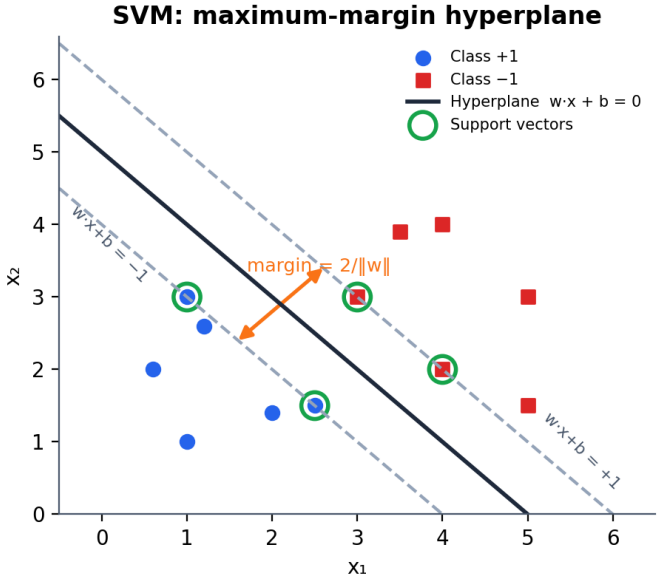

Figure: Maximum-margin hyperplane, margin and support vectors

Q10 **Compare and contrast linear and non-linear SVMs. Discuss when you would choose one over the other.** **[4]**

Point	Linear SVM	Non-linear SVM
Boundary	Straight hyperplane	Curved, via the **kernel trick** (RBF, polynomial)
Data	Linearly separable	Not linearly separable
Speed / tuning	Fast, only C	Slower; C and gamma/degree
Overfit risk	Low	Higher

Choose linear for separable, very high-dimensional or sparse data (text) and large datasets. **Choose non-linear** when classes overlap in a curved pattern.

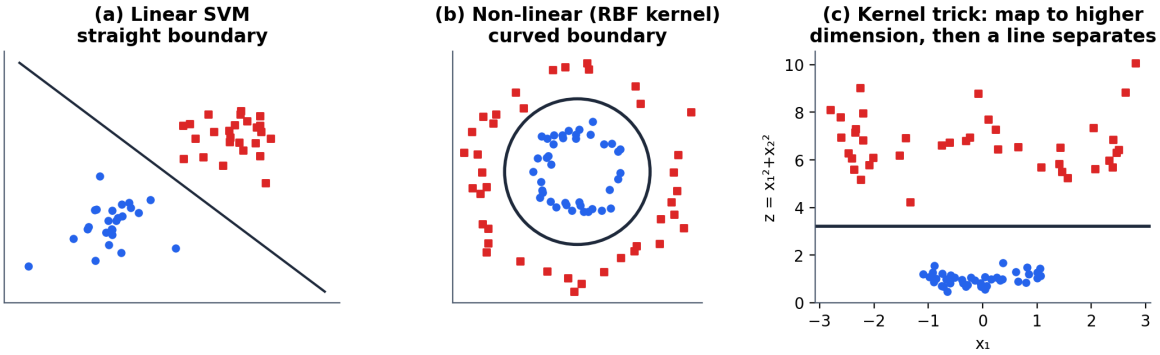

Figure: Linear boundary, curved RBF boundary, and the kernel trick

Q11 **Discuss the impact of choosing a larger value of "k" in KNN. How does it affect the model's bias and variance, and what considerations should be made when selecting an appropriate "k"?** **[4]**

- Larger k gives a smoother boundary: **higher bias, lower variance** (tends to underfit).
- Small k gives a jagged boundary: low bias, **high variance** (overfits noise).
- Choosing k: use cross-validation; use an odd k for binary problems to avoid ties; rule of thumb k ≈ √n; scale the features; consider the noise level.

Q12 **Consider the data with attributes (acid durability and strength) to classify whether a special paper tissue is good or not. For a new sample with X1=3 and X2=7, predict the classification value using KNN with K=3.** **[4]**

X1 (acid durability)	X2 (strength)	Class	Distance² to (3, 7)	Rank
7	7	Bad	$(3-7)^2 + (7-7)^2 = 16$	3
7	4	Bad	16 + 9 = 25	4
3	4	Good	0 + 9 = 9	1
1	4	Good	4 + 9 = 13	2

The 3 nearest neighbours are Good, Good, Bad. Majority vote gives **Good**.

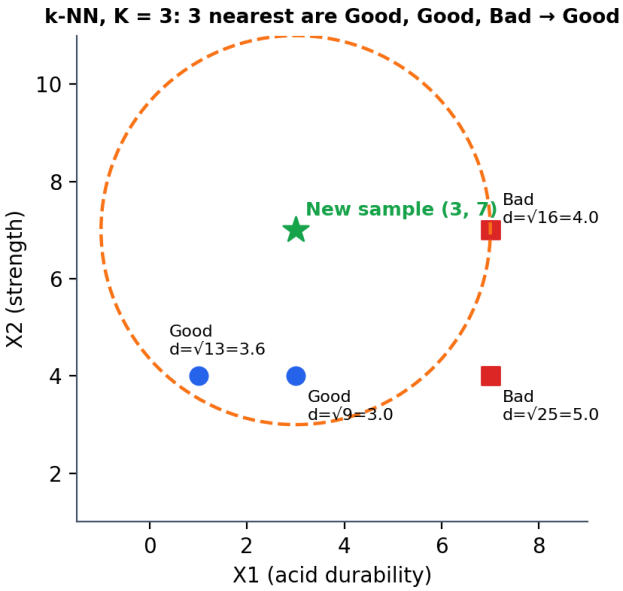

Figure: k-NN with K = 3 for the new sample (3, 7)

Q13 **Compare and contrast the Gini impurity and entropy as criteria for splitting nodes in a decision tree.** **[4]**

- **Gini = 1 − Σp_i^2** (range 0 to 0.5 for 2 classes). **Entropy = $-\Sigma p_i \log_2 p_i$** (range 0 to 1).
- Both are 0 for a pure node and maximum for a 50/50 node.
- Gini is faster (no logarithm) and is used by **CART**. Entropy (information gain) is used by **ID3/C4.5**.
- In practice both give very similar trees.

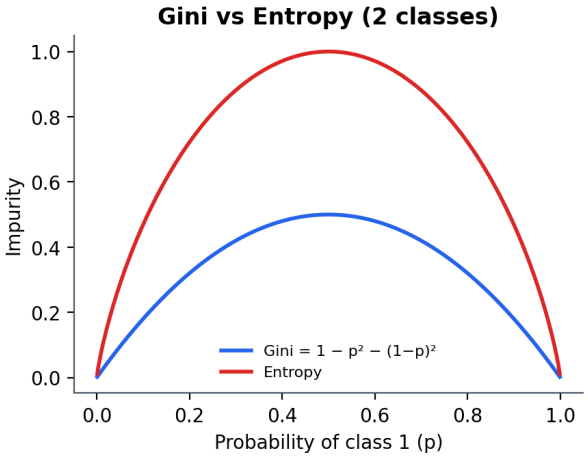

Figure: Gini and entropy against class probability

Q14 **For the given sports data (Age, Gender, Sportive), find the root node of a decision tree using CART. Consider age ≤ 25 and age > 25.** Also a PYQ: "explain CART with this example, find the Gini index, create the tree" (10 marks) **[4]**

13 rows: 7 Yes and 6 No. Gini(parent) = 1 − (7/13)² − (6/13)² = 0.497.

Split	Branch	Yes / No	Gini of branch	Weighted Gini
Gender	Female (7)	5 / 2	1 − (25+4)/49 = 0.408	(7/13)(0.408) + (6/13)(0.444) = **0.425**
	Male (6)	2 / 4	1 − (4+16)/36 = 0.444	
Age ≤ 25 / > 25	Age ≤ 25 (6)	5 / 1	1 − (25+1)/36 = 0.278	(6/13)(0.278) + (7/13)(0.408) = **0.348**
	Age > 25 (7)	2 / 5	1 − (4+25)/49 = 0.408	

Age has the lowest weighted Gini, so **the root node is Age**. Age ≤ 25 is mostly Yes, so it is split further by Gender (Female = Yes 4/4; Male = tie). Age > 25 is mostly No (5 of 7), so it becomes a leaf = No.

CART decision tree: sports data

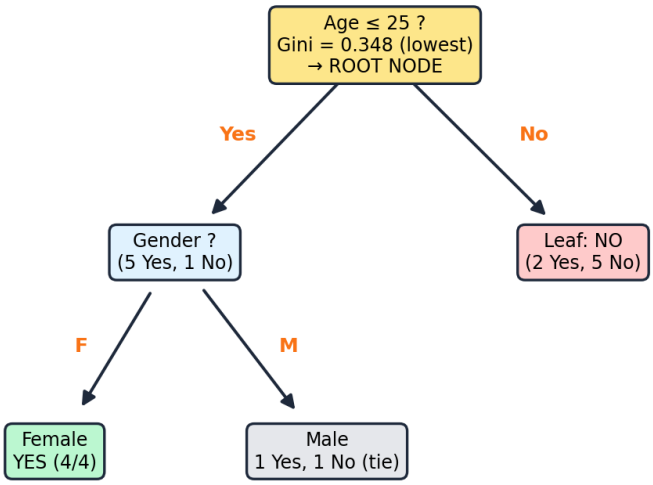

Figure: CART tree for the sports data

Q15 **For the given PlayTennis dataset (14 days), draw a decision tree using the Gini index.** **[4]**

14 examples: 9 Yes, 5 No. Weighted Gini of each attribute:

Attribute	Weighted Gini
Outlook	(5/14)(0.48) + (4/14)(0) + (5/14)(0.48) = **0.343** (lowest)
Humidity	0.367
Wind	0.429
Temperature	0.441

- **Outlook is the root.**
- **Overcast** (D3, D7, D12, D13): all Yes, so leaf = **Yes**.
- **Sunny** (2 Yes, 3 No): split on **Humidity**. High gives No (D1, D2, D8); Normal gives Yes (D9, D11).
- **Rain** (3 Yes, 2 No): split on **Wind**. Weak gives Yes (D4, D5, D10); Strong gives No (D6, D14).

Decision tree for PlayTennis (Gini index)

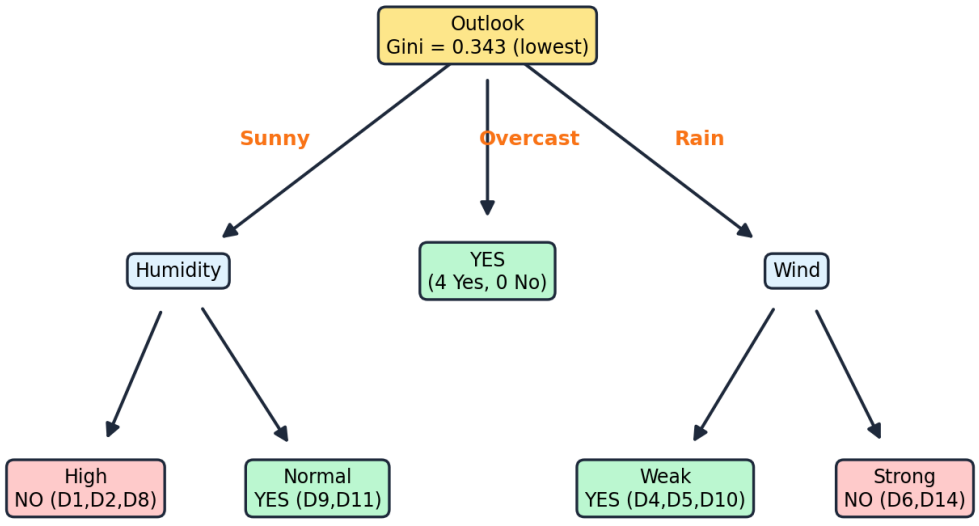

Figure: Final PlayTennis decision tree

Q16 **Discuss the theoretical foundations and practical applications of Support Vector Machines (SVM) in machine learning. Illustrate your answer with appropriate examples and diagrams where necessary.** **[4]**

- **Foundation:** maximum-margin hyperplane. Minimise $\frac{1}{2}\|w\|^2$ subject to $y_i(w \cdot x_i + b) \geq 1$, solved with Lagrange multipliers; only support vectors have non-zero multipliers.
- **Soft margin:** slack variables and parameter C allow some errors on noisy data.
- **Kernel trick:** a kernel (linear, polynomial, RBF) computes dot products in a higher-dimensional space, so non-linear data becomes separable without explicit mapping.

Applications: spam filtering, text/document classification, face detection, handwritten digit recognition, cancer classification from gene data, image classification.

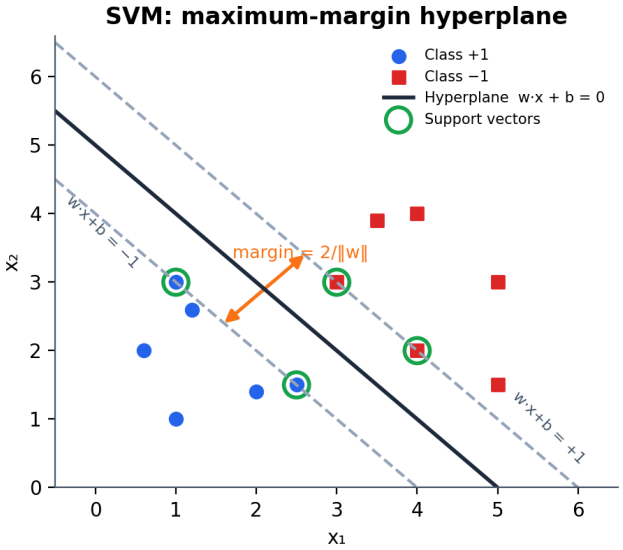

Figure: SVM hyperplane, margin (2/‖w‖) and support vectors (see also Q9 and Q10)

Q17 **Compare and contrast Bagging and Boosting in terms of their approach to handling model diversity. How does each method encourage diversity among base models, and how does this impact ensemble performance?** **[4]**

- **Bagging:** diversity comes from **random data subsets** (bootstrap; Random Forest also randomises features). Models are independent. Averaging cancels errors, so **variance falls**; robust to noise.
- **Boosting:** diversity comes from **re-weighting** hard examples so each model has a different emphasis. Models are dependent and complementary. This **reduces bias** and gives high accuracy, but can overfit noise and outliers.
- **Impact:** both beat a single model; diversity is what makes the ensemble better than its members. (Diagram: Q8.)

Q18 **Compare and contrast the strengths and weaknesses of classification algorithms like KNN, decision trees, and support vector machines. Under what circumstances would you prefer to use these alternatives?** **[4]**

	Strengths	Weaknesses	Prefer when
k-NN	Simple, no training, handles non-linear boundaries	Slow prediction, needs scaling, hurt by irrelevant features and high dimensions	Small data, few features, quick baseline
Decision Tree	Interpretable, handles numeric and categorical data, little preprocessing	Overfits, unstable	Explanation and rules are needed
SVM	Effective in high dimensions, strong with kernels, good generalisation (margin)	Slow on very large data, kernel and parameter tuning, hard to interpret	Medium data with many features (e.g. text)

Q19 **Design a scenario in which KNN can be applied to solve a practical problem. Specify the dataset characteristics, explain how you would preprocess the data, and outline the steps to choose the most suitable "k" value.** **[4]**

Problem: classify loan applicants as **risky / safe**.

- **Dataset:** labelled historical applicants; features such as age, income, credit score, loan amount, employment type; a few thousand rows.
- **Preprocessing:** remove duplicates, impute missing values, treat outliers, one-hot encode categoricals, **scale features** (min-max or standardisation; essential for distance-based methods), drop irrelevant features.
- **Choosing k:** split into train/validation/test; try odd k = 1, 3, 5 ... 25; use **5- or 10-fold cross-validation**; plot validation error against k and pick the lowest (the elbow); finally check on the test set.

Q20 **Design a scenario where an SVM can be applied to solve a practical problem. Outline the dataset requirements, preprocessing steps, kernel choice, and hyperparameter tuning process.** **[4]**

Problem: spam email detection.

- **Dataset needs:** thousands of labelled emails (spam/ham), reasonably balanced.
- **Preprocessing:** lowercase, remove punctuation and stop words, tokenise, convert to **TF-IDF** vectors, scale, split train/test.
- **Kernel: linear** (text is high-dimensional and sparse, usually linearly separable). Try RBF if the linear score is poor.
- **Tuning:** grid search (or random search) over C (and gamma for RBF) with 5-fold CV; pick the best F1/AUC; retrain on the full training set; evaluate on the test set.

Q21 **Suppose you have trained an SVM model for a classification task. Discuss the performance evaluation metrics you would use to assess the model's quality and provide guidelines on interpreting the results.** Also covers the PYQ "Define Precision and Recall" **[4]**

Build the **confusion matrix** (TP, FP, TN, FN), then compute:

- **Accuracy** = (TP+TN)/Total
- **Precision** = TP/(TP+FP): of all predicted positives, how many are truly positive.
- **Recall** = TP/(TP+FN): of all actual positives, how many were found.
- **F1** = 2PR/(P+R)
- **ROC-AUC:** area under the TPR vs FPR curve.

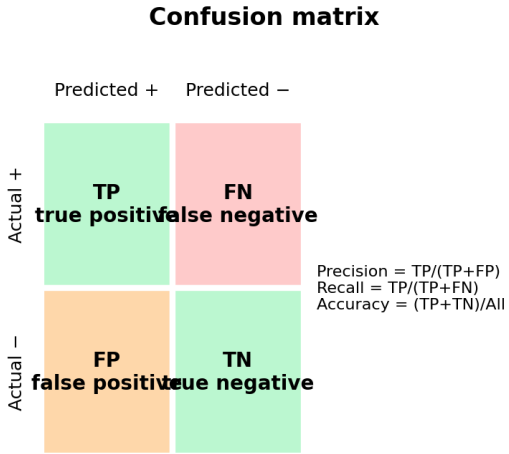

Figure: Confusion matrix and metric formulas

Interpreting: on imbalanced data trust F1/AUC more than accuracy. High precision with low recall means many positives are missed; the reverse means many false alarms. AUC near 1 is excellent, 0.5 is random guessing. A large gap between training and test scores signals overfitting. Use cross-validation for a stable estimate.

Unit 5: Optimization Techniques (Q22 to Q42)

Q22 **What is the difference between training error and validation error in model selection?** **[5]**

- **Training error:** error on the data the model was fitted on.
- **Validation error:** error on held-out data not used for fitting; it estimates generalisation and is used to choose models and hyperparameters.
- Low training error with high validation error means **overfitting**. Both high means **underfitting**.

Q23 **Provide a brief overview of why optimization is crucial in the context of training machine learning models.** **[5]**

Training a model means **minimising a loss function** by finding the best parameters (weights). Optimization decides how accurate the model becomes, how fast it converges and how much compute it costs. Example: gradient descent updates the weights to reduce the loss.

Q24 **What challenges can arise when applying optimization techniques to large-scale machine learning models?** **[5]**

- High computation and memory cost; data may not fit in memory (needs mini-batches or distributed training).
- **Non-convex** loss: local minima and saddle points.
- Vanishing or exploding gradients in deep networks.
- Sensitive to learning rate and other hyperparameters; slow convergence; high risk of overfitting.

Q25 **What is the bias-variance tradeoff, and how does it impact model selection?** Also covers PYQs: bias and variance, define variance, trade-off with diagram **[5]**

- **Bias:** error from overly simple assumptions (underfitting). **Variance:** error from sensitivity to the training data; the model fits noise (overfitting).
- Total error = Bias² + Variance + irreducible noise.
- More complexity lowers bias but raises variance; both cannot be minimised together.
- **Impact on model selection:** pick the complexity at the lowest validation error, using cross-validation, regularisation and ensembles (bagging reduces variance, boosting reduces bias).

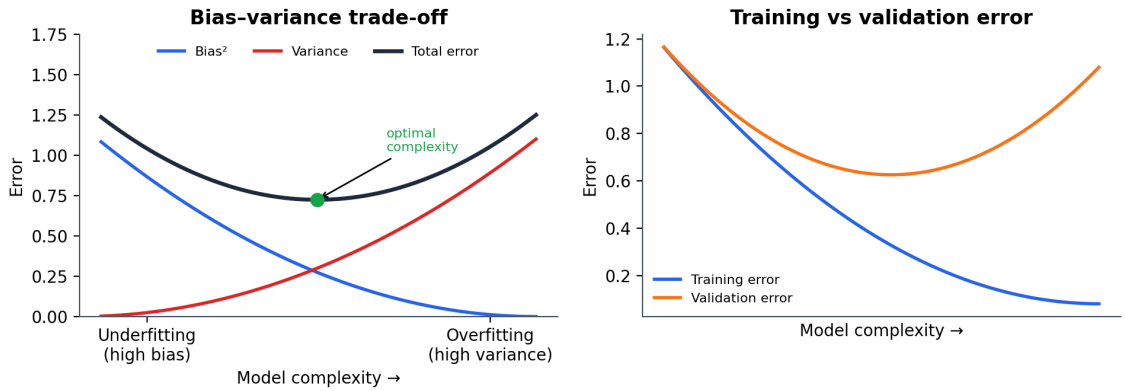

Figure: Bias–variance trade-off (left) and training vs validation error (right)

Q26 What is the validation set used for in hyperparameter tuning? [5]

It is used to try different hyperparameter values and compare models **without touching the test set**, so the test set remains an honest, unbiased final check. Typical split: 60/20/20 or 70/15/15.

Q27 How does cross-validation help in assessing the performance of a machine learning model? [5]

Data is split into k folds. The model trains on k−1 folds and tests on the remaining one, repeated k times; the scores are averaged. This is more reliable than one split, uses all data for both training and testing, and exposes overfitting. (Diagram: Q37.)

Q28 Differentiate between model selection and model evaluation. [5]

	Model selection	Model evaluation
Purpose	Choose the best model and hyperparameters among candidates	Measure the final chosen model's real performance
Data used	Validation set or cross-validation	Unseen **test** set
When	First	After selection

Q29 What are the advantages and disadvantages of using grid search for hyperparameter tuning? [5]

- **Advantages:** simple, exhaustive within the grid, reproducible, easy to run in parallel.
- **Disadvantages:** cost grows exponentially with the number of hyperparameters; wastes time on unimportant parameters; misses values between grid points; ranges must be chosen in advance. (Diagram: Q39.)

Q30 Discuss how Cross-Validation and Regularization techniques can be employed to address the issues of overfitting and underfitting? Also covers [5]

PYQs: what is regularization, how it works with an example, and types of regularization

Regularisation adds a **penalty on large weights** to the loss: Loss = Error + λ × Penalty. A larger λ gives a simpler model, so it cannot memorise noise.

- **L2 (Ridge):** penalty = Σw^2; shrinks weights towards zero.
- **L1 (Lasso):** penalty = $\Sigma |w|$; makes some weights exactly zero, so it also does feature selection.
- **Elastic Net:** a mix of L1 and L2.
- **Example:** a degree-9 polynomial fitted to 14 noisy points wiggles wildly (overfit). Adding an L2 penalty shrinks its coefficients and the curve becomes smooth; too large a λ flattens it (underfit).

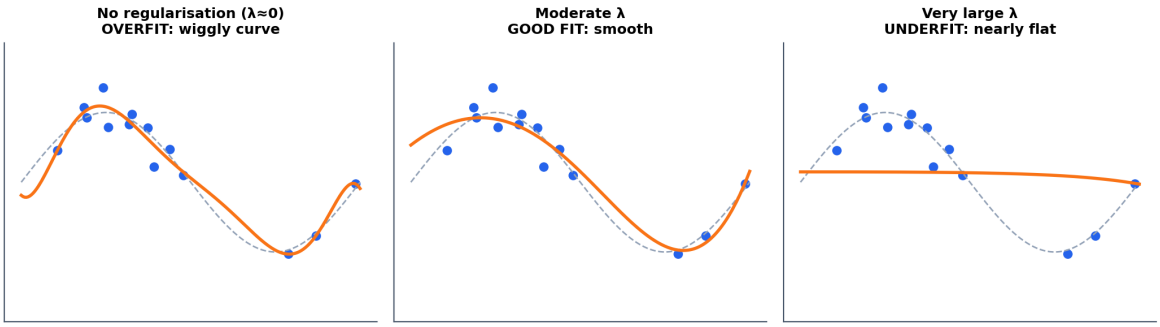

Figure: Effect of regularisation strength λ on a polynomial fit

Cross-validation detects overfitting (a large train-versus-validation gap) and helps choose λ, model complexity and k.

Problem	Fix
Overfitting	Increase λ, use CV to pick a simpler model, add data, prune, early stopping

Problem	Fix
Underfitting	Decrease λ, use a more complex model, add features

Q31 **Differentiate between local and global optimization strategies. How do these approaches impact the performance of machine learning algorithms?** **[5]**

- **Local:** finds the best solution in a neighbourhood of the starting point (e.g. gradient descent). Fast, but can get stuck in a local minimum.
- **Global:** searches for the overall best (simulated annealing, genetic algorithms, random restarts, Bayesian optimization). Slower but better for non-convex problems.
- For **convex** problems (linear regression, SVM) local = global. For neural networks (non-convex) the initialisation and restarts affect final accuracy.

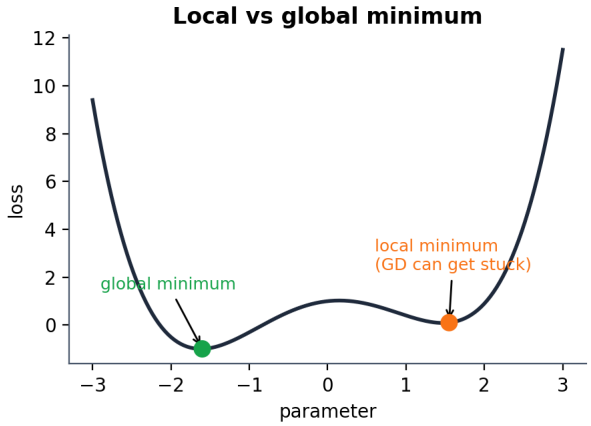

Figure: Local vs global minimum

Q32 **Compare and contrast stochastic gradient descent (SGD) with mini-batch gradient descent.** **[5]**

	SGD	Mini-batch GD
Samples per update	1	A small batch (e.g. 32 to 256)
Updates	Very noisy, zig-zag	Smoother
Speed	Fast per update	Uses vectorisation / GPU well
Convergence	Fluctuates around the minimum	More stable
Use	Online learning	Standard in deep learning

Q33 **Clarify the importance of model selection in machine learning. Provide three key factors that influence the decision-making process when selecting a model for a specific task.** **[5]**

Model selection picks the model that generalises best, which decides accuracy, cost and reliability. Three factors:

1. **Size and quality of data** (small data suits simple models; large data suits complex ones).
2. **Problem type and complexity** (classification or regression, linear or non-linear relationship).
3. **Interpretability and computational cost** (training time, memory, need to explain results).

Q34 **Explain how the gradient descent algorithm works and provide an example of a scenario where it can be applied.** **[5]**

Gradient descent minimises a loss J(w) by repeatedly moving the weights opposite to the gradient: **$w := w - \alpha \cdot \nabla J(w)$** (α = learning rate). Steps: initialise w, compute the gradient, update w, repeat until the loss stops decreasing.

Example: minimise $J = w^2$, start $w = 4$, $\alpha = 0.1$, gradient $= 2w$. $w_1 = 4 - 0.1(8) = 3.2$; $w_2 = 3.2 - 0.1(6.4) = 2.56$; ... it approaches the minimum at $w = 0$. **Scenario:** training linear regression to predict house prices by minimising MSE.

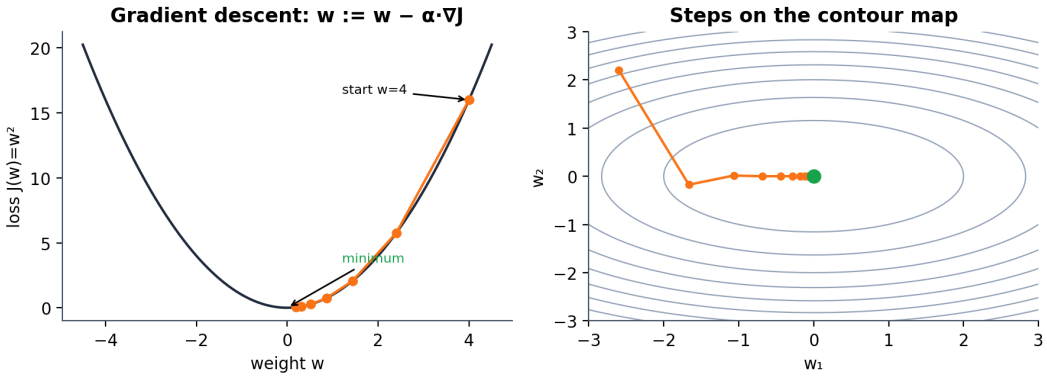

Figure: Gradient descent on a loss curve and on a contour map

Q35 **Explain the concept of hyperparameter tuning in machine learning. Why is it important, and how does it differ from model training?** **[5]**

- **Hyperparameters** are set *before* training (k in k-NN, C in SVM, learning rate, tree depth). **Parameters** (weights) are *learned during* training.
- **Tuning** = searching for the best hyperparameter values (grid search, random search, Bayesian optimization) using validation or CV scores.
- **Why important:** the same algorithm can underfit or overfit depending on them.
- **Difference:** training learns parameters from data by optimisation; tuning is an outer loop that retrains many times with different settings and keeps the best.

Q36 **Differentiate between types of gradient descent algorithm with an example.** **[5]**

Type	Data per update	Pros	Cons
Batch	Whole dataset	Stable, smooth	Slow, high memory for big data
Stochastic (SGD)	1 sample	Fast, can escape shallow minima	Very noisy
Mini-batch	A batch (e.g. 100)	Balanced, GPU friendly	Needs batch-size tuning

Example: with 1000 samples and batch size 100, one epoch gives 1 update (batch), 1000 updates (SGD) or 10 updates (mini-batch).

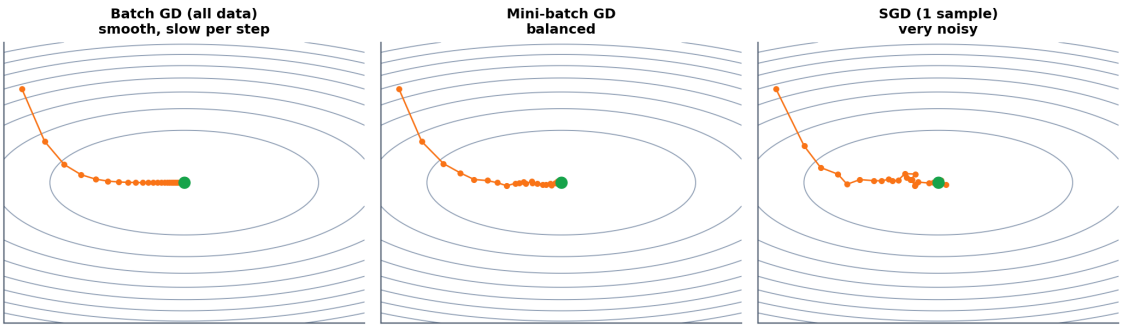

Figure: Typical paths of batch, mini-batch and stochastic gradient descent

Q37 **Describe the concept of cross-validation in machine learning. Explain how k-fold cross-validation works, and highlight its significance in assessing and improving the performance of machine learning models.** Also covers the PYQ "Infer K-fold cross validation with respect to overfitting" **[5]**

Cross-validation measures how well a model generalises to unseen data. **k-fold steps:**

1. Shuffle the data and split it into k equal folds (e.g. k = 5).

2. Train on k−1 folds, test on the remaining fold.
3. Repeat k times so every fold is the test fold once.
4. Average the k scores.

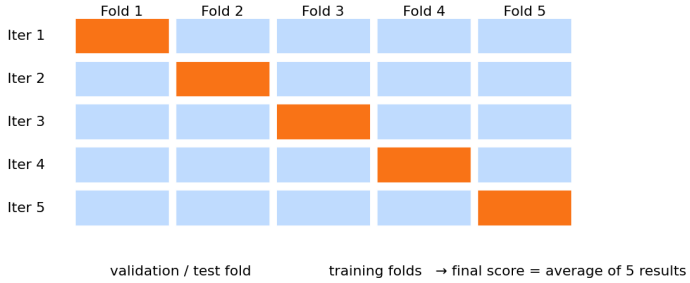

Figure: 5-fold cross-validation

Significance: a less biased estimate than one split; every point is used for training and testing; good for small datasets; detects overfitting (a big spread across folds or a large train-test gap); used to tune hyperparameters. Variants: stratified k-fold, leave-one-out.

Q38 **Describe the algorithm of the Steepest Descent method in detail by considering one example.** **[5]**

Steepest descent moves along the **negative gradient** (the direction of fastest decrease), choosing the step size α by a **line search**: $x_{k+1} = x_k - \alpha_k \nabla f(x_k)$, where α_k minimises $f(x_k - \alpha \nabla f(x_k))$.

Algorithm: start at x_0; compute ∇f; find the best α along −∇f; update; stop when ‖∇f‖ is small.

Example: $f(x, y) = x^2 + y^2$, $x_0 = (2, 2)$.

- $\nabla f = (2x, 2y) = (4, 4)$.
- Along the line: $f(2 - 4\alpha, 2 - 4\alpha) = 2(2 - 4\alpha)^2$, minimised at $\alpha = 0.5$.
- $x_1 = (2 - 2, 2 - 2) =$ **(0, 0)**, where $\nabla f = 0$. It converges to the minimum in one step.

Q39 **Discuss the importance of model performance metrics in evaluating different hyperparameter settings. Evaluate how Random Search can be a more efficient alternative to Grid Search, including its advantages and potential drawbacks.** **[5]**

Metrics: each setting is compared using a validation or CV metric (accuracy, F1, AUC for classification; RMSE, R^2 for regression). The metric must match the goal (e.g. recall for disease detection).

- **Random search** samples random combinations from the ranges instead of trying all.
- **Advantages:** more efficient, since the same budget tries many more distinct values of each important hyperparameter; you control the number of trials; handles continuous ranges; easy to parallelise.
- **Drawbacks:** no guarantee of finding the best setting; results vary with the seed; may miss a narrow good region.

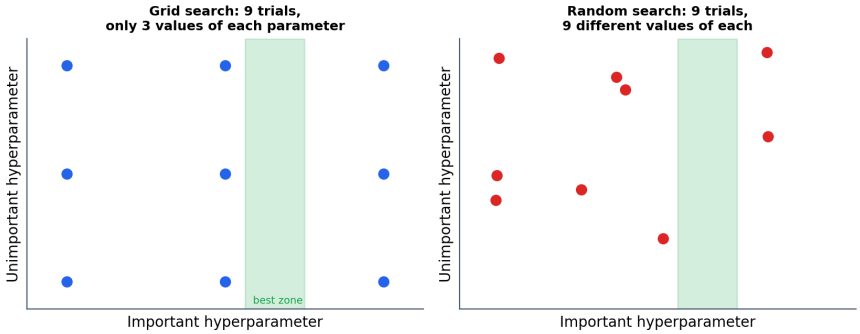

Figure: Grid search vs random search with 9 trials each

Q40	**Discuss the steps taken to select a machine learning model for a classification problem. Include the functions of model training, validation, and testing. How does each step help to ensure that the selected model performs effectively on new data?**	[5]

1. Define the problem and choose the metric (F1, AUC, etc.).
2. Prepare the data and split into **train / validation / test**.
3. **Training set:** fit several candidate models (k-NN, tree, SVM, ...).
4. **Validation set (or CV):** compare candidates and tune hyperparameters; pick the best.
5. **Test set:** evaluate the chosen model **once**, giving an unbiased estimate on new data.
6. Retrain on all training data, deploy and monitor.

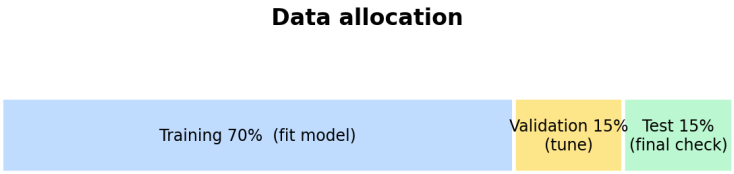

Figure: Typical train / validation / test allocation

Each step keeps the evaluation honest and guards against overfitting and data leakage.

Q41	**Discuss the steps for defining the search space, assessing performance, and selecting the best hyperparameters while designing a Random Search-based hypertuning strategy.**	[5]

1. **Define the search space:** the hyperparameters and their ranges or distributions (e.g. C log-uniform 0.001 to 1000, gamma log-uniform, tree depth integer 2 to 20).
2. **Set the budget:** the number of random trials n.
3. **Sample** n random combinations.
4. **Assess performance:** train each with k-fold CV and record the chosen metric.
5. **Select** the combination with the best mean CV score (optionally narrow the ranges around it and search again).
6. Retrain with the best settings on the whole training set and report on the test set.

Q42	**Discuss how the learning rate impacts the convergence of Gradient Descent. Discuss the differences between Adaptive Gradient Descent methods (such as AdaGrad, RMSprop, and Adam) and traditional Gradient Descent. Include the advantages and limitations of these adaptive methods.**	[5]

Learning rate (α): too small gives very slow convergence; too large overshoots, oscillates or diverges; a well-chosen α converges quickly and steadily. Decay schedules (shrinking α over time) help.

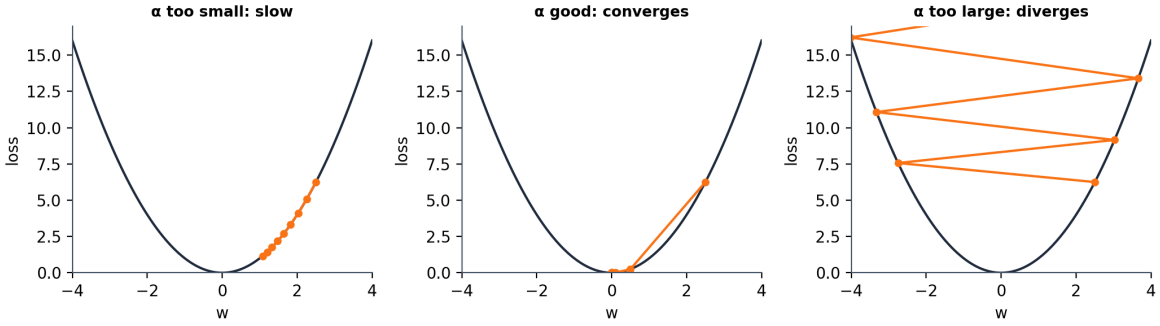

Figure: Effect of the learning rate on convergence (loss = w²)

Traditional GD uses one fixed α for all parameters and needs manual tuning. **Adaptive methods** adjust the rate per parameter:

- **AdaGrad:** divides α by the root of the sum of past squared gradients. Good for sparse features, but α keeps shrinking and learning can stall.
- **RMSprop:** uses a moving average of squared gradients, fixing AdaGrad's vanishing rate.
- **Adam:** momentum + RMSprop with bias correction (defaults $\alpha = 0.001$, $\beta_1 = 0.9$, $\beta_2 = 0.999$). Fast and widely used.

Advantages: less manual tuning, faster convergence, copes with sparse or noisy gradients. **Limitations:** extra memory and hyperparameters, and Adam can sometimes generalise worse than well-tuned SGD with momentum.

Unit 6: Semi-Supervised and Reinforcement Learning (Q43 to Q63)

Your PDF repeats these 21 questions in a second table (Sr. No. 1 to 21). They are answered once here as Q43 to Q63.

Q43 **Define semi-supervised learning in the context of machine learning.** **[6]**

A learning approach that uses a **small amount of labelled data together with a large amount of unlabelled data** to train a model. It sits between supervised and unsupervised learning.

Q44 **How does semi-supervised learning leverage both labeled and unlabeled data in training?** **[6]**

1. Train an initial model on the small labelled set.
2. Use it to predict labels for the unlabelled data (**pseudo-labels**) and keep the confident ones (self-training).
3. Retrain on labelled + pseudo-labelled data and repeat.

- Other techniques: **co-training** (two models label data for each other), **label propagation** (labels spread over a similarity graph), **consistency regularisation** (predictions stay the same under small input changes).
- The unlabelled data reveals the *shape* of the data (clusters), which helps place the decision boundary.

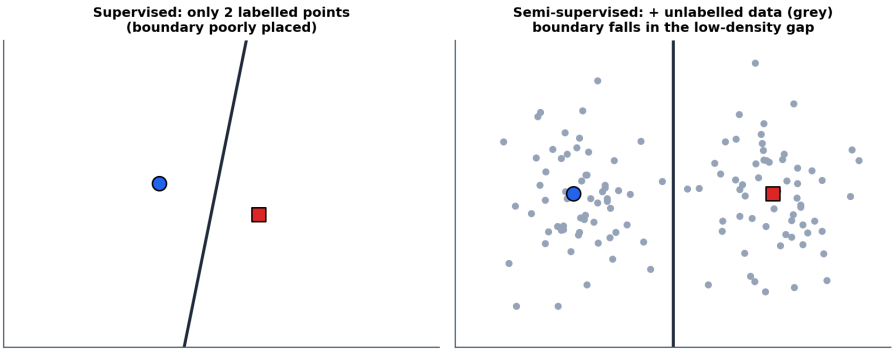

Figure: Unlabelled data helps place the boundary in the low-density gap

Q45 **Define reinforcement learning.** Also a PYQ (2 marks) **[6]**

A learning paradigm in which an **agent** learns by interacting with an **environment**, taking **actions** and receiving **rewards or penalties**, so as to maximise the total cumulative reward over time. It learns by trial and error with no labelled data.

Q46 **Analyze the potential risks or challenges of using semi-supervised learning in situations with limited labeled data.** **[6]**

- **Wrong pseudo-labels** are reinforced again and again (confirmation bias), so errors snowball.
- Unlabelled data may come from a **different distribution** than the labelled data.
- It relies on assumptions (smoothness, cluster, manifold) that may not hold.

- A tiny labelled set may not represent all classes; imbalance gets worse.
- Hard to validate (few labels to test with); more computation than supervised learning.

Q47	**State and explain the advantages of semi-supervised learning.**	[6]

- Cuts the cost and time of labelling.
- Better accuracy than supervised learning on only a few labels.
- Uses the abundant unlabelled data.
- Better generalisation; works well where labels are scarce.

Q48	**Provide examples of real-world applications where reinforcement learning has been successfully employed.**	[6]

- **Games:** AlphaGo (Go), AlphaZero (chess), Atari games.
- **Robotics:** grasping, walking. **Autonomous driving:** lane changing, parking.
- **Recommendation systems** and ad placement.
- **Resource management:** data-centre cooling, traffic-signal control.
- **Finance** (trading), healthcare (treatment planning), and **RLHF** to tune chat models.

Q49	**Describe two key benefits of using reinforcement in machine learning applications.**	[6]

1. It learns **without labelled data**, directly from interaction.
2. It optimises **long-term cumulative reward**, so it handles sequential decisions and adapts to changing environments (and can find strategies humans did not think of).

Q50	**Discuss the challenges of semi-supervised learning in machine learning.**	[6]

Same core issues as Q46: wrong pseudo-labels, distribution mismatch between labelled and unlabelled data, assumptions that may fail, class imbalance and difficult evaluation. Add: sensitivity to *which* few samples are labelled, and higher training cost.

Q51	**Provide an example of a real-world problem where semi-supervised learning would be beneficial.**	[6]

Medical image diagnosis: hospitals hold millions of X-rays but only a few hundred are labelled by radiologists (expensive and slow). A semi-supervised model learns from the labelled scans plus the unlabelled bulk to reach good accuracy. Others: speech recognition, web-page classification, fraud detection.

Q52 / Q62	**Analyze the benefits of using semi-supervised learning techniques, especially in scenarios with limited labeled data. How does leveraging unlabeled data enhance the accuracy and efficiency of machine learning models?** Q62 asks the same question; one answer covers both	[6]

- **Accuracy:** unlabelled data shows the true structure (clusters, density), so the decision boundary is placed in low-density gaps and not just between the few labelled points. This usually beats a purely supervised model.
- **Efficiency:** far fewer labels are needed, saving labelling cost and time.
- **Generalisation:** the model sees more variety, so it overfits less.
- **Scalability:** easy to keep improving as more unlabelled data arrives.

Q53 **Discuss the application of Autonomous Vehicles and how to improve decision-making processes for self-driving cars.** **[6]**

As an RL problem: agent = the car; **environment** = road, traffic, pedestrians; **state** = camera, LiDAR and radar readings plus speed; **actions** = steer, accelerate, brake; **reward** = positive for safe progress and smooth driving, large negative for collisions or rule violations.

- Train deep RL (DQN, PPO) in **simulation** first, then in the real world.
- Careful reward design and safety constraints.
- Sensor fusion for a better state estimate.
- Combine with imitation learning from human drivers.
- Challenges: safety, rare events, the simulation-to-reality gap.

Q54 **Explain the key elements of reinforcement learning, including agents, environments, actions, and rewards. How do these elements interact within the reinforcement learning framework?** Also covers the PYQ "key elements of RL and the workflow" **[6]**

- **Agent:** the learner / decision maker. **Environment:** everything the agent interacts with.
- **State (s):** the current situation. **Action (a):** a choice the agent can make. **Reward (r):** numeric feedback.
- **Policy (π):** the strategy mapping states to actions. **Value function:** expected future reward.

Workflow: the agent observes state s_t, chooses action a_t; the environment returns reward r_{t+1} and next state s_{t+1}; the agent updates its policy to maximise cumulative reward. This repeats over many episodes.

Reinforcement learning loop

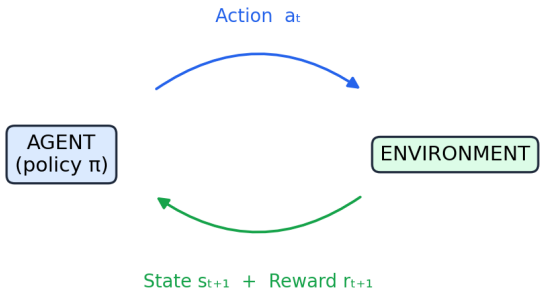

Figure: The reinforcement learning interaction loop

Q55 **Justify with the following diagram that a Markov Process is a memoryless random process.** **[6]**

A Markov process obeys **P(next state | current state, all past states) = P(next state | current state)**. The next state depends only on the present state, not on how we got there. Transition probabilities from the diagram (each row sums to 1):

From \ To	Sleep	Ice Cream	Run
Sleep	0.2	0.2	0.6
Ice Cream	0.2	0.1	0.7
Run	0.1	0.3	0.6

Justification: from **Run**, the chance of next going to Ice Cream is always **0.3**, whether we reached Run from Sleep, from Ice Cream or from Run itself. The history never changes these numbers, so the process is memoryless.

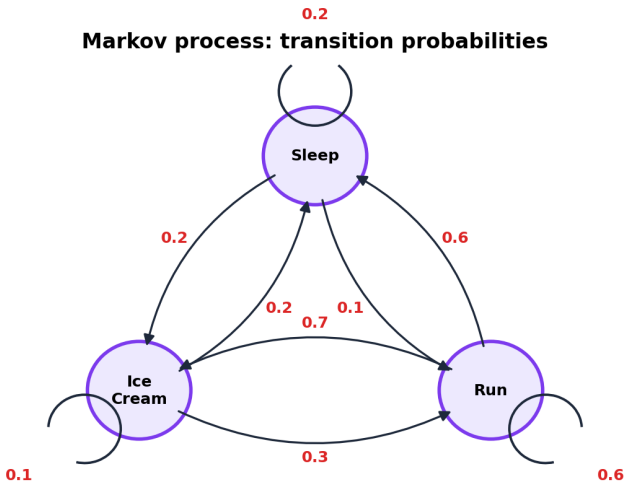

Figure: Markov process (Sleep, Ice Cream, Run) with transition probabilities

Q56 **For the given Markov Decision Process calculate the rewards for (a) start state Class 1, move to Class 2, then Class 3, then Pass, then Sleep; (b) start at Pub and go to Sleep through Class 2.** **[6]**

The Pub reward is not labelled in your image. I assumed the standard Student MDP value +1. If your sheet says otherwise, replace it (for part b the answer becomes −2 if the Pub reward is 0).

Rewards: Class 1 = −2, Class 2 = −2, Class 3 = −2, Facebook = −1, Pass = +10, Sleep = 0, Pub = +1. Return = sum of the rewards of the states visited (no discounting).

- **(a)** Class 1 → Class 2 → Class 3 → Pass → Sleep: (−2) + (−2) + (−2) + 10 + 0 = **4**.
- **(b)** Pub → Class 2 → Sleep: (+1) + (−2) + 0 = **−1**. Probability of this path = 0.4 × 0.2 = 0.08.

Student Markov Decision Process (rewards on states)

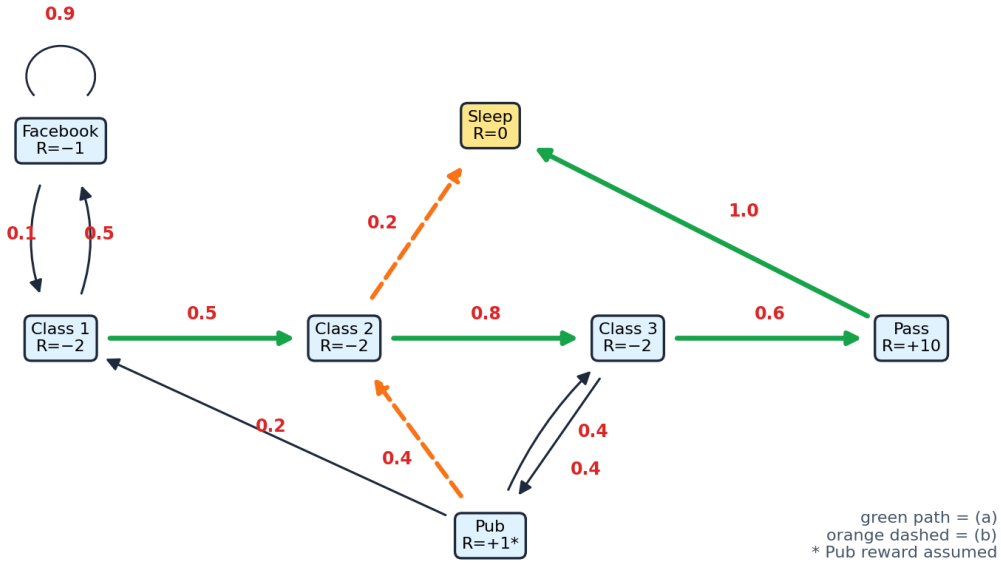

Figure: Student MDP. Green = path (a); orange dashed = path (b)

Q57 **What is Reinforcement Learning? How does it differ from other types of Machine Learning, such as Supervised Learning and unsupervised learning?** **[6]**

RL = an agent learns the best actions by trial and error to maximise cumulative reward.

	Supervised	Unsupervised	Reinforcement
Data	Labelled (input, output)	Unlabelled	No fixed dataset; learns from interaction
Feedback	Correct answer given	None	Reward / penalty, often delayed
Goal	Predict labels	Find structure / clusters	Maximise long-term reward
Example	Spam detection	Customer segmentation	Game-playing agent, robot

Q58 **How is Reinforcement Learning utilized in game AI? Provide an example of a popular game where Reinforcement Learning has been applied.** Also covers the PYQ "Explain how a system can play chess using RL" **[6]**

In a game the **agent** is the player, the **environment** is the game, the **state** is the board or screen, **actions** are legal moves and the **reward** is win (+1), loss (−1), draw (0) or the score. The agent plays many games (often **self-play**) and learns which moves lead to winning.

- **Popular examples:** AlphaGo (Go), AlphaZero (chess, shogi, Go), DeepMind's DQN on Atari games.
- **Chess:** state = board position, action = a legal move, reward at the end of the game. Through thousands of self-play games the system learns a *value* for positions and a *policy* for moves, improving until it plays at a very high level.

Q59 **Explain the concept of the reward function in Reinforcement Learning. Why is it important for the learning process? Evaluate the limitations and challenges faced by reinforcement learning methods. Propose potential solutions.** **[6]**

The **reward function** defines the goal: it gives the agent a numeric score for each action or state. The agent learns by maximising the total (discounted) reward, so good reward design is essential; a badly designed one makes the agent learn the wrong behaviour.

Challenge	Solution
Sparse or delayed rewards (credit assignment)	Reward shaping, discount factor, temporal-difference learning
Reward hacking (agent exploits loopholes)	Careful reward design, human feedback, inverse RL
Exploration vs exploitation	ε-greedy, UCB
Sample inefficiency (needs many trials)	Simulation, experience replay, model-based RL
Safety during learning	Constrained RL, safe exploration

Q60 **Explain the basic idea behind Q-learning as a Reinforcement Learning algorithm. How does it help an agent learn optimal actions?** **[6]**

Q-learning is a **model-free, off-policy** RL algorithm. It learns a **Q-table**: Q(s, a) = the expected total future reward of taking action a in state s and acting well afterwards.

```
Q(s,a) ← Q(s,a) + α [ r + γ · max Q(s',a') − Q(s,a) ]
(α = learning rate, γ = discount factor)
```

How it finds optimal actions: the agent explores with ε-greedy, updates Q after each step, and over many episodes Q converges to Q*. The optimal policy is then π(s) = argmax Q(s, a). **Mini example:** Q = 0, α = 0.5, r = 10, γ = 0.9, max Q(s') = 0, so Q ← 0 + 0.5(10 + 0 − 0) = **5**.

Q61 **Elaborate the application of Reinforcement Learning in robotics. How is it used to improve robotic behaviors and optimize decision-making?** **[6]**

- **Setup:** state = sensor readings and joint angles; actions = motor torques or movements; reward = reaching the goal or a stable walk, with penalties for falling or wasting energy.
- **Uses:** robot-arm grasping, legged walking, drones, warehouse robots.

- **How it improves behaviour:** the robot practises by trial and error (often in simulation, then transferred to the real robot) and learns a policy that maximises long-term reward, so decisions improve with experience and adapt to new conditions. Algorithms: DDPG, PPO, SAC.
- **Challenges:** many trials needed, safety, simulation-to-real gap.

Q63 **Analyze specific use cases of reinforcement learning in different domains, such as robotics, gaming, or autonomous systems. How does reinforcement learning optimize decision-making in these applications?** **[6]**

- **Robotics:** learns grasping and walking policies (Q61).
- **Gaming:** learns winning strategies via self-play (Q58).
- **Autonomous systems:** self-driving cars, drones and traffic control learn safe, efficient control (Q53).
- **How it optimises decisions:** in every domain the agent learns a **policy that maximises cumulative reward**, weighing immediate gains against long-term outcomes. It improves through repeated interaction and adapts when the environment changes.

Part B: PYQs Not Covered in the Question Bank

Papers are labelled by upload order: **Paper A** (photos 1 and 2), **Paper B** (photo 3), **Paper C** (photos 4 and 5). Section I = 2 marks each, Section II = 5 marks, Section III = 10 marks. Only PYQs **not** already answered in the question bank are answered below; the table says where the others are answered.

Where the already-covered PYQs are answered

PYQ topic (and paper)	Answered in
Bias and variance (A-I-1, B-I-3); define variance (C-I-3); bias-variance trade-off with diagram (C-II-4)	**Q25**
Hyperplane and support vectors (A-I-4); SVM with a use case (A-III-4); SVM margin and optimal hyperplane (B-III-1)	**Q1, Q9, Q16, Q20**
Define reinforcement learning (A-I-6)	**Q45**
Regularization: what, how it works, example (A-II-4, C-II-5); types of regularization (B-II-4)	**Q30**
Bagging with example (A-II-5); Boosting vs Bagging (B-II-2, C-II-3)	**Q4, Q7, Q8, Q17**
Hyperparameter tuning and strategies (A-II-6)	**Q35, Q29, Q39, Q41**
CART with example, Gini index and decision tree (A-III-3) (same sports data)	**Q14** (also Q3, Q13)
Gradient descent and its types (A-III-5)	**Q34, Q36**
Key elements and workflow of RL (A-III-6)	**Q54**
Precision and recall (B-I-4)	**Q21**
Pruning in decision trees (B-I-5)	**Q5**
How a system can play chess using RL (B-I-2)	**Q58**
KNN classification and why it is lazy (B-II-3)	**Q2, Q6**
K-fold cross-validation and overfitting (C-II-6)	**Q27, Q30, Q37**

Unit 1 and 2: Introduction and Data Handling

B1 **Why is data in the real world dirty?** Paper A, Section I **[2]**

Real data is collected from many sources by people and machines, so it is rarely perfect. Common problems: **missing values**, **noise and outliers**, **duplicates**, **inconsistent formats or units** (e.g. “M” vs “Male”, kg vs lb), **wrong entries**, and irrelevant or imbalanced data. Causes: human entry errors, sensor failure, merging different systems, no validation checks. It must be cleaned (preprocessing) or the model gives poor results (garbage in, garbage out).

B2 **What is Type 1 and Type 2 error?** Paper A, Section I **[2]**

- **Type 1 error (false positive):** rejecting a true null hypothesis; predicting positive when the truth is negative. Probability = α. Example: a healthy person is told they have a disease.
- **Type 2 error (false negative):** failing to reject a false null hypothesis; predicting negative when the truth is positive. Probability = β. Example: a sick person is told they are healthy.

B3 **Use the FIND-S algorithm to find the most specific hypothesis for the given data (Time, Weather, Temperature, Company, Humidity, Wind, Goes).** Paper A, Section II **[5]**

FIND-S uses only **positive** examples. Start with the most specific hypothesis h = ⟨∅, ∅, ∅, ∅, ∅, ∅⟩. For each positive example keep an attribute value if it matches; otherwise generalise it to “?”. Negative examples are ignored.

Example	Class	Hypothesis h after this example
1: Morning, Sunny, Warm, Yes, Mild, Strong	Yes	⟨Morning, Sunny, Warm, Yes, Mild, Strong⟩
2: Evening, Rainy, Cold, No, Mild, Normal	No	unchanged (negative ignored)
3: Morning, Sunny, Moderate, Yes, Normal, Normal	Yes	⟨Morning, Sunny, ?, Yes, ?, ?⟩ (Temperature, Humidity, Wind differ)
4: Evening, Sunny, Cold, Yes, High, Strong	Yes	⟨?, Sunny, ?, Yes, ?, ?⟩ (Time differs)

Most specific hypothesis = ⟨?, Sunny, ?, Yes, ?, ?⟩ (goes out when the weather is sunny and company is available).

B4 **Find Q1, Q2 and Q3 for the data {5, 40, 42, 46, 48, 49, 50, 50, 52, 53, 55, 56, 58, 75, 102}. Identify any outliers. Design a box plot.** Paper A, Section II **[5]**

- 15 values, already sorted. **Q2 (median)** = 8th value = **50**.
- **Q1** = median of the lower half (5, 40, 42, 46, 48, 49, 50) = **46**. **Q3** = median of the upper half (52, 53, 55, 56, 58, 75, 102) = **56**.
- IQR = 56 − 46 = 10. Lower fence = 46 − 1.5×10 = **31**; upper fence = 56 + 1.5×10 = **71**.
- **Outliers:** values below 31 or above 71, so **5, 75 and 102**. Whiskers run from 40 (smallest non-outlier) to 58 (largest non-outlier).

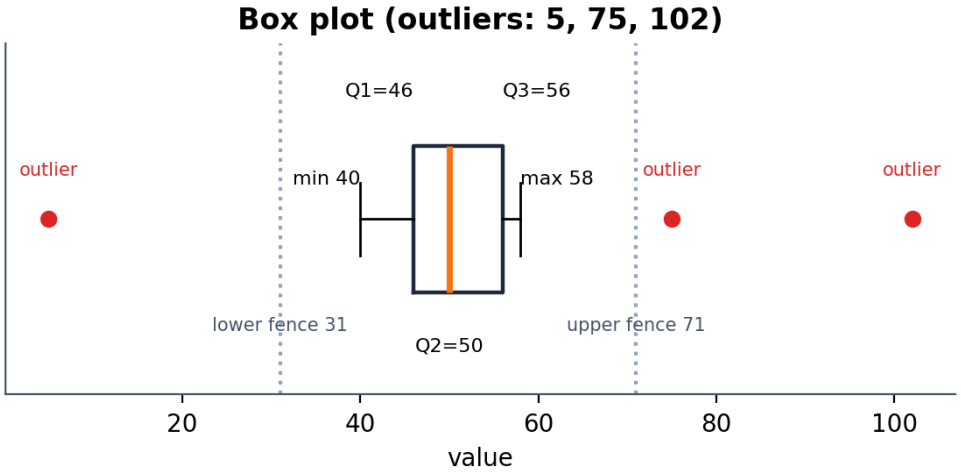

Figure: Box plot with fences and outliers

B5 **Explain different data transformation methods with examples.** Paper A, Section II **[5]**

- **Min-max normalisation:** x′ = (x − min)/(max − min). Example: 10, 20, 50 becomes 0, 0.25, 1.
- **Standardisation (z-score):** x′ = (x − mean)/std; gives mean 0 and std 1.
- **Log transformation:** reduces skew and outlier effect. Example: 10, 100, 1000 becomes 1, 2, 3.
- **Square-root / power (Box-Cox):** also reduce skewness.
- **Binning:** convert ages into groups such as 0-18, 19-40, 41+.
- **Encoding categorical data:** label encoding (Red = 0, Blue = 1) or one-hot encoding (Red = 100, Blue = 010).
- **Capping (winsorisation):** limit extreme values to a boundary.

B6 **Discuss the advantages and disadvantages of the Expectation-Maximization algorithm.** Paper A, Section II **[5]**

EM finds maximum-likelihood estimates when data has **hidden (latent) variables** or missing values, e.g. Gaussian Mixture Models. **E-step:** estimate the hidden variables using the current parameters. **M-step:** update the parameters to maximise the likelihood. Repeat until convergence.

- **Advantages:** handles missing and latent data; the likelihood never decreases per iteration; simple closed-form updates in many models; gives **soft** (probabilistic) cluster membership.
- **Disadvantages:** converges only to a **local** optimum and depends on initial values; can be slow; the number of components must be chosen; assumes a distribution form; can fail (singularities) on small data; costly for large data.

B7 **Enumerate different feature selection techniques in machine learning.** [10]
Paper A, Section III

Feature selection chooses the most useful input features, giving a simpler, faster, more accurate model that overfits less.

Method	How it works	Examples
Filter	Uses statistics before training; fast and model-independent	Variance thresholding (drop near-constant features), correlation analysis (keep features correlated with the target, drop mutually correlated ones), chi-square, ANOVA, information gain
Wrapper	Trains the model repeatedly on feature subsets; accurate but costly	Forward selection (add best features one by one), backward elimination (remove weakest one by one), recursive feature elimination (RFE)
Embedded	Selection happens during training; good balance of speed and accuracy	LASSO (L1 shrinks some coefficients to zero), Ridge, tree-based feature importance

B8 **Evaluate different evaluation metrics for regression problems. Also write the formulas for MSE, RMSE, MAE, R^2 and adjusted R^2.** Paper A, Section III and Paper B, Section I [10]

Let y = actual, $\hat{y}$ = predicted, n = number of samples, k = number of predictors.

Metric	Formula	Meaning
MAE	$(1/n)\ \Sigma\ \lvert y - \hat{y}\rvert$	Average absolute error; easy to read, robust to outliers
MSE	$(1/n)\ \Sigma\ (y - \hat{y})^2$	Squares errors, so it punishes large errors heavily
RMSE	$\sqrt{\text{MSE}}$	Error in the same unit as y
R^2	$1 - SS_{res}/SS_{tot} = 1 - \Sigma(y - \hat{y})^2 / \Sigma(y - \bar{y})^2$	Fraction of variance explained (1 = perfect, 0 = no better than the mean)
Adjusted R^2	$1 - [(1 - R^2)(n - 1)/(n - k - 1)]$	R^2 corrected for the number of features; it falls if a useless feature is added

B9 **When will you use classification over regression?** Paper B, Section I [2]

Use **classification** when the output is a **category or class** (spam / not spam, disease yes / no, cat / dog / bird). Use **regression** when the output is a **continuous number** (house price, temperature). Example: predicting *whether* a loan defaults is classification; predicting *how much* a customer will spend is regression.

B10 **What is a "training set" and "test set" in a machine learning model? How much data will you allocate for your training, validation and test sets?** [5]
Paper B, Section II

- **Training set:** the data the model learns from. **Validation set:** used to tune hyperparameters and choose a model. **Test set:** unseen data used once at the end to measure real performance.
- **Allocation:** commonly **70 / 15 / 15** or **60 / 20 / 20** (train / validation / test). With k-fold CV, 80 / 20 (train / test) is common. With very large datasets even 98 / 1 / 1 is fine. Shuffle first and use stratified splits for imbalanced classes.

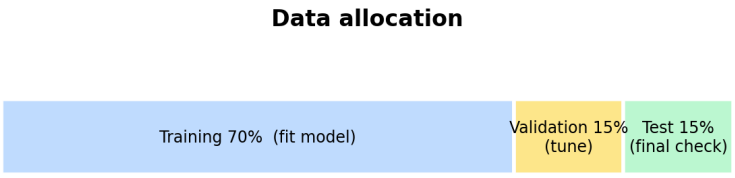

Figure: Data allocation

B11 **Define supervised learning and give two examples.** Paper C, Section I **[2]**

The model learns from **labelled data** (input-output pairs) and then predicts the output for new inputs. Examples: **spam detection** (classification) and **house price prediction** (regression).

B12 **Summarize dimensionality reduction.** Paper C, Section I **[2]**

Reducing the number of input features while keeping most of the important information. **Why:** avoids the curse of dimensionality, reduces storage and training time, lowers overfitting, removes noise and helps visualisation. **Types: feature selection** (keep a subset) and **feature extraction** (create new features, e.g. **PCA**, **LDA**, SVD).

B13 **List down the steps in developing ML applications.** Paper C, Section I **[2]**

1. Problem definition
2. Data collection
3. Data preprocessing
4. Feature engineering
5. Dataset splitting
6. Model selection
7. Model training
8. Model evaluation
9. Hyperparameter tuning
10. Model deployment
11. Monitoring and maintenance

B14 **What is Version space? Give an explanation.** Paper C, Section I **[2]**

The **version space** is the set of **all hypotheses consistent with the training examples** (those that classify every training example correctly). It is described by two boundaries: the **specific boundary S** (most specific consistent hypotheses) and the **general boundary G** (most general consistent hypotheses). Every hypothesis between S and G belongs to it. The Candidate Elimination algorithm (B25) shrinks the version space as each example arrives.

B15 **Outline the use cases of Linear Regression.** Paper C, Section I **[2]**

House price prediction, sales and demand forecasting, salary from experience, marks from study hours, stock or revenue trend estimation, temperature forecasting, insurance risk, and marketing spend versus revenue.

B16 **List down the disadvantages of Linear Regression.** Paper A, Section I **[2]**

- Assumes a **linear** relationship, so it cannot capture complex patterns.
- **Sensitive to outliers.**
- Suffers from **multicollinearity** between features.
- Depends on assumptions (independent errors, constant variance, normal errors).
- Overfits with many features; extrapolates poorly beyond the data range; numeric targets only.

B17 **Analyze the assumptions of Linear regression.** Paper C, Section III **[10]**

1. **Linearity:** the relation between X and y is linear.
2. **Independence:** the errors (residuals) are independent (no autocorrelation).
3. **Homoscedasticity:** the residuals have constant variance.
4. **Normality:** the residuals are normally distributed.
5. **No multicollinearity:** the features are not strongly correlated with each other.
6. The errors have zero mean and there are no influential outliers.

B18 **Find the equation of the linear regression line using the data points (0,72), (5,66), (10,70), (15,64), (20,60).** Paper B, Section III **[10]**

n = 5. $\Sigma x = 50$, $\Sigma y = 332$, $\Sigma xy = 0 + 330 + 700 + 960 + 1200 = 3190$, $\Sigma x^2 = 0 + 25 + 100 + 225 + 400 = 750$.

- Slope $b = (n\Sigma xy - \Sigma x\Sigma y)/(n\Sigma x^2 - (\Sigma x)^2) = (5\times3190 - 50\times332)/(5\times750 - 2500) = (15950 - 16600)/1250 = \mathbf{-0.52}$
- Intercept $a = \bar{y} - b\cdot\bar{x} = 66.4 - (-0.52)(10) = \mathbf{71.6}$

Regression line: y = 71.6 − 0.52 x

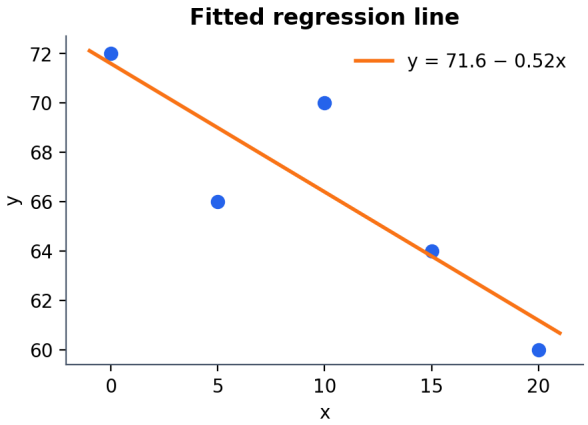

Figure: Data points and the fitted line

Unit 3, 4 and Other PYQ Topics

B19 **Explain the working of the Random Forest algorithm.** Paper B, Section II **[5]**

Random Forest = **many decision trees + bagging + random feature selection**.

1. Draw many **bootstrap samples** (random samples with replacement) from the training data.
2. Grow one tree per sample. At **each split** consider only a **random subset of features** (about √p), which makes the trees different from each other.
3. Grow trees fully (usually no pruning).
4. **Predict:** majority vote of all trees (classification) or the average (regression).

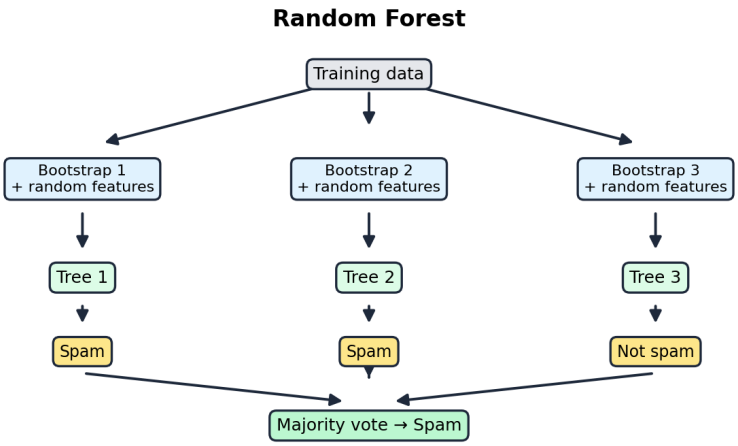

Figure: Random Forest

The data left out of each bootstrap sample gives a free **out-of-bag (OOB) error** estimate. **Pros:** less overfitting than one tree, handles large data, gives feature importance. **Cons:** less interpretable, slower than one tree.

B20 **Explain Ensemble Learning with an example.** Paper B, Section II **[5]**

Ensemble learning combines **several models (base learners)** to get a better and more stable prediction than any single model; different models make different errors, which cancel out. Types: **bagging** (parallel, reduces variance), **boosting** (sequential, reduces bias) and **stacking** (a meta-model combines the outputs). **Example:** three classifiers label an email spam, spam, not spam. Majority vote gives **spam**. Random Forest (bagging of trees) and AdaBoost are standard ensembles.

B21 **How many types of clustering are available in ML? Explain each type with examples.** Paper B, Section III **[10]**

Type	Idea	Example
Partitioning	Divide the data into k groups	k-Means (centroids), k-Medoids (real points as centres); customer segmentation
Hierarchical	Builds a tree (**dendrogram**); agglomerative (bottom-up merge) or divisive (top-down split)	Grouping species; **BIRCH** is a scalable hierarchical method for large data
Density-based	Clusters are dense regions; sparse points are noise; finds arbitrary shapes	DBSCAN; geographic hotspots
Model / distributi on-based	Each cluster is a probability distribution (soft clustering)	Gaussian Mixture Model with EM; speaker or image segmentation
Grid-based	Divides space into grid cells	STING

Also: **hard** clustering (each point in exactly one cluster) vs **soft** clustering (probabilities).

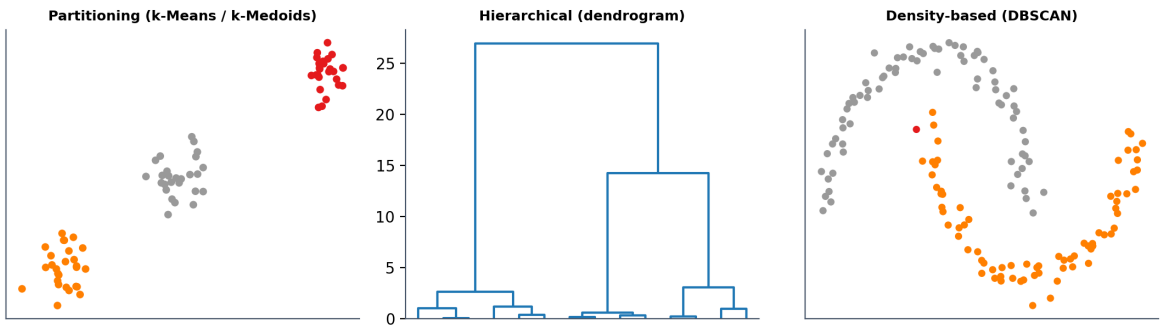

Figure: Partitioning, hierarchical and density-based clustering

B22 **Explain in detail: Geometric intuition of Adaptive boosting.** Paper B, Section III **[10]**

AdaBoost builds a strong classifier from many weak ones (e.g. **decision stumps**: a single straight-line split on one feature).

1. Start with **equal weights** on all points.
2. Train a stump (draw one line). Some points are misclassified.
3. Compute its error ε and its **vote weight α = ½ ln((1 − ε)/ε)**; a more accurate stump gets a bigger α.
4. **Increase the weights of misclassified points** (they are drawn bigger) and decrease the rest, then renormalise: $w_i \leftarrow w_i \cdot e^{\alpha}$ for mistakes, $w_i \cdot e^{-\alpha}$ for correct ones.
5. The next stump is forced to focus on the heavy, previously wrong points.
6. Repeat T times. **Final classifier = sign(Σ α_t $h_t(x)$)**.

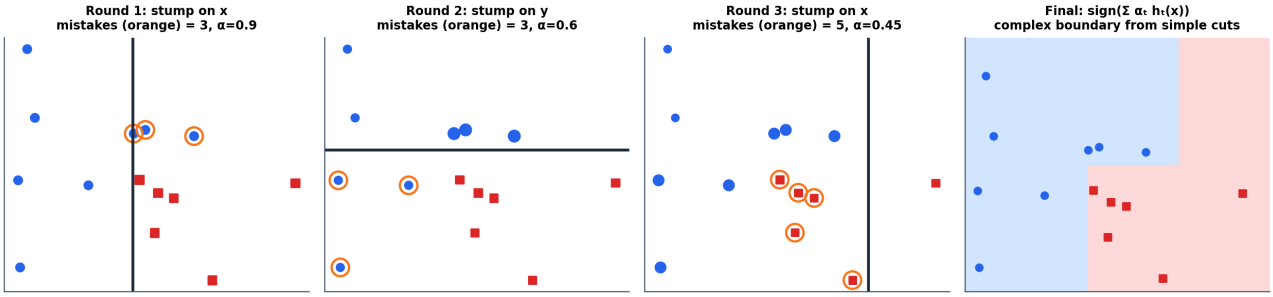

Figure (illustrative): each round re-weights the mistakes (orange rings); the final boundary combines the weighted stumps

Intuition: each stump is a straight cut of the plane. Adding the weighted votes of many cuts produces a **complex, staircase-shaped non-linear boundary** even though every piece is very simple. Hard points keep getting more attention round after round.

B23 **Examine the probability of a dangerous Fire when there is Smoke using naive Bayes. Dangerous fires are rare (1%), smoke is fairly common (10%), and 90% of dangerous fires make smoke.** Paper C, Section II **[5]**

Given P(Fire) = 0.01, P(Smoke) = 0.10, P(Smoke | Fire) = 0.9. By Bayes theorem:

```
P(Fire | Smoke) = P(Smoke | Fire) x P(Fire) / P(Smoke)
                = (0.9 x 0.01) / 0.10 = 0.009 / 0.10 = 0.09
```

Answer: 0.09 (9%). Even with smoke, a dangerous fire is only 9% likely, because fires are rare and smoke is common.

B24 **Explain Feed forward Neural Network in detail.** Paper C, Section II **[5]**

An FFNN is a neural network in which information flows in **one direction only**: input → hidden layer(s) → output, with **no loops**.

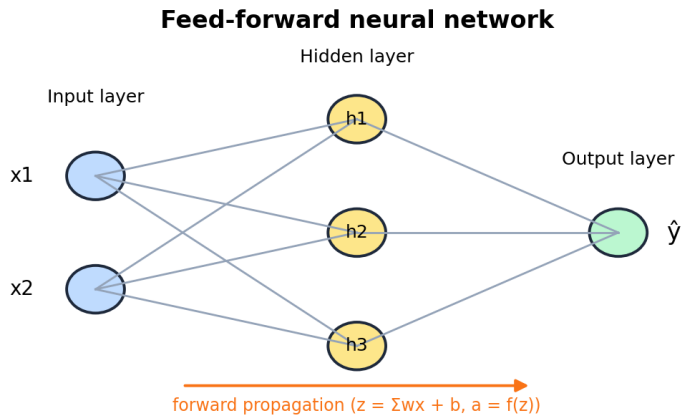

Figure: A feed-forward network (2 inputs, 3 hidden neurons, 1 output)

- **Layers:** input layer (one node per feature), one or more hidden layers, an output layer.

- **Neuron:** computes $z = \Sigma\, w_i x_i + b$, then applies an **activation function** $a = f(z)$ (sigmoid, tanh, ReLU; softmax at the output for classes).
- **Forward propagation:** inputs are passed layer by layer to produce the output $\hat{y}$.
- **Training:** a **loss function** (MSE or cross-entropy) compares $\hat{y}$ with the true y; **backpropagation** computes the gradient of the loss for every weight; **gradient descent** updates the weights; repeat for many epochs.
- **Uses:** image and digit recognition, spam detection, price prediction. **Limitations:** needs much data, hard to interpret, may overfit.

B25 **Apply the Candidate Elimination algorithm on the given dataset (Citations, Size, InLibrary, Price, Editions) and find the resultant hypothesis.** Paper C, Section III **[10]**

The target column is cut off in your photo. I used the standard version of this question ("will the book be bought?"): Example 1 = No; Examples 2, 3 and 4 = Yes. Please check this against your paper.

Start with $S_0 = \langle \varnothing, \varnothing, \varnothing, \varnothing, \varnothing \rangle$ and $G_0 = \langle ?, ?, ?, ?, ? \rangle$.

Step	Example	Class	S	G
1	Some, Small, No, Affordable, One	**No**	⟨∅, ∅, ∅, ∅, ∅⟩	{⟨Many,?,?,?,?⟩, ⟨?,Big,?,?,?⟩, ⟨?,Medium,?,?,?⟩, ⟨?,?,?,Expensive,?⟩, ⟨?,?,?,?,Few⟩, ⟨?,?,?,?,Many⟩}
2	Many, Big, No, Expensive, Many	**Yes**	⟨Many, Big, No, Expensive, Many⟩	remove ⟨?,Medium,..⟩ and ⟨?,?,?,?,Few⟩ (do not cover it)
3	Many, Medium, No, Expensive, Few	**Yes**	⟨Many, ?, No, Expensive, ?⟩	remove ⟨?,Big,..⟩ and ⟨?,?,?,?,Many⟩
4	Many, Small, No, Affordable, Many	**Yes**	⟨Many, ?, No, ?, ?⟩	remove ⟨?,?,?,Expensive,?⟩

Final: S = {⟨Many, ?, No, ?, ?⟩}, G = {⟨Many, ?, ?, ?, ?⟩}. The version space holds the hypotheses between them: **⟨Many, ?, No, ?, ?⟩** (most specific) and **⟨Many, ?, ?, ?, ?⟩** (most general). A book is bought when it has **many citations** (and is not in the library).

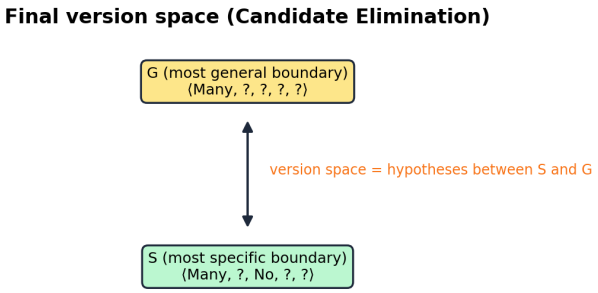

Figure: Final version space

B26 **Distinguish between Generative classifiers and Discriminative classifiers. Name some examples of each one.** Paper C, Section III **[10]**

	Generative	Discriminative
Learns	Joint distribution P(x, y), i.e. P(x|y) and P(y); classifies using Bayes rule	Directly P(y|x) or the decision boundary
Can generate new data?	Yes	No
Needs	More assumptions; works well with small data	Fewer assumptions; usually more accurate with large data
Examples	Naive Bayes, Hidden Markov Model, Gaussian Mixture Model, LDA, GANs	Logistic regression, SVM, decision trees, k-NN, neural networks

B27 Determine the significance of Dimensionality Reduction using the PCA algorithm. Paper C, Section III [10]

PCA converts correlated features into a smaller set of **uncorrelated principal components**, ordered by the variance they capture (PC1 captures the most; PC2 is orthogonal to PC1 with the next most, and so on).

Steps: (1) standardise the data; (2) compute the covariance matrix; (3) find its eigenvalues and eigenvectors; (4) sort by eigenvalue and keep the top k components (enough to explain about 90 to 95% of the variance); (5) project the data onto them.

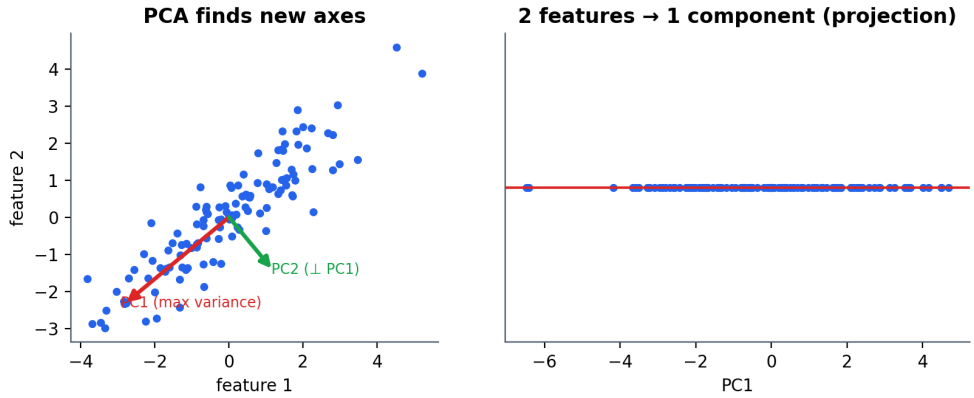

Figure: PCA finds the direction of maximum variance and projects onto it

- Fewer features, so **faster training** and less storage.
- **Removes redundancy and correlation** between features.
- **Reduces overfitting** and noise.
- Allows **visualising** high-dimensional data in 2D or 3D.
- **Example:** 10 features become 3 components preserving 90% of the information; or Maths, Physics and Chemistry marks combine into one “academic performance” component.
- **Limitations:** components are hard to interpret, only linear relations are captured, and it is sensitive to feature scale.

B28 Explain how the k-Medoids clustering algorithm works by giving an example. Paper C, Section III [10]

Like k-Means, but each cluster is represented by a **medoid**, an **actual data point** (the most central one), not a mean. This makes it **robust to outliers**. (The standard algorithm is PAM.)

1. Choose k points as initial medoids.
2. Assign every point to the nearest medoid.
3. For each medoid and each non-medoid point, try swapping them and recompute the total distance (cost).
4. Keep a swap only if the cost falls.
5. Stop when no swap improves the cost.

Example: data = 2, 3, 4, 10, 11, 12, 25 and k = 2 (distance = absolute difference). Initial medoids: 2 and 10.

Step	Medoids	Clusters	Total cost
Start	2, 10	{2,3,4}, {10,11,12,25}	(0+1+2) + (0+1+2+15) = **21**
Swap 2 → 3	3, 10	{2,3,4}, {10,11,12,25}	(1+0+1) + 18 = **20** (keep)
Swap 10 → 11	3, 11	{2,3,4}, {10,11,12,25}	2 + (1+0+1+14) = **18** (keep)
Further swaps			no swap lowers the cost, so stop

Result: clusters {2, 3, 4} with medoid **3** and {10, 11, 12, 25} with medoid **11**; total cost 18. The outlier 25 does not drag the medoid away; in k-Means the mean would be pulled to 14.5.