Comprehensive Machine Learning Practice Set

Based on Mid-Semester Patterns & Syllabus Topics

42 Questions (7 Per Topic) with Detailed Step-by-Step Solutions

Contents

1 Topic 1: Introduction to Machine Learning

Sub-topics: Definitions (E-T-P), Types of Learning, Design a Learning System.

Q1. (Definition of Learning) A computer program is said to learn from experience E with respect to task T and performance measure P. Consider a **Self-Driving Car**.

1. Define T, P, and E.
2. Why is this considered a machine learning problem rather than a traditional programming problem?

Detailed Solution:

1. **Parameters:**
   - **Task (T):** The specific operation to be performed, e.g., driving autonomously on public roads (steering, braking, accelerating, changing lanes).
   - **Performance Measure (P):** The metric used to evaluate the system, e.g., average distance traveled before a human safety driver must intervene, or the rate of accidents per million miles.
   - **Experience (E):** The data the system learns from, e.g., a database of millions of miles of recorded video feeds, Lidar sensor data, and human driver reactions in specific situations.
2. **Reasoning:** Traditional programming requires explicitly defining rules (e.g., `if raining then speed = 40`). Driving involves infinite edge cases (glare, unpredictable pedestrians, faded lane markings) that cannot be hard-coded. ML allows the system to *infer* these rules from data (E) to optimize performance (P) on the task (T).

Q2. (Types of Learning) Classify the following scenarios into Supervised, Unsupervised, or Reinforcement Learning:

1. A system analyzes network traffic to find unusual spikes (anomalies) without knowing what "attacks" look like beforehand.
2. A chess engine learns to play by playing against itself and receiving a +1 for a win and -1 for a loss.
3. A hospital predicts if a patient has diabetes based on their age, weight, and blood pressure using historical records where the diagnosis is known.

Detailed Solution:

1. **Unsupervised Learning (Anomaly Detection):** The data is unlabeled. The system is looking for structural deviations from the norm rather than mapping inputs to known outputs.
2. **Reinforcement Learning:** There is no dataset of "correct moves." The agent takes actions (moves) and receives delayed feedback (rewards/penalties) to update its policy.
3. **Supervised Learning (Classification):** The training data includes the "ground truth" labels (Diabetes: Yes/No), and the goal is to learn the mapping $f(x) \rightarrow y$.

Q3. (Design a Learning System) You are designing a system to play Checkers.

1. What is the **Target Function** $V(b)$?
2. Since we cannot store $V(b)$ for every board state (state space is too large), how do we represent it?

Detailed Solution:

1. **Target Function:** V : Board $\to \mathbb{R}$. The ideal function $V(b)$ maps any legal board state b to a real number representing the probability that the current player will win starting from that state (e.g., 100 for a definite win, -100 for a definite loss).
2. **Function Approximation:** Since there are approximately 10^{20} states, we cannot store a lookup table. We approximate $V(b)$ as $\hat{V}(b)$ using a linear combination of features:

$$\hat{V}(b) = w_0 + w_1 x_1(b) + w_2 x_2(b) + \cdots + w_n x_n(b)$$

where w_i are weights to be learned, and x_i are features (e.g., x_1 = number of black pieces, x_2 = number of kings).

Q4. (Inductive Bias) What is the "Inductive Bias" of a Decision Tree? How does it differ from the bias of Linear Regression?

Detailed Solution: Inductive Bias refers to the set of assumptions a learner uses to predict outputs for unseen inputs.

- **Decision Tree Bias:** Assumes the decision boundary consists of axis-parallel hyper-rectangles. It also prefers smaller (shorter) trees over complex ones (Occam's Razor).
- **Linear Regression Bias:** Assumes the relationship between input variables and the output is smooth and linear (can be represented by a straight line or hyperplane).

Q5. (Overfitting) A student claims: "My model has 100% Training Accuracy, so it is perfect." Explain why this is likely incorrect using the concepts of **Signal** and **Noise**.

Detailed Solution: Data consists of two components: **Signal** (the true underlying pattern) and **Noise** (random errors or fluctuations).

$$\text{Data} = \text{Signal} + \text{Noise}$$

A model with 100% training accuracy has likely memorized the specific noise in the training set along with the signal. This is called **Overfitting**. When presented with new data (which has different noise), the model will fail because it is relying on random quirks of the training data rather than the general rule.

Q6. (Semi-Supervised) Describe a scenario where Semi-Supervised Learning is useful.

Detailed Solution: Scenario: Medical Imaging Analysis.

- **Problem:** We have 10,000 X-ray scans.
- **Cost:** Getting a doctor to label (diagnose) a scan costs \$50/scan. We can only afford to label 100 scans.
- **Solution:** We use the 100 labeled scans (Supervised) combined with the structure inherent in the 9,900 unlabeled scans (Unsupervised) to build a classifier that performs better than one trained on just the 100 labeled images. This is Semi-Supervised Learning.

Q7. (Classification vs Regression) Is predicting the *price* of a house Classification or Regression? Is predicting if the house will sell *within a month* (Yes/No) Classification or Regression?

Detailed Solution:

1. **Price Prediction → Regression:** The output is a continuous numerical value (e.g., $500,000.50).
2. **Time-frame Prediction → Classification:** The output is a discrete category/class label (Class 1: "Yes", Class 0: "No").

2 Topic 2: Mathematical Preliminaries & ML Workflow

Sub-topics: Metrics (Precision, Recall, Specificity), Scaling, Sampling.

Q1. (Performance Metrics) A medical test for a disease is given to 1000 people.

- Actual Positive (Sick): 100
- Actual Negative (Healthy): 900
- Predicted Positive: 80 (of which 60 are actually Sick).

Calculate **Precision**, **Recall**, and **Specificity**.

Detailed Solution: Step 1: Construct the Confusion Matrix

	Predicted Pos	**Predicted Neg**	**Total**
Actual Pos (Sick)	TP = 60	FN = 40	100
Actual Neg (Healthy)	FP = 20	TN = 880	900
Total	80	920	1000

Calculations:

- $TP = 60$ (Given).
- $FP =$ (Total Pred Pos) $- TP = 80 - 60 = 20$.
- $FN =$ (Total Actual Pos) $- TP = 100 - 60 = 40$.
- $TN =$ (Total Actual Neg) $- FP = 900 - 20 = 880$.

Step 2: Compute Metrics

- **Precision** (Accuracy of positive predictions):

$$\frac{TP}{TP+FP} = \frac{60}{60+20} = \frac{60}{80} = \mathbf{0.75}$$

- **Recall** (Ability to find all positive cases):

$$\frac{TP}{TP+FN} = \frac{60}{60+40} = \frac{60}{100} = \mathbf{0.60}$$

- **Specificity** (Ability to identify healthy cases correctly):

$$\frac{TN}{TN+FP} = \frac{880}{880+20} = \frac{880}{900} \approx \mathbf{0.98}$$

Q2. (Feature Scaling) You have a dataset with ”Age” (0-100) and ”Salary” (20,000-200,000). You are using k-NN.

1. Why is scaling necessary here?
2. Calculate the Min-Max normalized value for a Salary of 110,000.

Detailed Solution:

1. **Reason:** k-NN relies on Euclidean distance: $d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$.
   - Age range is 100; Salary range is 180,000.

  - A small % change in salary (e.g., 1000 units) will mathematically dwarf a huge % change in Age (e.g., 50 units). The algorithm will bias entirely towards Salary, effectively ignoring Age.

2. **Calculation:** Formula: $x_{norm} = \frac{x - x_{min}}{x_{max} - x_{min}}$

$$x_{norm} = \frac{110,000 - 20,000}{200,000 - 20,000} = \frac{90,000}{180,000} = \mathbf{0.5}$$

Q3. (Validation Set) What is the specific purpose of a Validation Set? How does it differ from a Test Set?

Detailed Solution:

- **Training Set:** Used to learn parameters (weights/biases).
- **Validation Set:** Used **during development** to tune Hyperparameters (e.g., Learning Rate α, k in k-NN, Depth of Tree). It provides feedback to adjust the model architecture.
- **Test Set:** Used **only once** at the very end. It acts as an unbiased evaluation of the final model. If you use the Test Set to tune parameters, you commit ”Data Leakage,” and your reported accuracy will be overly optimistic.

Q4. (Sampling) You have a dataset with 990 healthy patients and 10 sick patients. You want to split it into Train (80%) and Test (20%). Why must you use **Stratified Sampling**?

Detailed Solution: The dataset is highly imbalanced (1% sick).

- **Random Sampling:** In a 20% split (200 samples), there is a statistical probability that **zero** sick patients end up in the Test set (or all of them do).
- **Stratified Sampling:** This technique forces the split to respect the original class distribution. It ensures exactly 2 sick people (1% of 200) go to Test and 8 go to Train, ensuring the model is evaluated on its ability to detect the minority class.

Q5. (Data Leakage) Explain why filling missing values with the **mean of the entire dataset** before splitting into Train/Test is considered Data Leakage.

Detailed Solution:

- If you calculate the mean of the *entire* column, that mean includes values from rows that will eventually belong to the **Test set**.
- When you fill the Training set with this global mean, the Training set essentially ”peeks” at the Test data.
- **Correct Approach:** Split data first. Calculate the mean of the **Training set only**. Fill missing values in Training set with Train-Mean. Fill missing values in Test set with Train-Mean.

Q6. (Outliers) Which metric is more robust to outliers: Mean or Median? Explain with the set $\{1, 2, 3, 100\}$.

Detailed Solution: The Median is robust.

- **Mean:** $\frac{1+2+3+100}{4} = \frac{106}{4} = 26.5$. The value 100 drags the mean far away from the ”typical” data (1, 2, 3).

- **Median:** The middle value of $\{1, 2, 3, 100\}$ is the average of 2 and 3, which is 2.5. This accurately represents the majority of the data, ignoring the extreme value.

Q7. (F1 Score) Why is F1 Score preferred over Accuracy for imbalanced datasets?
Detailed Solution: Consider a dataset with 99 healthy people and 1 sick person.

- A model that predicts "Everyone is Healthy" has:
  - **Accuracy:** 99% (Misleadingly high).
  - **Recall:** 0% (It found 0 sick people).
- **F1 Score** is the Harmonic Mean of Precision and Recall:

  $$F1 = 2 \times \frac{P \times R}{P + R}$$

  If Recall is 0, the F1 score becomes 0. It penalizes the model heavily for ignoring the minority class, whereas Accuracy rewards the model for simply guessing the majority class.

3 Topic 3: Linear Models for Regression

Sub-topics: Gradient Descent, Bias-Variance, Regularization (Lasso/Ridge).

Q1. (Bias-Variance Diagnostics)

- Model A: Train Error = 20%, Test Error = 22%. (Ideal Error $\approx$ 5%)
- Model B: Train Error = 5%, Test Error = 30%.

Identify which model suffers from High Bias and which from High Variance.

Detailed Solution:

- **Model A (High Bias / Underfitting):** The Training Error is high (20% vs 5
- **Model B (High Variance / Overfitting):** The Training Error is low (matches ideal), but the Test Error is high (huge gap). The model has captured the training data (including noise) perfectly but fails to generalize to new data.

Q2. (Gradient Descent) Given Cost $J(\theta) = (\theta - 2)^2$. Initial $\theta = 4$. Learning rate $\alpha = 0.1$. Perform one update step.

Detailed Solution: Step 1: Calculate the Gradient (Derivative)

$$\frac{\partial J(\theta)}{\partial \theta} = \frac{d}{d\theta}(\theta - 2)^2 = 2(\theta - 2)$$

Step 2: Plug in current $\theta = 4$

$$\text{Gradient} = 2(4 - 2) = 2(2) = 4$$

Step 3: Apply Update Rule

$$\theta_{new} = \theta_{old} - \alpha \times \text{Gradient}$$

$$\theta_{new} = 4 - 0.1(4)$$

$$\theta_{new} = 4 - 0.4 = \mathbf{3.6}$$

(Check: The minimum is at 2. The value moved from 4 to 3.6, closer to the minimum.)

Q3. (Regularization) Which regularization technique (Lasso or Ridge) sets coefficients exactly to zero? Why is this useful?

Detailed Solution:

- **Lasso (L1 Regularization):** Adds a penalty term $\lambda \sum |\theta_j|$. The geometry of this penalty (diamond shape) encourages solutions where coefficients hit exactly zero.
- **Utility:** This acts as automatic **Feature Selection**. In a dataset with 1000 features where only 10 are relevant, Lasso can reduce the model to those 10 features, removing noise and improving interpretability. Ridge (L2) only shrinks coefficients to be *small*, but rarely zero.

Q4. (Normal Equation) Why might we use Gradient Descent instead of the Normal Equation $\theta = (X^T X)^{-1} X^T y$ for a dataset with 50,000 features?

Detailed Solution: The Normal Equation involves computing the inverse of the matrix $(X^T X)$.

- If features $n = 50,000$, the matrix is $50,000 \times 50,000$.
- Matrix inversion has a computational complexity of approximately $O(n^3)$.
- $50,000^3$ is an astronomically large number of operations, making it computationally infeasible.
- **Gradient Descent** scales with $O(kn^2)$ or $O(kn)$, making it much faster for large feature sets.

Q5. (R-Squared) Can R^2 be negative? What does it imply?
Detailed Solution: Yes, R^2 can be negative. The formula is: $R^2 = 1 - \frac{SS_{res}}{SS_{tot}}$.

- SS_{tot} is the error of a baseline model that just predicts the **Mean** ($y = \bar{y}$) for everyone.
- If your model has an error (SS_{res}) **larger** than the error of simply guessing the mean, the fraction $\frac{SS_{res}}{SS_{tot}} > 1$.
- Result: $1 - (> 1) =$ Negative.
- **Implication:** The model fits the data **worse** than a horizontal line. This usually happens if you forget to include an intercept term or the data is non-linear and you fit a straight line.

Q6. (Polynomial Regression) How do we model non-linear data (like a curve) using a linear regression algorithm?

Detailed Solution: We use **Basis Expansion**. We do not change the algorithm; we change the **features**.

- If input is x, and the data looks quadratic, we create a new feature $x_2 = x^2$.
- The model becomes: $h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2$.
- Even though the function is quadratic with respect to x, it is still **linear with respect to the parameters θ**. Therefore, we can still use Standard Linear Regression techniques to solve it.

Q7. (Collinearity) What happens to Linear Regression if two features are perfectly correlated ($x_1 = 2x_2$)?
Detailed Solution:

- **Mathematical Issue:** The matrix $X^T X$ becomes **Singular** (determinant is 0). This means it is non-invertible, and the Normal Equation fails.
- **Interpretability Issue:** There are infinite combinations of weights θ_1 and θ_2 that yield the same prediction (e.g., $2w_1 + w_2 =$ const). The algorithm cannot decide which feature is "important," leading to highly unstable weights (large positive and large negative values canceling out).

4 Topic 4: Linear Models for Classification

Sub-topics: Decision Boundaries, LDA, One-vs-Rest.

Q1. (Decision Boundary) A classifier has weights $w = [1, -1]$ and bias $b = 0$.

1. Write the equation of the boundary.
2. Classify the point $(5, 2)$.

Detailed Solution:

1. **Equation:** The boundary is where $w^T x + b = 0$.

$$1(x_1) + (-1)(x_2) + 0 = 0 \implies x_1 - x_2 = 0 \implies \mathbf{x_1 = x_2}$$

   This is a 45-degree diagonal line passing through the origin.

2. **Classification:** We calculate the score z.

$$z = 1(5) - 1(2) = 3$$

   Since $z = 3 > 0$, the point lies on the positive side of the boundary. It is classified as **Class 1 (Positive)**.

Q2. (Fisher's LDA) What is the objective of Fisher's Linear Discriminant Analysis (LDA) regarding "Between-class" and "Within-class" variance?

Detailed Solution: LDA aims to reduce dimensions while preserving class separability. It seeks a projection vector w that maximizes the Fisher Criterion:

$$J(w) = \frac{\text{Between-Class Variance}}{\text{Within-Class Variance}}$$

- **Maximize Between-Class Variance (S_B):** We want the means of the two classes to be as far apart as possible.
- **Minimize Within-Class Variance (S_W):** We want the data points of each class to be tightly clustered around their mean (low spread).

Q3. (One-vs-Rest) You have 5 classes. How many binary classifiers do you train in a One-vs-Rest strategy? How do you make the final prediction?

Detailed Solution:

- **Training:** You train **5** separate binary classifiers.
  1. Class 1 vs (2,3,4,5)
  2. Class 2 vs (1,3,4,5) ... and so on.
- **Prediction:** For a new input x, you run it through all 5 classifiers. Each classifier outputs a probability score. You select the class corresponding to the classifier with the **highest confidence score**.

Q4. (Least Squares Failure) Why is minimizing Squared Error (Linear Regression) bad for classification, even for binary tasks?

Detailed Solution:

- **Problem 1 (Outliers):** If you have a data point that is "very correct" (e.g., a positive case far away from the boundary), Linear Regression sees this as a large error because it tries to fit a line directly through the points (predicting a value like 5.0 instead of 1.0).
- **Result:** To reduce this "error," the regression line shifts, which often moves the decision boundary ($y = 0.5$) incorrectly, causing misclassification of points near the center.

Q5. (XOR Problem) Why can't a single Perceptron solve the XOR problem?
Detailed Solution:

- The XOR dataset is: $(0, 0) \to 0, (1, 1) \to 0, (0, 1) \to 1, (1, 0) \to 1$.
- If you plot these on a 2D graph, the Class 1 points (0,1 and 1,0) are diagonally opposite.
- A single Perceptron (or Linear Classifier) can only draw **one straight line**.
- It is geometrically impossible to draw a single straight line that separates the (0,1) and (1,0) points from the (0,0) and (1,1) points. The problem is **not Linearly Separable**.

Q6. (Generative vs Discriminative) Is Naive Bayes Generative or Discriminative? What about Logistic Regression?
Detailed Solution:

- **Naive Bayes (Generative):** It models the underlying probability distribution of each class $P(x|y)$ and the prior $P(y)$. It learns "how the data is generated."
- **Logistic Regression (Discriminative):** It models the posterior probability $P(y|x)$ directly. It focuses solely on finding the boundary that separates the classes, without caring how the data was generated.

Q7. (Encoding) Why must we use One-Hot Encoding for categorical targets (Red, Green, Blue) instead of labels 1, 2, 3 in regression?
Detailed Solution: If we assign Red=1, Green=2, Blue=3:

- The model assumes a mathematical order: Blue > Green > Red.
- The model assumes distance: The difference between Blue and Red (3-1=2) is twice the difference between Green and Red (2-1=1).
- Since colors are nominal (no order), this confuses the model. **One-Hot Encoding** creates binary vectors $[1, 0, 0], [0, 1, 0], [0, 0, 1]$, treating all categories as equidistant and independent.

5 Topic 5: Logistic Regression

Sub-topics: Sigmoid, Log-Loss, Softmax.

Q1. (Sigmoid) Calculate the output of the sigmoid function $\sigma(z)$ when $z = 0$. What does this signify?

Detailed Solution: Formula: $\sigma(z) = \frac{1}{1+e^{-z}}$

$$\sigma(0) = \frac{1}{1 + e^{-0}} = \frac{1}{1+1} = \mathbf{0.5}$$

Significance: In Logistic Regression, the output represents probability. 0.5 implies a 50/50 chance. Geometrically, any point where $z = 0$ lies exactly on the **Decision Boundary**.

Q2. (Log-Loss) The model predicts $\hat{y} = 0.9$. The actual label is $y = 0$. Calculate the Log-Loss.

Detailed Solution: Formula: $J = -[y \ln(\hat{y}) + (1 - y) \ln(1 - \hat{y})]$

1. Substitute $y = 0$ and $\hat{y} = 0.9$.
2. The first term $y \ln(\hat{y})$ becomes $0 \times \ln(0.9) = 0$.
3. The second term is $(1 - 0) \ln(1 - 0.9) = 1 \times \ln(0.1)$.
4. $\ln(0.1) \approx -2.302$.
5. $J = -[0 + (-2.302)] = \mathbf{2.302}$.

Interpretation: The model was 90% confident it was Class 1, but it was Class 0. The loss is high to penalize this confident error.

Q3. (Overfitting Signs) You observe that the weights in your Logistic Regression model are extremely large (e.g., $\theta_1 = 5000$). What does this indicate?

Detailed Solution: This indicates **Overfitting**.

- The Sigmoid function $\frac{1}{1+e^{-\theta^T x}}$ becomes sharper as θ increases.
- With $\theta = 5000$, the sigmoid resembles a **Step Function**. It jumps from 0 to 1 instantly.
- This means the model is making extremely "hard" decisions based on microscopic differences in input, likely fitting noise in the training data to separate classes perfectly.

Q4. (Softmax) In a 3-class problem, logits are $z = [2.0, 1.0, 0.1]$. Calculate probability of Class 1.

Detailed Solution: Formula: $P(y = k) = \frac{e^{z_k}}{\sum e^{z_j}}$

1. Calculate Exponentials:
   - $e^{2.0} \approx 7.389$
   - $e^{1.0} \approx 2.718$
   - $e^{0.1} \approx 1.105$
2. Calculate Sum: $7.389 + 2.718 + 1.105 = 11.212$.
3. Calculate Probability for Class 1 ($z = 2.0$):

$$P(1) = \frac{7.389}{11.212} \approx \mathbf{0.659} \text{ (approx 66\%)}$$

Q5. (Log-Odds) If the Log-Odds ($\theta^T x$) is 0, what is the probability $P(y = 1)$?
Detailed Solution: The "Log-Odds" is defined as $\ln\left(\frac{P}{1-P}\right) = z$.

1. Set $z = 0$: $\ln\left(\frac{P}{1-P}\right) = 0$.
2. Exponentiate both sides: $\frac{P}{1-P} = e^0 = 1$.
3. Solve for P:

$$P = 1(1 - P) \implies P = 1 - P \implies 2P = 1 \implies \mathbf{P = 0.5}$$

Q6. (Convexity) Why do we use Log-Loss instead of MSE for Logistic Regression?
Detailed Solution:

- If we substitute the Sigmoid function into the Mean Squared Error (MSE) formula, the resulting Cost Function $J(\theta)$ is **Non-Convex**.
- A non-convex function has many "wavy" hills and valleys (local minima). Gradient Descent is likely to get stuck in a local minimum and fail to find the best weights.
- Log-Loss is **Convex** (bowl-shaped), guaranteeing that Gradient Descent will find the Global Minimum.

Q7. (Regularization) How does increasing the regularization parameter λ affect the Variance of the model?
Detailed Solution:

- Increasing λ increases the penalty on the magnitude of the weights.
- This forces the model to learn smaller, simpler weights, effectively reducing the model's complexity.
- A simpler model is less sensitive to small fluctuations in the training data.
- Therefore, increasing λ **Decreases Variance** (reduces overfitting), though it may risk increasing Bias (underfitting).

6 Topic 6: Decision Trees

Sub-topics: Entropy, Information Gain, Gain Ratio, Pruning.

Q1. (Entropy) A bag has 6 Red balls and 0 Blue balls. Calculate Entropy.

Detailed Solution: Formula: $H(S) = -p_+ \log_2(p_+) - p_- \log_2(p_-)$.

- $p_{red} = \frac{6}{6} = 1$.
- $p_{blue} = \frac{0}{6} = 0$.
- Note: By definition, $0 \log_2 0 = 0$.
- Calculation:
$$H = -(1 \log_2 1 + 0 \log_2 0) = -(1 \times 0 + 0) = \mathbf{0}$$

Interpretation: The set is perfectly pure. There is no uncertainty (Entropy = 0).

Q2. (Information Gain) Parent Entropy = 1.0. Split A creates two children with Entropy 0.5 each. Split is 50/50. Calculate Gain.

Detailed Solution: Formula: $Gain = H(Parent) - \sum \frac{|S_v|}{|S|} H(S_v)$

1. **Weighted Average Entropy of Children:**
$$(0.5 \times 0.5) + (0.5 \times 0.5) = 0.25 + 0.25 = 0.5$$
2. **Gain:**
$$1.0 - 0.5 = \mathbf{0.5}$$

Q3. (Gain Ratio) Why is **Gain Ratio** preferred over Information Gain? (Think of the "Customer ID" problem).

Detailed Solution:

- **Flaw of Information Gain:** It favors attributes with many unique values. If you split on "Customer ID," every leaf has 1 sample (Pure). Entropy is 0, so Information Gain is maximized. However, this model is useless (overfitting).
- **Gain Ratio Solution:** It introduces a denominator called "Split Information" (Entropy of the attribute itself).
$$\text{GainRatio} = \frac{\text{InfoGain}}{\text{SplitInfo}}$$
Since "Customer ID" has very high entropy (very chaotic), the denominator is large, reducing the Gain Ratio score and preventing the tree from picking that attribute.

Q4. (Continuous Attributes) How does a decision tree handle "Temperature" (Continuous)?

Detailed Solution: The algorithm converts the continuous variable into a binary split (Temperature $< X$).

1. Sort the data by Temperature.
2. Identify all possible split points (usually midpoints between adjacent distinct values).
3. For every split point, calculate the Information Gain.

4. Select the split point that yields the **Maximum Information Gain**.

Q5. (Overfitting) Why do deep trees overfit?
Detailed Solution:

- As a tree gets deeper, the number of samples in the leaf nodes decreases.
- Eventually, a leaf might contain only 1 or 2 samples.
- At this depth, the tree is learning rules specific to the **noise** or idiosyncrasies of those specific samples, rather than general population patterns.
- This results in poor generalization to new data (High Variance).

Q6. (Gini Impurity) Calculate Gini for a node with $p_+ = 0.5, p_- = 0.5$.
Detailed Solution: Formula: $Gini = 1 - \sum(p_i)^2$

1. Sum of squares:
$$(0.5)^2 + (0.5)^2 = 0.25 + 0.25 = 0.5$$
2. Subtract from 1:
$$1 - 0.5 = \mathbf{0.5}$$

Note: 0.5 is the maximum possible impurity for a binary classification using Gini Index (representing a 50/50 random mix).

Q7. (Pruning) Explain **Post-Pruning** (Reduced Error Pruning).
Detailed Solution:

1. **Train Full Tree:** Allow the tree to grow until it overfits (leaves are pure).
2. **Evaluate Bottom-Up:** Start at the leaf nodes. Consider replacing a subtree (a decision node and its children) with a single leaf node carrying the majority label.
3. **Validation Check:** Test the accuracy of the pruned tree on a separate **Validation Set**.
4. **Decision:** If removing the branch improves (or does not reduce) validation accuracy, prune it permanently. This removes noise-specific branches.