Post-Class Assignments: Decision Trees and Random Forests

Assignment 1: Basic - Client Problem Classification with Decision Trees

Context: As a new AI engineer for the One Hour AI Solution platform, you need to develop a model that helps classify client problems to route them to the appropriate AI engineer with matching expertise.

Task: Build a simple decision tree classifier to categorize AI problems based on a dataset of previous client requests.

Dataset: client_problem.csv

Requirements:

1. Load and preprocess the dataset
2. Create appropriate feature encodings for categorical variables
3. Split the data into training and testing sets
4. Build a decision tree classifier to predict the 'category' based on other features
5. Evaluate the model's performance using accuracy, precision, and recall metrics
6. Visualize the decision tree and interpret how the model makes decisions
7. Identify the most important features in categorizing AI problems

Deliverables:

- Python script with your complete solution
- Brief explanation of your approach and findings (comments in code)
- Visualization of the decision tree

Assignment 2: Intermediate - AI Engineer Matching with Random Forests

Context: The One Hour AI Solution platform needs to match clients with the most suitable AI engineers based on problem requirements and engineer expertise. You'll develop a more robust solution using random forests to handle this complex matching problem.

Task: Implement a random forest model to predict which engineer profiles best match specific client problem descriptions.

Dataset: Use the following datasets:

1. engineer_profiles.csv
2. successful_matches.csv
3. client_problem_id.csv

Requirements:

1. Load and merge the datasets appropriately
2. Engineer relevant features for the matching process
3. Define success metrics (based on satisfaction rating and completion time)
4. Implement a random forest model to predict successful matches
5. Apply pruning techniques to avoid overfitting
6. Evaluate the model using cross-validation
7. Identify the most important features for successful matches
8. Test your model on new, unseen data

Deliverables:

- Python script with your complete solution
- Analysis of feature importance
- Brief report on model performance and insights

Assignment 3: Advanced - One Hour AI Solution Platform Optimization

Context: The One Hour AI Solution platform wants to optimize overall client satisfaction and engineer utilization. You'll develop an advanced random forest system that considers multiple factors to suggest optimal session scheduling and pricing.

Task: Build a comprehensive system using random forests for multiple related prediction tasks (multi-output regression) to optimize the One Hour AI Solution platform's operations.

Dataset: platform_operations.csv

Requirements:

1. Develop a multi-output random forest system that can simultaneously predict:
    - Expected client satisfaction
    - Probability of complete solution delivery
    - Optimal session pricing
    - Client return likelihood
2. Implement feature engineering to extract maximum insights from the provided data, including:
    - Time-based features
    - Engineer-client expertise gap measures
    - Problem complexity relative to expertise
3. Apply advanced pruning and hyperparameter tuning techniques
    - Implement cross-validation with a custom scoring metric
    - Compare different feature selection approaches
4. Build a simulation component that can estimate overall platform performance based on different scheduling and pricing strategies
5. Create a function that can recommend the best engineer for a new client problem while optimizing for multiple objectives

Deliverables:

- Complete Python implementation with well-structured code organization
- Analysis of feature importance across different prediction targets
- Evaluation of model performance with appropriate metrics
- Report on insights and recommendations for platform optimization