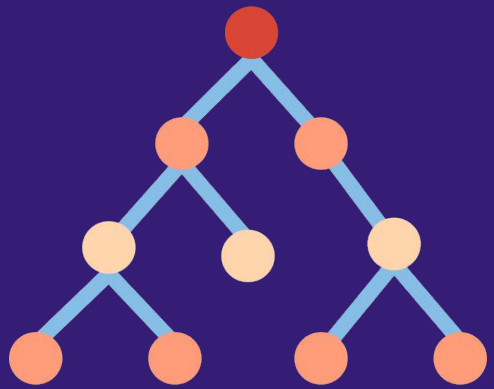

DATA2060 | CART for Regression

Group Name: Print (“Hello World”)
Group Members: Matthew Dall’Asen, Arpit Dang, Varun Satheesh, Devraj Raghuvanshi
Data Science Institute, Brown University
Date: December 10, 2024

Introduction | Overview

The objective of this report is to develop, implement and test a Classification and Regression Tree (CART) algorithm for regression problems. This algorithm uses a decision tree like structure to predict regression problems.

Advantages

✔ **Non-linearity Handling:** Models complex, non-linear relationships between features and the target variable

✔ **Feature Selection:** Splits only on relevant features, reducing the risk of overfitting to irrelevant variables

✔ **Interpretability:** Easy to understand and interpret, with clear decision paths that explain predictions

Disadvantages

X **Overfitting:** Prone to overfitting, especially when the tree grows too deep

X **Instability:** Small changes in the training data can result in significantly different trees because of its greedy nature during splitting

X **Bias in Splitting Criteria:** May favour features with more levels or categories when splitting

Mathematical Model | Representation & Loss Function

Regression Decision Tree

Representation

Decision Tree: Despite being a regression problem, this is a binary decision tree. At each node, the tree partitions the input space based on a single feature (X = Rd) at a specific threshold that minimizes the loss function:

$$\text{Direction}(\mathbf{x}_i, N) = \begin{cases} \text{left child node}, & x_{ij} \leq t \\ \text{right child node}, & x_{ij} > t \end{cases}$$

For any data point that reaches a specific leaf node, the prediction (Y = R) is the mean of the target values for all training observations in that leaf:

$$\hat{y}_k = \frac{1}{n_k} \sum_{m \in L_k} y_m$$

Loss Function

Loss Function: A Regression Tree uses Sum of Squared Errors (SSE) as its loss function. SSE measures the total squared difference between the actual target values and the predicted values within each node:

$$\text{SSE} = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

Where:

- N is the total number of observations
- Y_i is the target value of the ith sample
- Y is the predicted value for the ith sample given the learned model weights

Mathematical Model | Optimization

Primary Goal of Optimization: To split the data at each node into two subsets where we can *minimize* the SSE (Sum of Squared Errors).

We use a *Greedy Algorithm*, which selects the best split at each step to minimize the SSE.

Greedy Algorithm: It is a problem-solving approach where at each step, the *logically optimal choice* is made without future implications down the line.

What is the choice?
We assess all possible splits across every feature and potential split thresholds, selecting the combination of feature and threshold that yields the *highest possible gain* for splitting the data.

Mathematical Model | Optimization (cont'd)

What is Gain?

Gain measures the quality of the split in the decision tree's ability to predict the target variable.

Gain is calculated as the reduction in the SSE between the parent node and its child nodes. The feature and the split threshold that result in the *highest gain* (highest reduction in SSE) are selected for the split to occur. Minimizing SSE reduces the overall variance in target values, resulting in a more accurate prediction.

How do we determine Gain?

j = feature t = split threshold

- **Group 1**: $D_1 = \{(\mathbf{x}_i, y_i) \mid x_{ij} \leq t\}$, containing all data points where the j-th feature value is less than or equal to t.
- **Group 2**: $D_2 = \{(\mathbf{x}_i, y_i) \mid x_{ij} > t\}$, containing all data points where the j-th feature value is greater than t.

$$\text{SSE}(D_1) = \sum_{(\mathbf{x}_i, y_i) \in D_1} (y_i - \hat{y_{D_1}})^2$$

$$\text{SSE}(D_2) = \sum_{(\mathbf{x}_i, y_i) \in D_2} (y_i - \hat{y_{D_2}})^2$$

$$\text{Gain} = \text{SSE}_{\text{parent}} - (\text{SSE}_{\text{left child}} + \text{SSE}_{\text{right child}})$$

$$\text{Best Split} = \arg\max_{t,j} \text{Gain}(t, j)$$

Mathematical Model | Optimization (cont'd)

Stopping Criteria

- **Maximum Tree Depth:** A predefined maximum depth limits the growth of the tree. Once the limit has been reached, no further splits are performed. Prevents from making deep and complex trees. Also, prevents the likelihood of high variance and overfitting.

- **Minimum Samples per Leaf:** If the number of samples within a node is below a specified threshold, the split does not happen. This prevents the likelihood of high variance and overfitting.

- **Homogenous Target Values:** If all the target values in a node are identical, no further splits are made (SSE (D) = 0 OR Gain < specific threshold). This ensures that the tree does not split unnecessarily.

Implementation | Pseudocode

Class Node:
- Represents a tree node with attributes:
  - `left`, `right` (child nodes)
  - `depth`, `index_split_on`, `threshold`
  - `isleaf` (is it a leaf?)
  - `pred` (mean prediction)
  - `info` (split details like gain and sample count)

Class CART:
- Initializes tree with:
  - Root node using the dataset mean
  - `min_samples_split` (stop splitting below this sample count)
  - `max_depth` (maximum depth of the tree)
  - Recursive splitting starting from root

Implementation | Pseudocode

Methods:

- `_predict_recurs(features)`:
  - *Traverse tree recursively, return prediction at leaf node.*

- `_variance(values)`:
  - *Compute variance of a list of values.*

- `_calc_gain(data, split_index, threshold)`:
  - *Calculate variance reduction from splitting the data.*

- `_split_recurs(node, data, indices)`:
  - *Check if node is terminal (empty, uniform target values, max depth, or insufficient samples).*
  - *If not terminal:*
    - *Iterate over features and potential split points.*
    - *Compute gain for each split and find the best one.*
    - *Store split info in the node.*
    - *Split data into left and right subsets.*
    - *Recur on child nodes.*

Implementation | Data & Results

Dataset: Forest Fires (UCI ML Repository)

Paper: A Data Mining Approach to Predict Forest Fires using Meteorological Data

Cortez et al. (2007)

→ 517 samples
→ 12 features

Target variable - Area burnt by forest fires

Evaluation metric: Mean Absolute Error (MAE)

$$\mathrm{MAE} = \frac{1}{n}\sum_{i=1}^{n} | x_i - x |$$

Implementation | Data & Results

Start

- → **[Data Preprocessing]**
  - → One-hot encode categorical features
  - → Standardize all the features (StandardScaler)
- → **[Loop through 30 random states]**
  - → **[K-Fold Cross-Validation (10 folds)]**
    - → For each fold:
      - → Split data into training and test sets
      - → Train model on training data
      - → Make predictions on test set
      - → Calculate MAE for the fold
  - → **M_{rs} = [Average MAE over all 10 folds]**
- → **Final Score = [Average M_{rs} over all 30 random states]**

End

Implementation | Data & Results

Start

- → **[Data Preprocessing]**
  - → One-hot encode categorical features
  - → Standardize all the features (StandardScaler)
- → **[Loop through 30 random states]**
  - → **[K-Fold Cross-Validation (10 folds)]**
    - → For each fold:
      - → Split data into training and test sets
      - → Train model on training data
      - → Make predictions on test set
      - → Calculate MAE for the fold
  - → **M_{rs} = [Average MAE over all 10 folds]**
- → **Final Score = [Average M_{rs} over all 30 random states]**

End

Mean Absolute Error (MAE)

Paper	13.46 ± 0.04
Our	13.61 ± 0.38

Hyperparameters

→ max_depth: 5
→ min_sample_split: 42

Implementation | Data & Results

Additional Results:

Dataset	MSE (sklearn's DTR)	MSE (Our)
California Housing	0.5245	0.5245
Diabetes	3358.6384	3378.0334

Mean Squared Error (MSE) comparison between sklearn's Decision Tree Regressor (DTR) and Our Implementation

Hyperparameters

→ max_depth: 5

→ min_sample_split: 20

Conclusion | Final Remarks & Challenges

✅ Implemented CART regression tree from scratch

What Was Interesting?

- **Interpretable predictions:** Provides clear, interpretable rules for prediction, making it easier to understand decision boundaries.
- **Greedy approach:** Greedily selects the feature and threshold that maximize gain, optimizing the tree's predictive power.

Challenges Faced:

- **Recursion:** Implementing recursive splitting in code.
- **Limited resources:** Finding academic resources to understand the algorithm.
- **Creating toy datasets for the unit tests.**

Thank You!

Questions?

Bibliography

[1] Cortez, P. and Morais, A.D.J.R., 2007. A data mining approach to predict forest fires using meteorological data.

[2] Breiman, L., 2017. *Classification and regression trees*. Routledge.

[3] Scikit-learn: Tree Algorithms (CART)