Decision Trees and Random Forest - Comprehensive Question Bank

QUESTION 1: Decision Tree Fundamentals and Manual Calculation (12 marks)

Context: A bank wants to predict loan approval based on customer data. Consider the following training dataset:

ID	Income	Credit_Score	Employment	Loan_Approved
1	High	Good	Yes	Yes
2	Low	Poor	No	No
3	Medium	Good	Yes	Yes
4	High	Poor	Yes	No
5	Low	Good	No	No
6	Medium	Poor	No	No
7	High	Good	No	Yes
8	Low	Poor	Yes	No

Part A (4 marks): Calculate the Shannon entropy for the root node. Show all steps in your calculation.

Part B (4 marks): Calculate the information gain for splitting on the "Income" attribute. Show detailed calculations for conditional entropy.

Part C (2 marks): Explain why decision trees use greedy strategies and what this means for the optimality of the final tree.

Part D (2 marks): Compare the advantages and disadvantages of using Information Gain versus Gini Index as splitting criteria.

QUESTION 2: Python Implementation - Decision Tree from Scratch (10 marks)

Part A (6 marks): Implement the following Python functions for a decision tree classifier:

```
python
```

```python
def calcShannonEnt(dataSet):
    """
    Calculate Shannon Entropy of a dataset
    Args: dataSet - list of lists, where each inner list represents a sample
    Returns: Shannon entropy value
    """
    # Your implementation here
    pass

def chooseBestMultiSplit(dataSet):
    """
    Choose the best feature for splitting based on information gain
    Args: dataSet - list of lists with features and class labels
    Returns: index of best feature to split on
    """
    # Your implementation here
    pass
```

Part B (2 marks): Write a function to split the dataset based on a feature value:

```python
def splitDataSet(dataSet, axis, value):
    """
    Split dataset based on feature value
    Args:
        dataSet - input dataset
        axis - feature index to split on
        value - feature value to match
    Returns: subset of data matching the condition
    """
    # Your implementation here
    pass
```

Part C (2 marks): Explain how you would modify your implementation to handle continuous attributes. Provide pseudocode for the modification.

QUESTION 3: Tree Pruning and Overfitting Analysis (8 marks)

Context: You've built a decision tree for predicting student academic performance with 95% accuracy on training data but only 70% on test data.

Part A (3 marks):

1. Identify the problem described above

2. Explain why this occurs in decision trees
3. List three specific causes that lead to this issue

Part B (3 marks): Compare pre-pruning and post-pruning approaches:

- Explain the methodology of each approach
- Discuss the advantages and disadvantages of each
- When would you choose one over the other?

Part C (2 marks): Implement a pre-pruning modification to the `chooseBestMultiSplit` function that stops splitting when:

- Information gain is below 0.1
- Node contains fewer than 5 samples

QUESTION 4: Gini Index Manual Calculation (8 marks)

Context: Consider a marketing dataset for predicting customer purchase behavior:

Customer_Type	Age_Group	Purchase
Premium	Young	Yes
Regular	Middle	No
Premium	Old	Yes
Regular	Young	No
Premium	Middle	Yes
Regular	Old	No
Premium	Young	Yes
Regular	Middle	Yes

Part A (3 marks): Calculate the Gini index for the root node. Show all calculation steps.

Part B (4 marks): Calculate the Gini index after splitting on "Customer_Type". Show:

- Gini index for each child node
- Weighted average Gini index
- Gini gain (reduction in impurity)

Part C (1 mark): Compare this Gini gain with what you would expect from Information Gain and explain which splitting criterion you would prefer for this dataset.

QUESTION 5: Random Forest Implementation and Theory (7 marks)

Part A (3 marks): Explain the three randomization techniques used in Random Forest:

1. Bagging
2. Forest-RI (Random Input Selection)
3. Forest-RC (Random Linear Combinations)

Provide specific examples of how each technique introduces randomness.

Part B (2 marks): Implement a simple bagging function:

```python
def createBaggedDataset(dataSet, numSamples):
    """
    Create a bagged dataset using sampling with replacement
    Args:
        dataSet - original training dataset
        numSamples - number of samples to draw
    Returns: bagged dataset
    """
    # Your implementation here
    pass
```

Part C (2 marks): A Random Forest with 100 trees achieves 85% accuracy, while individual trees average 78% accuracy. Explain:

1. Why the ensemble performs better than individual trees
2. What factors determine the optimal number of trees in the forest

---

QUESTION 6: Real-world Application - Flight Delay Prediction (5 marks)

Context: An airline wants to predict flight delays using the following features:

- Weather conditions (Clear, Cloudy, Rainy, Stormy)
- Time of day (Morning, Afternoon, Evening, Night)
- Aircraft age (New, Medium, Old)
- Route popularity (High, Medium, Low)

Part A (2 marks): Design the decision tree structure:

- Suggest the order in which you would consider splitting on these attributes
- Justify your reasoning based on expected information gain

Part B (2 marks): The airline has 50,000 flight records. Explain:

1. How Random Forest would handle this large dataset better than a single decision tree
2. What preprocessing steps you would recommend for this real-world dataset

Part C (1 mark): If the model shows high variance (different results on different data samples), suggest two specific techniques to reduce variance and improve model stability.

ADDITIONAL PRACTICE QUESTIONS

Short Answer Questions (3-5 marks each)

Q7: Explain why decision trees are considered "white box" models compared to neural networks. Provide three specific advantages this offers in business applications.

Q8: A decision tree for medical diagnosis has 1000 leaf nodes. Discuss whether this indicates overfitting and suggest three strategies to optimize the model.

Q9: Compare the computational complexity of training a single decision tree versus a Random Forest with 100 trees. Consider both time and space complexity.

Q10: In a Random Forest for image classification, explain how you would handle:

1. Continuous pixel values
2. Missing pixel data
3. Feature selection from thousands of pixel features

Coding Challenges (6-8 marks each)

Q11: Implement a complete decision tree classifier class with methods for:

- Training (`fit`)
- Prediction (`predict`)
- Tree visualization (`print_tree`)

Q12: Create a function that converts a trained decision tree into a set of if-else rules in natural language for business users.

Q13: Implement cross-validation for Random Forest hyperparameter tuning (number of trees, max depth, min samples per leaf).

Case Study Questions (10-12 marks each)

Q14: **E-commerce Recommendation System** An online retailer wants to predict customer purchase probability using decision trees. Given features like browsing history, demographics, and seasonal patterns, design a complete solution including data preprocessing, model selection, and evaluation metrics.

Q15: **Healthcare Diagnosis Support** Design a Random Forest system for early disease detection using patient symptoms and test results. Address ethical considerations, interpretability requirements, and handling of imbalanced medical data.

Q16: **Financial Fraud Detection** Implement a real-time fraud detection system using ensemble methods. Consider the challenges of concept drift, false positive costs, and the need for model updates with streaming transaction data.

EXAM SIMULATION QUESTIONS

3-Hour Practice Exam (50 marks total)

Question 1 (12 marks): Complete decision tree implementation with entropy calculations **Question 2** (10 marks): Random Forest ensemble with bagging implementation
Question 3 (8 marks): Manual Gini index calculations and pruning analysis **Question 4** (8 marks): Real-world application with data preprocessing **Question 5** (7 marks): Comparative analysis of splitting criteria **Question 6** (5 marks): Algorithm optimization and complexity analysis

---

MARKING RUBRICS

Theoretical Questions:

- **Excellent (90-100%)**: Complete understanding with clear explanations and examples
- **Good (75-89%)**: Solid understanding with minor gaps in explanation
- **Satisfactory (60-74%)**: Basic understanding with some conceptual errors
- **Needs Improvement (<60%)**: Significant gaps in understanding

Practical Implementation:

- **Excellent (90-100%)**: Correct, efficient code with proper error handling
- **Good (75-89%)**: Mostly correct code with minor syntax/logic errors
- **Satisfactory (60-74%)**: Basic functionality with some implementation issues
- **Needs Improvement (<60%)**: Non-functional or significantly flawed code

Manual Calculations:

- **Excellent (90-100%)**: All steps shown correctly with accurate final answer
- **Good (75-89%)**: Correct methodology with minor calculation errors
- **Satisfactory (60-74%)**: Correct approach but some steps missing or incorrect
- **Needs Improvement (<60%)**: Incorrect methodology or major calculation errors