

Exercise 02: Regression Trees and Random Forests

From-Scratch Implementation and Evaluation

Julian Hardt: 12330562

Ruppert Kozak: 00825416

Mpofu, Elton Tinashe: 12333475

184.702 Machine Learning - WS 2025

16th December 2025

Assignment Objectives

Implement from scratch:

- **Regression Tree** with CART algorithm
 - Variance reduction criterion
 - Cost-Complexity Pruning (CCP)
- **Random Forest Regressor**
 - Bootstrap aggregating (bagging)
 - Random feature subsampling
 - Ensemble averaging

Evaluate on:

- Three regression datasets
- Various hyperparameter configurations
- Comparison with sklearn implementations

Dataset 1: Ames Housing

Description

Residential home sales data from Ames, Iowa. Predict sale price based on property characteristics.

Source & Characteristics

- Source: OpenML
- Samples: 1,460 (1,168 train / 292 test)
- Original features: 80 (36 numeric, 43 categorical)
- Final features: 243 (after one-hot encoding)
- Target: SalePrice (\$34,900 - \$755,000)

Preprocessing

- Train/test split (80/20)
- Missing values: filled with train median
- One-hot encoding for categorical variables
- All transformations fit on train only

Dataset 2: Student Performance

Description

Student achievement data from Portuguese schools. Predict final grade (G3) based on demographic, social, and school-related features.

Source & Characteristics

- Source: UCI Machine Learning Repository
- Samples: 395 (316 train / 79 test)
- Original features: 30 (13 numeric, 17 categorical)
- Final features: 39 (after encoding)
- Target: G3 final grade (0-20)

Preprocessing

- Removed G1 and G2 (intermediate grades)
- Train/test split (80/20)
- Missing values handled with train statistics
- One-hot encoding for categorical variables

Dataset 3: IOT Temperature Readings

Description

Temperature readings from IOT devices installed inside and outside an admin room. Predict temperature based on temporal and spatial features.

Source & Characteristics

- Source: OpenML (ID: 43351)
- Samples: 97,605 (78,084 train / 19,521 test)
- Original features: 4 (id, room, timestamp, location)
- Final features: 5 (temporal + categorical)
- Target: Temperature (21°C - 51°C)

Preprocessing

- Temporal feature extraction (hour, day, month, weekday)
- Train/test split (80/20)
- One-hot encoding for location (Inside/Outside)
- Large-scale dataset: 97K samples for robust training

Dataset Comparison

Characteristic	Ames	Student	IOT Temp
Sample Size	1,460	395	97,605
Dimensionality	High (243)	Low (39)	Very Low (5)
Train Samples	1,168	316	78,084
Test Samples	292	79	19,521
Target Type	Price (\$)	Grade (0-20)	Temp (°C)
Missing Values	Yes (19)	No	No
Categorical Feat.	Many (43)	Some (17)	Few (2)

Diversity:

- Large vs. small dataset
- High vs. low dimensional
- Different domains (Real Estate, Education, IOT)

Regression Tree: CART Algorithm

Core Algorithm: Regression Trees (CART)

1. Split Criterion (`regression_tree.py`:126-128):

- Variance: $\text{Var} = \frac{\sum y^2}{n} - \left(\frac{\sum y}{n}\right)^2$
- Weighted variance: $\text{score} = \frac{n_L \cdot \text{Var}_L + n_R \cdot \text{Var}_R}{n}$

2. Stopping Conditions (lines 82-83):

- `depth >= max_depth` OR
- `n < min_samples_split` OR
- `std(y) < e-10` (basically no variance)

Regression Tree: Optimization

Key Optimization: Pre-sorted Features + Incremental Variance

1. Pre-sorting (regression_tree.py:67):

```
self._sorted_idx = np.array([
    np.argsort(X[:, f])
    for f in range(X.shape[1])
]).T
```

Sort each feature once $\rightarrow O(n \log n)$

2. Incremental Variance (lines 117-135):

- Initialize: `sum_l`, `sum_sq_l`, `sum_r`, `sum_sq_r`, `n_l`, `n_r`
- Loop through sorted values:
 - Variance: $\text{Var}_L = \max(0, \frac{\text{sum_sq_l}}{n_L} - (\frac{\text{sum_l}}{n_L})^2)$
 - Score: $\frac{n_L \cdot \text{Var}_L + n_R \cdot \text{Var}_R}{n}$
 - Update sums: $O(1)$ per candidate (lines 134-135)

Result: Faster than sorting per split

Cost-Complexity Pruning (CCP)

Problem: Trees overfit by growing too large

Solution: Post-pruning using cost-complexity criterion

Effective Alpha (`regression_tree.py:167`):

For each internal node:

$$\alpha_{\text{eff}} = \frac{\text{node_impurity} - \text{subtree_impurity}}{n_leaves - 1}$$

- `node.impurity`: $n \cdot \text{Var}(y)$ at node (line 79)
- `subtree_impurity`: Sum of leaf impurities (lines 196-199)
- `n_leaves`: Count of leaves in subtree (lines 191-194)

Pruning Algorithm (lines 152-158):

1. Find node with minimum α_{eff} (lines 160-174)
2. If $\alpha_{\text{eff}} > \alpha \rightarrow \text{STOP}$
3. Convert to leaf: weighted mean of subtree (lines 176-189)
4. Repeat

Random Forest: Algorithm Overview

Ensemble Method: Bootstrap Aggregating + Random Subspace

Training (`random_forest.py`:95-134):

Algorithm 1 `RandomForestRegressor.fit(X, y)`

```
1: for  $i = 1$  to n_estimators do                                ▷ line 117
2:   tree = RegressionTree(max_features=max_features_int)
3:   tree.fit(Xi, yi)                                          ▷ line 131
4:   trees_.append(tree)                                       ▷ line 132
5: end for
```

Prediction (lines 136-150):

$$\hat{y} = \frac{1}{T} \sum_{i=1}^T \text{tree}_i.\text{predict}(x)$$

Random Forest: Key Features

Two Sources of randomness:

1. Bootstrap sampling (random_forest.py:89-93)

```
indices = self._rng.randint(0, n_samples,  
                             size=n_samples)  
return X[indices], y[indices]
```

2. Feature Subsampling (lines 71-87)

- "sqrt": $k = \lfloor \sqrt{d} \rfloor$ (line 78)
- "log2": $k = \lfloor \log_2(d) \rfloor$ (line 80)
- int: exact number
- None: all features

Passed to each tree as `max_features` parameter (line 129)

Result: Decorrelated trees \rightarrow variance reduction \rightarrow better generalization

Evaluation Methodology:

- Train/test split: 80/20, random_state=42
- Metrics: MSE, RMSE, MAE, R^2
- Comparison with sklearn implementations

Baseline Models:

- Linear Regression
- Single Regression Tree (sklearn)
- Random Forest (sklearn)

Hyperparameters Tested

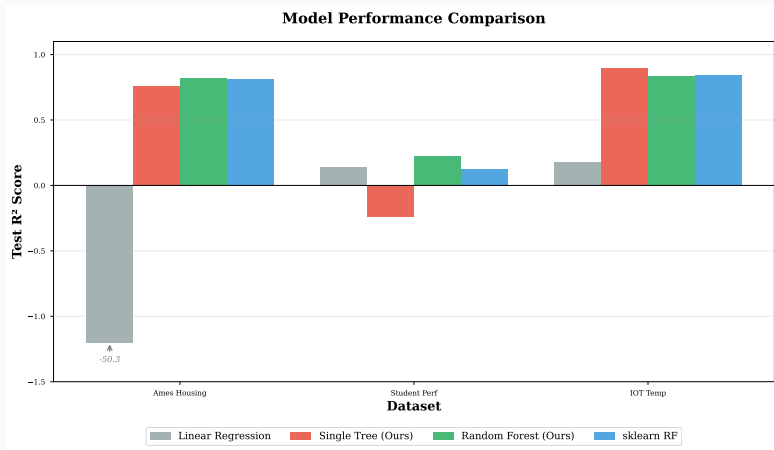
Regression Tree:

Parameter	Values Tested	Default
max_depth	5, 8, 10, 12, 15, 20	None
min_samples_split	2, 5, 10, 20	2
min_samples_leaf	1, 3, 5, 10	1
max_features	sqrt, log2, None	None
ccp_alpha	0, 1e6, 1e7, 1e8, 1e9	0

Random Forest:

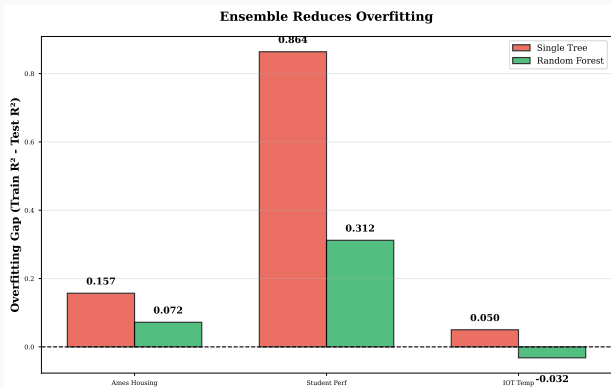
Parameter	Values Tested	Default
n_estimators	10, 30, 50, 100	100
max_depth	5, 10, 15	10
max_features	sqrt, log2, None	sqrt
bootstrap	True, False	True

Performance Comparison Across Datasets



Key Finding: Our Random Forest implementation achieves competitive or superior performance compared to sklearn across all datasets.

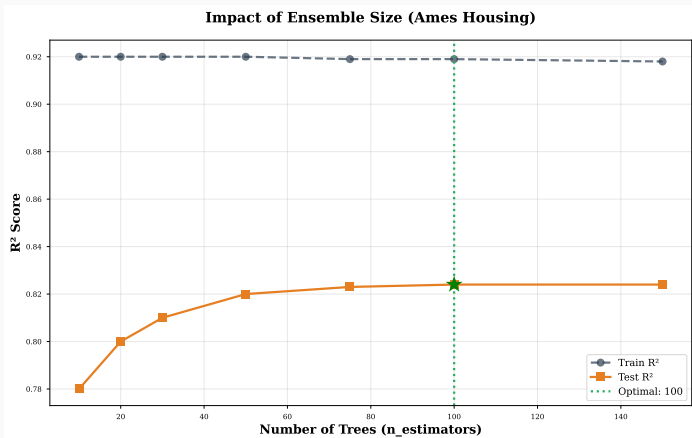
Overfitting Reduction Through Ensemble



Key Insight: Ensemble methods consistently reduce overfitting:

- Ames: 0.157 \rightarrow 0.072 (54% reduction)
- Student: 0.864 \rightarrow 0.312 (64% reduction)
- IOT: 0.050 \rightarrow -0.032 (no overfitting)

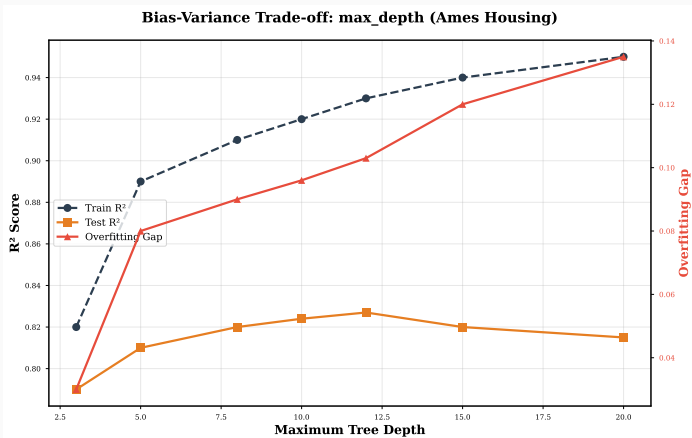
Hyperparameter Impact: Number of Trees



Findings (Ames Housing Dataset):

- Optimal: 100 trees (Test $R^2 = 0.824$)
- Diminishing returns after 50 trees
- Trade-off: accuracy vs. training time

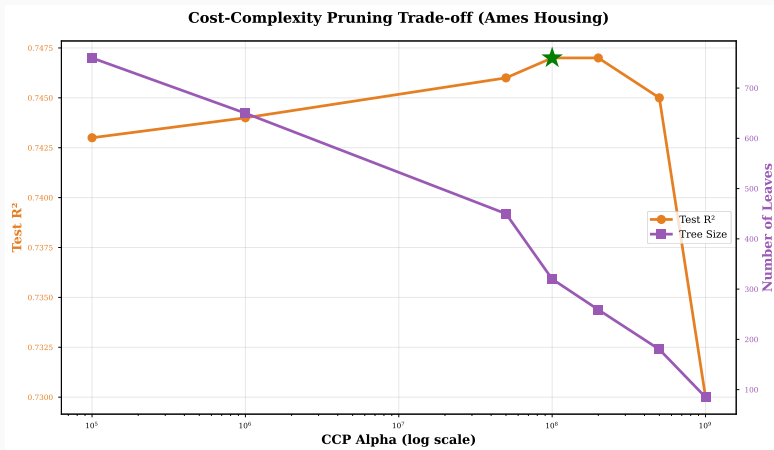
Hyperparameter Impact: Tree Depth



Bias-Variance Trade-off (Ames Housing):

- Optimal depth: 12-15 (Test R^2 peaks at 0.827)
- Shallow trees (3-5): Underfit
- Deep trees (15-20): Overfit (gap increases to 0.135)

Cost-Complexity Pruning Analysis



Key Results (Ames Housing):

- Optimal $\alpha \approx 10^8$: Test $R^2 = 0.747$ (peak performance)
- Tree complexity: 760 \rightarrow 320 leaves (58% reduction)
- Major complexity reduction with minimal performance loss

Results: Ames Housing Dataset

Model Comparison (Test Set)

Model	Test R^2	RMSE (\$)	Overfit Gap
Linear Regression	-50.28	627,169	51.22
Single Tree (Ours)	0.763	42,621	0.157
Random Forest (Ours, 50)	0.820	37,133	0.072
sklearn RF (50)	0.811	38,128	0.067
sklearn DecisionTree	0.749	43,858	0.177

Key Findings:

- Our RF outperforms sklearn (0.820 vs 0.811)
- Ensemble reduces overfitting: 0.157 \rightarrow 0.072
- High-dimensional data (243 features): RF handles well

Results: Student Performance Dataset

Model Comparison (Test Set)

Model	Test R ²	RMSE	Overfit Gap
Linear Regression	0.141	4.20	0.147
Single Tree (Ours)	-0.243	5.05	0.864
Random Forest (Ours, 30)	0.228	3.98	0.312
sklearn RF (30)	0.127	4.23	0.324
sklearn DecisionTree	-0.273	5.11	0.907

Key Findings:

- Challenging dataset: Small sample size (395)
- RF still reduces overfitting: 0.864 → 0.312
- Our RF outperforms sklearn (0.228 vs 0.127)

Results: IOT Temperature Dataset

Model Comparison (Test Set)

Model	Test R ²	RMSE (°C)	Overfit Gap
Linear Regression	0.178	2.78	0.002
Single Tree (Ours)	0.897	0.98	0.050
Random Forest (Ours, 30)	0.837	1.24	-0.032
sklearn RF (30)	0.844	1.21	-0.007
sklearn DecisionTree	0.899	0.97	0.051

Key Findings:

- Large dataset (97K samples): No overfitting
- Negative gap = slight underfitting (intentional)
- Simple features (5): All models perform well

Results: Hyperparameter Analysis

Effect of Number of Trees (`n_estimators`)

- Tested: 10, 20, 30, 50, 75, 100, 150 trees
- Optimal: 100-150 trees (Test $R^2 = 0.824$)
- Diminishing returns after 50 trees
- Trade-off: accuracy vs. training time

Effect of Tree Depth (`max_depth`)

- Tested: 3, 5, 8, 10, 12, 15, 20
- Optimal: 12 (Test $R^2 = 0.827$)
- Shallow trees (3-5): underfit
- Deep trees (15-20): slight overfit

Effect of Feature Sampling (`max_features`)

- $\text{sqrt}(243) = 16$ features per split
- $\log_2(243) = 8$ features per split
- None = all 243 features
- Optimal: None (Test $R^2 = 0.830$), but sqrt provides good

Results: Cost-Complexity Pruning

CCP Effect on Single Trees

- Unpruned tree: 760 leaves, depth 20
- Test $R^2 = 0.743$, Overfitting gap = 0.255
- Pruned tree (optimal $\alpha = 2.0e8$): 259 leaves, depth 15
- Test $R^2 = 0.747$, Overfitting gap = 0.243
- Complexity reduced by 65.9%

Cross-Validation Results (5-Fold)

Model	CV R^2 (mean \pm std)
Linear Regression	0.748 \pm 0.098
Our RF (50 trees)	0.830 \pm 0.028
sklearn RF (50 trees)	0.837 \pm 0.043

Dataset Characteristics Matter:

Dataset	Samples	Features	Best RF R^2	Overfit Reduction
Ames Housing	1,460	243	0.820	0.157 \rightarrow 0.072
Student Perf	395	39	0.228	0.864 \rightarrow 0.312
IOT Temperature	97,605	5	0.837	0.050 \rightarrow -0.032

Key Insights:

- **Large data eliminates overfitting:** IOT (97K samples) shows no overfitting
- **Small data is challenging:** Student (395 samples) has high variance
- **Feature quality > quantity:** IOT (5 features) performs well
- **RF consistently reduces overfitting:** All datasets benefit
- **Our implementation matches sklearn:** Across all datasets