Adventist University of Central Africa
P.O. Box 2461 Kigali, Rwanda | www.auca.ac.rw | info@auca.ac.rw

Faculty of Information Technology

Department of Big Data Analytics

Assignment IV
Introduction to statistical Learning: Chapter8

Course: Data Mining and Information Retrieval

Prepared by:
Godfrey Mawulizo (100901)
Shema Hugor (100763)
Shyaka Kevin (100915)
Gumira Theophile (100920)
Niyonsenga Jean Paul (100888)
Nyirmanzi Jean Claude (100882)

Lecturer: Dr. Pacifique Nizeyimana

June 28, 2025

Exercise 8.4, Question 1: Partition and Decision Tree

This exercise requires constructing an example of a partition of a two-dimensional feature space using recursive binary splitting, resulting in at least six regions, and drawing the corresponding decision tree. All regions, cutpoints, and nodes must be labeled, resembling Figures 8.1 and 8.2 from the book. The partition and decision tree are visualized using Python-generated plots, included below.

Partition of the Feature Space

The feature space is a 10×10 square with predictors X_1 (horizontal axis) and X_2 (vertical axis), both ranging from 0 to 10. Recursive binary splitting is applied with the following cutpoints, as defined in the Python code:

- t_1: $X_1 = 5$ (vertical line across the entire space)
- t_2: $X_2 = 7$ (horizontal line from $X_1 = 0$ to $X_1 = 5$)
- t_3: $X_2 = 3$ (horizontal line from $X_1 = 5$ to $X_1 = 10$)
- t_4: $X_1 = 8$ (vertical line from $X_2 = 3$ to $X_2 = 10$)
- t_5: $X_2 = 6$ (horizontal line from $X_1 = 5$ to $X_1 = 8$)

These cutpoints divide the space into six regions, labeled $R_1, R_2, \ldots, R_6$, with boundaries and centers as follows:

Table 1: Regions of the Feature Space

Region	**Boundaries**	**Center**
R_1	$0 \leq X_1 \leq 5$, $0 \leq X_2 \leq 7$	(2.5, 3.5)
R_2	$0 \leq X_1 \leq 5$, $7 < X_2 \leq 10$	(2.5, 8.5)
R_3	$5 < X_1 \leq 10$, $0 \leq X_2 \leq 3$	(7.5, 1.5)
R_4	$5 < X_1 \leq 8$, $3 < X_2 \leq 6$	(6.5, 4.5)
R_5	$5 < X_1 \leq 8$, $6 < X_2 \leq 10$	(6.5, 8.0)
R_6	$8 < X_1 \leq 10$, $3 < X_2 \leq 10$	(9.0, 6.5)

The partition is visualized in Figure 1, generated using Python (`matplotlib`). The plot shows a 10×10 square with vertical lines at $X_1 = 5$ (t_1) and $X_1 = 8$ (t_4), and horizontal lines at $X_2 = 7$ (t_2, from $X_1 = 0$ to 5), $X_2 = 3$ (t_3, from $X_1 = 5$ to 10), and $X_2 = 6$ (t_5, from $X_1 = 5$ to 8). Each region is labeled at its center.

A textual representation of the partition is:

```
X_2
10 |     R_2     |          R_5          |     R_6    |
   |-------------|----------------------|------------|
 7 |     R_1     |          R_5          |     R_6    |
   |-------------|-----------|----------|------------|
 6 |     R_1     |    R_4    |    R_4    |     R_6     |
   |-------------|-----------|----------|------------|
 3 |     R_1     |    R_3    |    R_3    |     R_3     |
   |------------|-----------|---------|-----------|
 0 |    R_1    |    R_3    |    R_3    |    R_3    |
   |------------|-----------|---------|-----------|
      0     5           8            10
         X_1
```

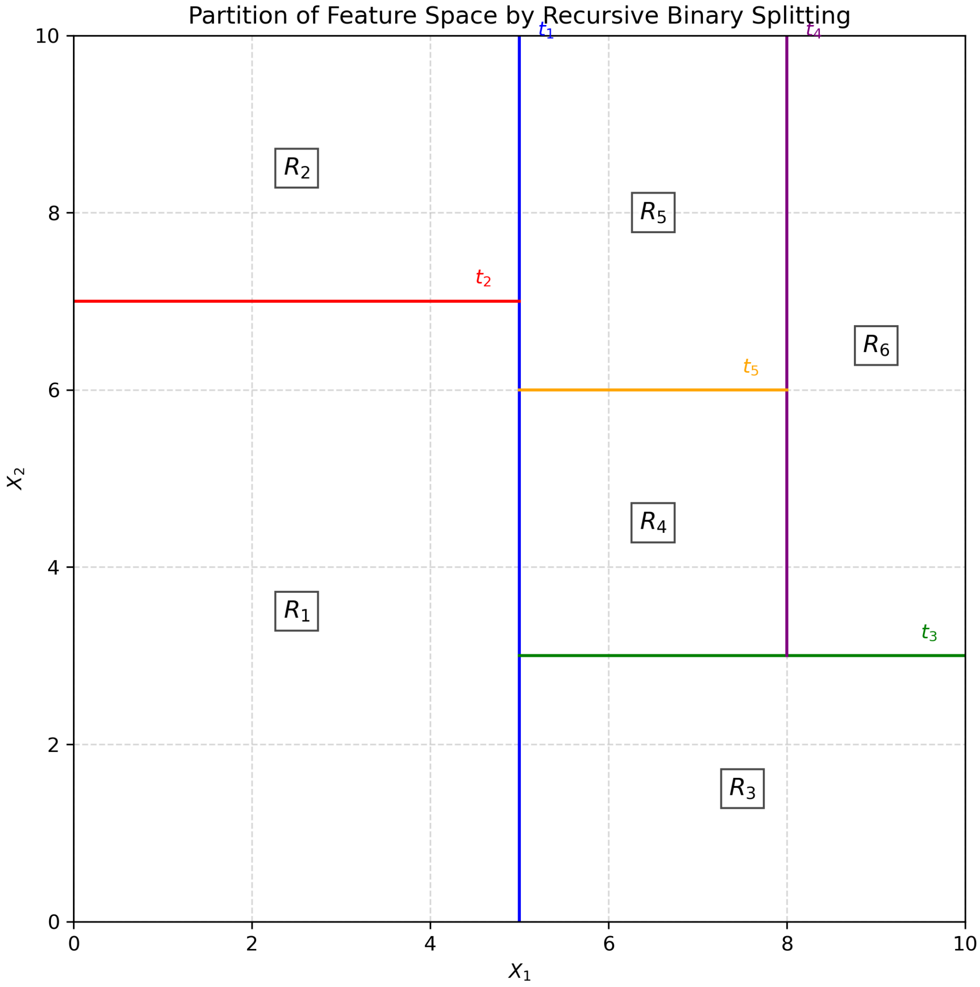

Figure 1: Partition of the feature space by recursive binary splitting, showing six regions (R_1 to R_6) and cutpoints (t_1 to t_5).

Decision Tree

The decision tree corresponding to the partition is shown in Figure 2, generated using Python. It follows the sequence of splits:

- **Root**: Split at $X_1 \leq 5$ (t_1).
- **Left Branch** ($X_1 \leq 5$): Split at $X_2 \leq 7$ (t_2).
  - Left: Region R_1 ($0 \leq X_1 \leq 5, 0 \leq X_2 \leq 7$).
  - Right: Region R_2 ($0 \leq X_1 \leq 5, 7 < X_2 \leq 10$).

- **Right Branch** ($X_1 > 5$): Split at $X_2 \leq 3$ (t_3).
    - Left: Region R_3 ($5 < X_1 \leq 10, 0 \leq X_2 \leq 3$).
    - Right: Split at $X_1 \leq 8$ (t_4).
        - Left: Split at $X_2 \leq 6$ (t_5).
            - Left: Region R_4 ($5 < X_1 \leq 8, 3 < X_2 \leq 6$).
            - Right: Region R_5 ($5 < X_1 \leq 8, 6 < X_2 \leq 10$).
        - Right: Region R_6 ($8 < X_1 \leq 10, 3 < X_2 \leq 10$).

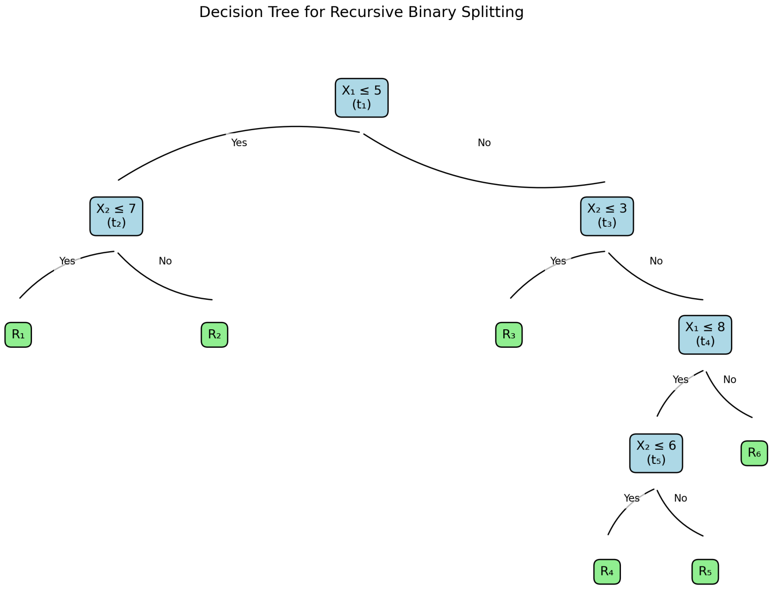

Figure 2: Decision tree corresponding to the feature space partition, with splits labeled t_1 to t_5 and leaves labeled R_1 to R_6.

Exercise 8.4, Question 6: Regression Tree Algorithm

This exercise requires a detailed explanation of the algorithm used to fit a regression tree. Algorithm 8.1 from *An Introduction to Statistical Learning* outlines the process for building a regression tree to predict a continuous response variable Y based on predictors $X_1, X_2, \ldots, X_p$. Below is a step-by-step explanation.

Algorithm 8.1: Building a Regression Tree

The algorithm consists of four steps, applied to a training dataset with n observations $\{(x_i, y_i)\}_{i=1}^n$, where $x_i = (x_{i1}, x_{i2}, \ldots, x_{ip})$ and y_i is continuous.

1. **Grow a Large Tree Using Recursive Binary Splitting**:

- *Objective*: Partition the feature space into regions by minimizing the sum of squared errors (SSE).
- *Process*:
  - Start with all observations in the root node.
  - For each node, evaluate splits on each predictor X_j ($j = 1, \ldots, p$) at split points s:

    $$R_1(j, s) = \{x \mid x_j \leq s\}, \quad R_2(j, s) = \{x \mid x_j > s\}$$

  - Compute SSE:

    $$\text{SSE} = \sum_{i: x_i \in R_1(j,s)} (y_i - \hat{y}_{R_1})^2 + \sum_{i: x_i \in R_2(j,s)} (y_i - \hat{y}_{R_2})^2$$

    where $\hat{y}_{R_m} = \frac{1}{n_m} \sum_{i: x_i \in R_m} y_i$.
  - Choose the (j, s) minimizing SSE, split into two child nodes, and assign observations.
  - Recursively split each child node.
- *Stopping Criterion*: Stop when a node has fewer than a minimum number of observations (e.g., 5).
- *Output*: Large tree T_0 with regions $R_1, \ldots, R_J$.

2. **Apply Cost-Complexity Pruning**:
   - *Objective*: Generate a sequence of subtrees to reduce overfitting.
   - *Criterion*: Minimize the cost-complexity:

     $$C_\alpha(T) = \sum_{m=1}^{|T|} \sum_{i: x_i \in R_m} (y_i - \hat{y}_{R_m})^2 + \alpha |T|$$

     where $|T|$ is the number of leaves, and α is a tuning parameter.
   - *Process*: For each α, find the subtree $T_\alpha \subset T_0$ minimizing $C_\alpha(T)$.
   - *Output*: Sequence of subtrees $T_{\alpha_1}, T_{\alpha_2}, \ldots$.
3. **Choose α Using K-Fold Cross-Validation**:
   - *Process*:
     - Divide data into K folds (e.g., $K = 5$).
     - For each fold $k = 1, \ldots, K$:

       (a) Grow and prune a tree on all but the kth fold.

       (b) Compute mean squared error on the kth fold:

       $$\text{MSE}_k(\alpha) = \frac{1}{n_k} \sum_{i \in \text{fold } k} (y_i - \hat{y}_i)^2$$

  - Average MSE:

$$\mathrm{CV}(\alpha) = \frac{1}{K}\sum_{k=1}^{K} \mathrm{MSE}_k(\alpha)$$

  - Choose α minimizing $\mathrm{CV}(\alpha)$.

- *Output*: Optimal α.

4. **Return the Optimal Subtree**:
    - Select subtree T_α for the chosen α.
    - Predict $\hat{y}_{R_m}$ for observation x in region R_m.
    - *Output*: Pruned tree T_α.

Key Features

- **Greedy Approach**: Local SSE minimization ensures efficiency.
- **Overfitting Control**: Pruning and cross-validation improve generalization.
- **Interpretability**: The tree is a sequence of decisions, as in Question 1.