Big Data in Finance

Week 3: Tree-Based Regression Methods

Dr Daniele Bianchi

Spring 2026

Today's Agenda

From Linear to Non-Linear Models

Regression Trees

Ensemble Learning: Bagging, Random Forests, and Boosting

Revisiting Market Timing

Implementing Regression Trees

Out-of-Sample Statistical Performance

Economic Performance

Summary and Next Steps

From Linear to Non-Linear Models

Recap: Linear Regression Methods

Lecture 2: Given that $f(X) = X^\top \boldsymbol{\beta}$, how do we estimate $\boldsymbol{\beta}$ well?

- **OLS:** Unbiased but high variance in high dimensions (overfitting)
- **Ridge:** Shrinks coefficients toward zero (bias for variance)
- **Lasso:** Shrinks + selects variables (sparse solutions)
- **Elastic Net:** Combines L1 and L2 penalties

Common Assumption!

All methods assume the true relationship is **linear in the set of features**

What If Linearity Fails?

This Lecture: What if $f(X) \neq X^\top \beta$? How do we let the **data reveal** the functional form?

Limitations of linear models:

1. **Linearity:** Effect of X_j on Y is constant across all values of X_j
2. **Additivity:** Effect of X_j does not depend on values of other predictors
3. **Global model:** Same coefficients everywhere in feature space

We can add polynomials (X_j^2) and interactions ($X_j \cdot X_i$) manually, but:

- Which ones? How many?
- Model dimension explodes with many predictors

Think of Equity Premium Prediction

Goyal-Welch setting: Predict excess market return using macro predictors

- **Non-linear effects:**
  - Dividend yield may predict differently at extremes
  - Valuation ratios: effect may saturate
- **Interactions matter:**
  - D/P predicts better when term spread is high?
  - Inflation interacts with real rates
- **Regime dependence:**
  - Predictability varies: recessions vs. expansions
  - Different models for different market states

Key Question

Can we let the data **discover** which predictors matter where?

The Tree Alternative

Core idea: Partition the feature space into regions, fit simple models within each region

Linear Model:

$$\widehat{f}(x) = \widehat{\beta}_0 + \sum_{j=1}^{p} \widehat{\beta}_j x_j$$

- Global: same β everywhere
- Smooth predictions
- Extrapolates beyond data

Tree Model:

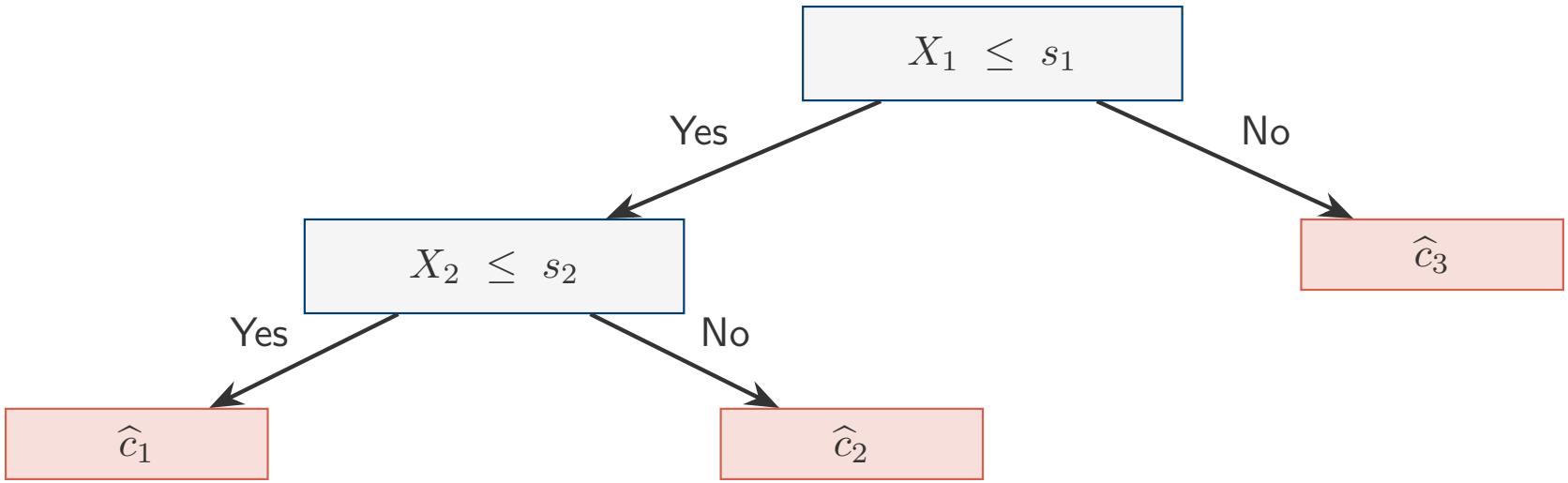
$$\widehat{f}(x) = \sum_{m=1}^{M} \widehat{c}_m \cdot \mathbf{1}(x \in R_m)$$

- Local: different prediction by averaging returns in a given region R_m
- Piecewise constant
- Cannot extrapolate

Important Trade-off: Flexibility vs. smoothness

Regression Trees

Decision Trees: The Basic Idea



- **Internal nodes:** Binary split on one variable
- **Terminal nodes (leaves):** Predict constant $\widehat{c}_m = \overline{y}_{R_m}$ (mean of training y in region)

The Prediction Function

A regression tree partitions feature space into M disjoint regions $R_1, \ldots, R_M$

Prediction function:

$$\widehat{f}(x) = \sum_{m=1}^{M} \widehat{c}_m \cdot \mathbf{1}(x \in R_m)$$

Optimal constant within each region:

$$\widehat{c}_m = \frac{1}{n_m} \sum_{x_i \in R_m} y_i = \overline{y}_{R_m}$$

(minimizes squared error within region)

Key Property

Given the partition $\{R_m\}$, the optimal predictions are just **regional averages**

Finding the Optimal Partition

Ideal objective: Find partition minimizing total RSS

$$\min_{\{R_m\}_{m=1}^{M}} \sum_{m=1}^{M} \sum_{x_i \in R_m} (y_i - \widehat{c}_m)^2$$

Problem: Computationally infeasible to search all possible partitions

Solution: Greedy recursive binary splitting

- Start with all data in one region
- Find the **single best split** (one variable, one threshold)
- Repeat recursively on each resulting region
- Stop when criterion met (e.g., minimum leaf size).

What Does "Greedy" Mean?

Pick the best split *right now* without looking ahead—like always taking the immediate shortcut without checking if a longer route would be faster overall

The Splitting Criterion

At each node, find split (j, s) minimizing:

$$\min_{j,s} \left[\sum_{x_i \in R_1(j,s)} (y_i - \widehat{c}_1)^2 + \sum_{x_i \in R_2(j,s)} (y_i - \widehat{c}_2)^2 \right]$$

where:

$$R_1(j, s) = \{X \mid X_j \leq s\} \quad \text{(left child)}$$
$$R_2(j, s) = \{X \mid X_j > s\} \quad \text{(right child)}$$

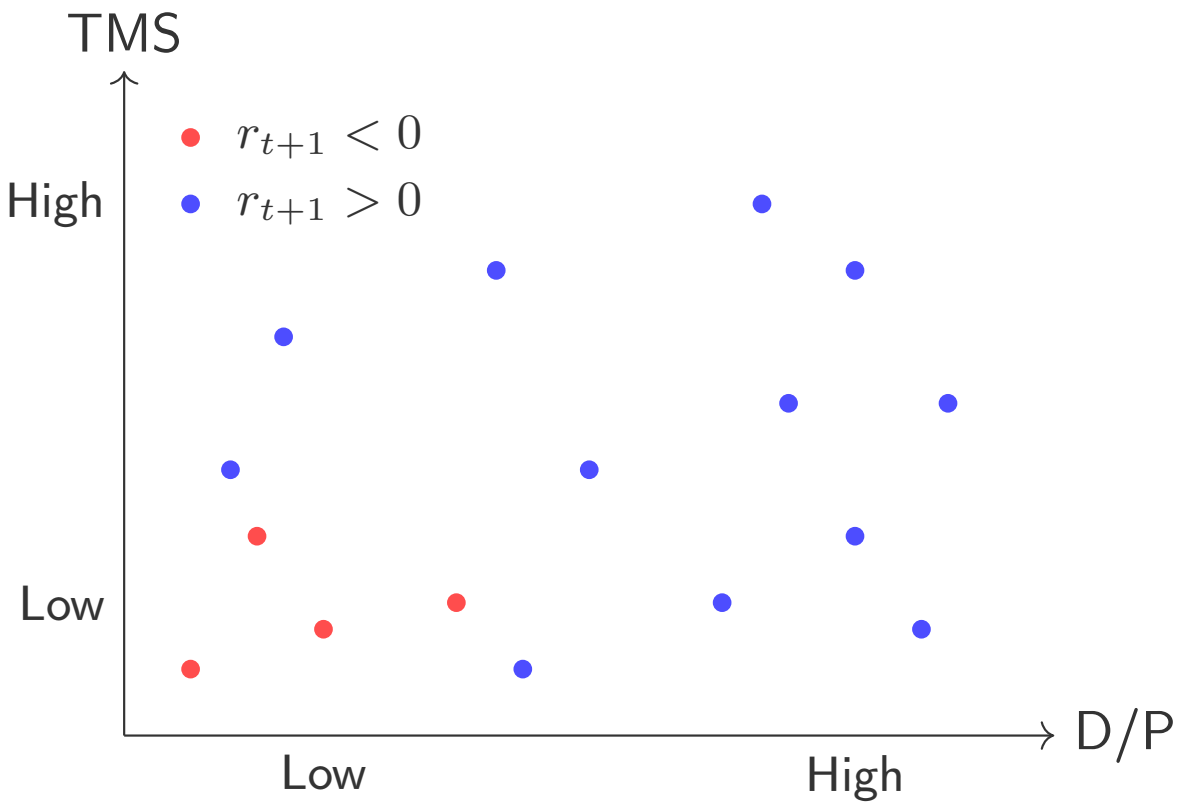
Algorithm:

1. For each predictor $j = 1, \ldots, p$:
   - For each unique value s of X_j in the data:
   - Compute RSS reduction from split (j, s)
2. Select (j^*, s^*) with largest RSS reduction

Complexity: $O(np)$ per split (efficient!)

Visual Example: Equity Premium with Two Predictors

Setup: Predict excess return r_{t+1} using D/P (dividend-price) and TMS (term spread)

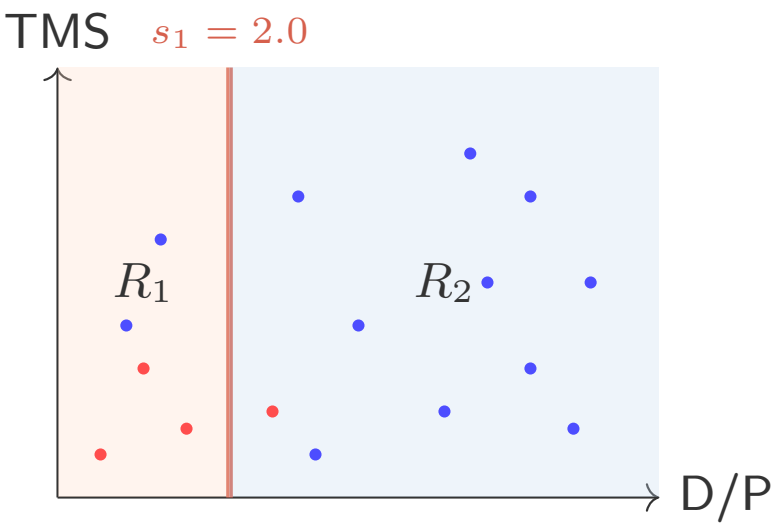


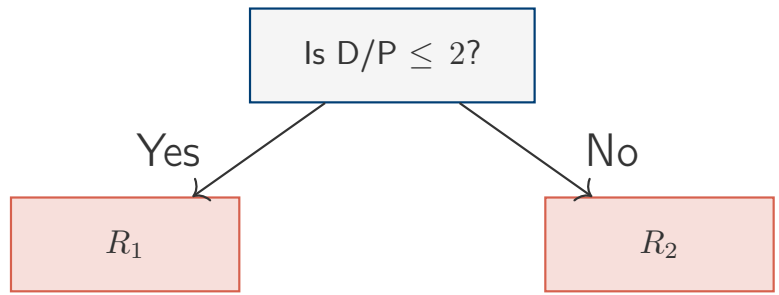
Question: How should we partition this space to predict returns?

Step 1: First Split

Algorithm searches all variables and thresholds:

- Best split: D/P ≤ 2.0



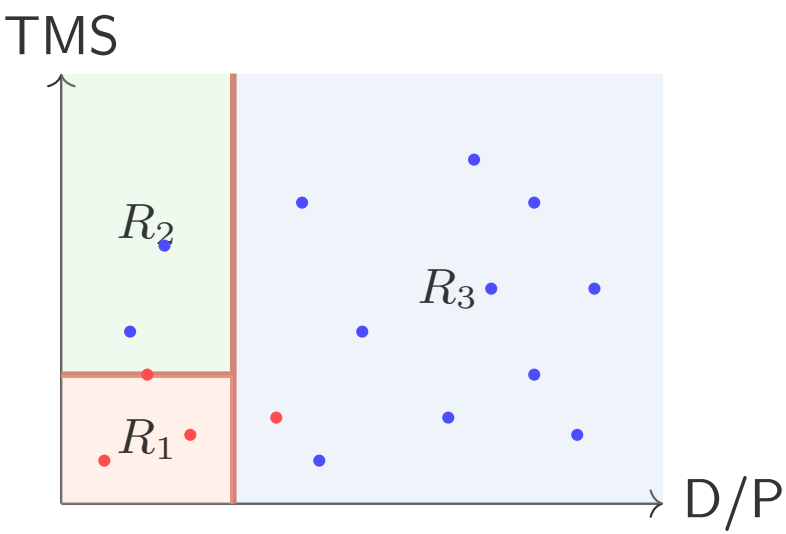


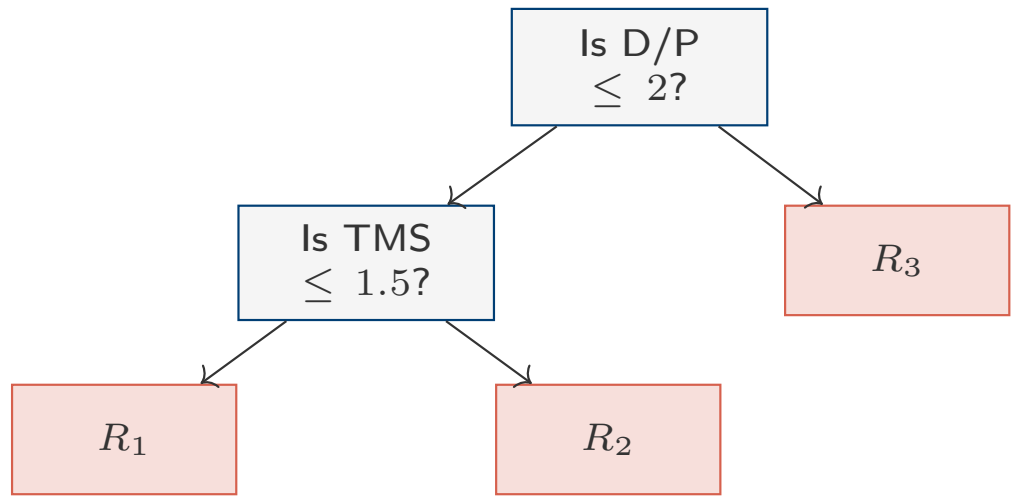
R_1: Low D/P
(mostly negative returns)
R_2: High D/P
(mostly positive returns)

Step 2: Split Left Region

Now split R_1 (low D/P region):

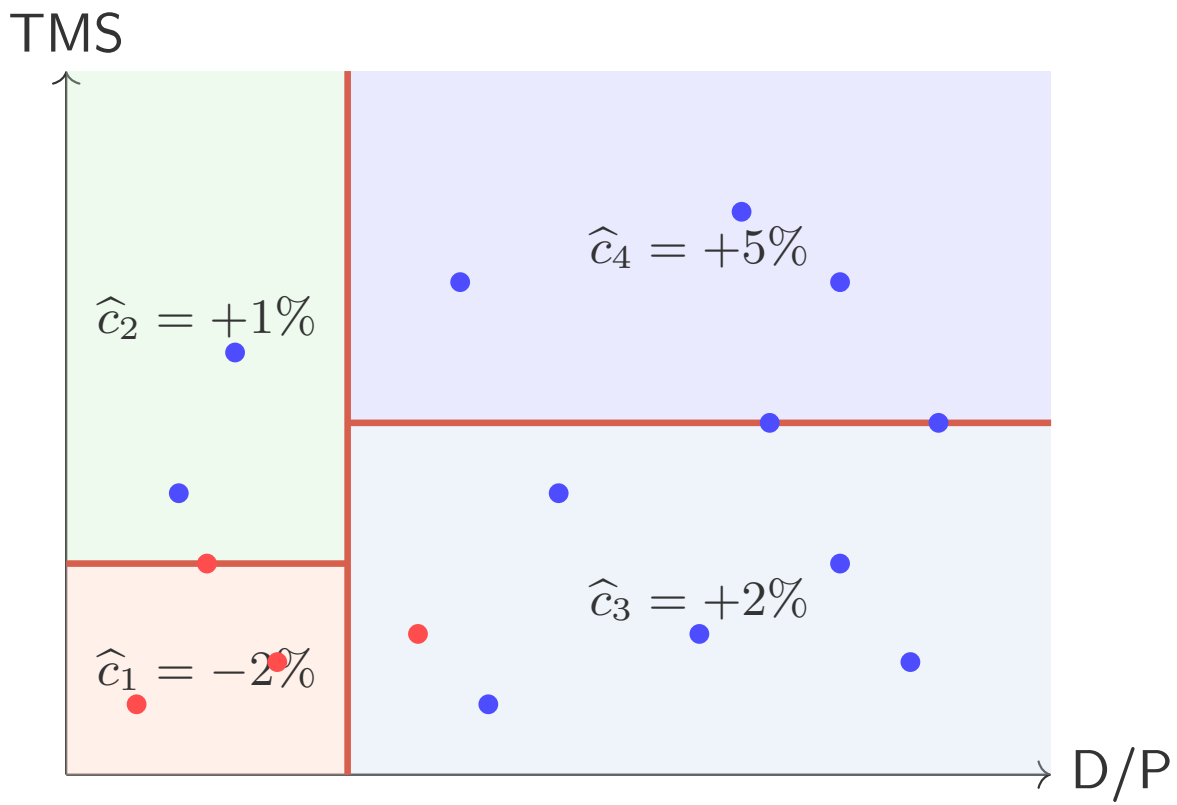
- Best split: TMS ≤ 1.5





R_1: Low D/P, Low TMS (predict: -2%)
R_2: Low D/P, High TMS (predict: $+1\%$)
R_3: High D/P (predict: $+3\%$)

Tree Partitions are Axis-Aligned Rectangles



Key properties:

- Splits are always perpendicular to axes (one variable at a time)
- Regions are rectangular (in any dimension)
- Prediction is **constant within each region**

Interpretation: What the Tree Learned

The fitted tree says:

1. High D/P $\rightarrow$ positive returns
2. Low D/P alone $\rightarrow$ depends on TMS
3. Low D/P + Low TMS $\rightarrow$ negative returns
4. Low D/P + High TMS $\rightarrow$ mildly positive

Compare to linear model:

$$\widehat{r}_{t+1} = \widehat{\beta}_0 + \widehat{\beta}_1 \text{D/P} + \widehat{\beta}_2 \text{TMS}$$

- Effects are additive
- Same β_2 everywhere
- Would need to specify D/P $\times$ TMS interaction manually

Interaction discovered:
TMS matters *only when* D/P is low

Trees Automatically Discover

- Non-linear effects (different predictions at different D/P levels)
- Interactions (TMS effect depends on D/P)

When to Stop Splitting?

Stopping rules (hyperparameters):

- **Minimum samples per leaf** (`min_samples_leaf`)
  - Stop if split would create node with fewer than k observations
  - Typical: 1–20 for regression
- **Maximum depth** (`max_depth`)
  - Limit number of sequential splits
  - Typical: 3–20
- **Minimum impurity decrease** (`min_impurity_decrease`)
  - Stop if RSS reduction below threshold

Problem: Early stopping can miss good splits that only become valuable after further splitting

Cost-Complexity Pruning

Better approach: Grow large tree, then prune back

Cost-complexity criterion:

$$C_\alpha(T) = \sum_{m=1}^{|T|} \sum_{x_i \in R_m} (y_i - \widehat{c}_m)^2 + \alpha|T|$$

- $|T|$ = number of terminal nodes (tree complexity)
- $\alpha \geq 0$ = complexity penalty (tuning parameter)

Compare to Lasso

Lasso: $\text{RSS} + \lambda \sum_j |\beta_j|$

Tree: $\text{RSS} + \alpha|T|$

Both trade off fit vs. complexity; α selected via cross-validation

Bias-Variance Trade-off for Trees

Deep trees:

- Many small regions
- Low bias (can fit complex patterns)
- **High variance** (sensitive to training data)
- Risk: overfitting

Shallow trees:

- Few large regions
- **High bias** (may miss patterns)
- Low variance (stable)
- Risk: underfitting

Key problem: Single trees are **notoriously unstable**

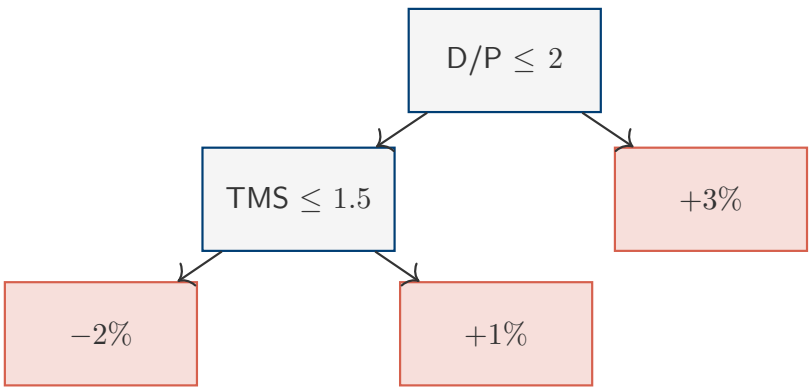
- Small change in data $\rightarrow$ completely different tree structure
- Different split at root cascades to entire tree

Can we reduce variance while keeping flexibility?

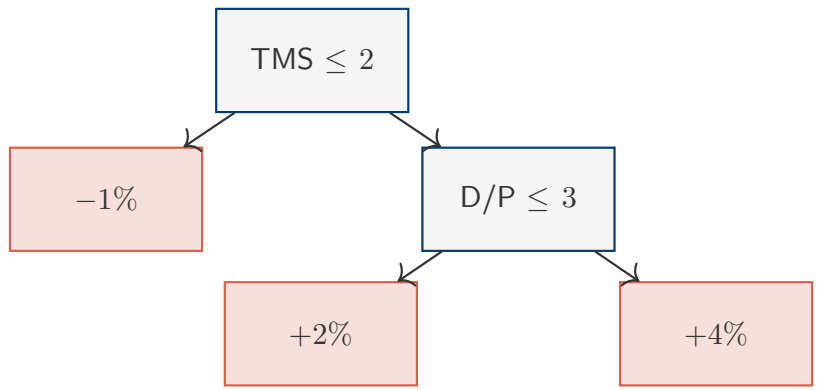
Visualizing the Instability Problem

Illustration: Same data, two different root splits

Sample A



Sample B



- High variance in predictions
  - Predictions are very sensitive to root splits
- This is the cost of flexibility

Ensemble Learning: Bagging, Random Forests, and Boosting

The Underlying Intuition

Think about macroeconomic forecasting:

- One agency forecast: 60% accuracy
- Average forecast from 10 independent agencies: 75% accuracy
- Why? Individual forecasting errors tend to cancel out

The Statistical Principle:
Given B independent observations $z_1, \ldots, z_B$, each with variance σ^2:

$$\text{Var}(\bar{z}) = \text{Var}\left(\frac{1}{B}\sum_{b=1}^{B} z_b\right) = \frac{\sigma^2}{B}$$

Averaging reduces variance by a factor of B!

The Challenge: We Don't Have Multiple Datasets

The Problem:

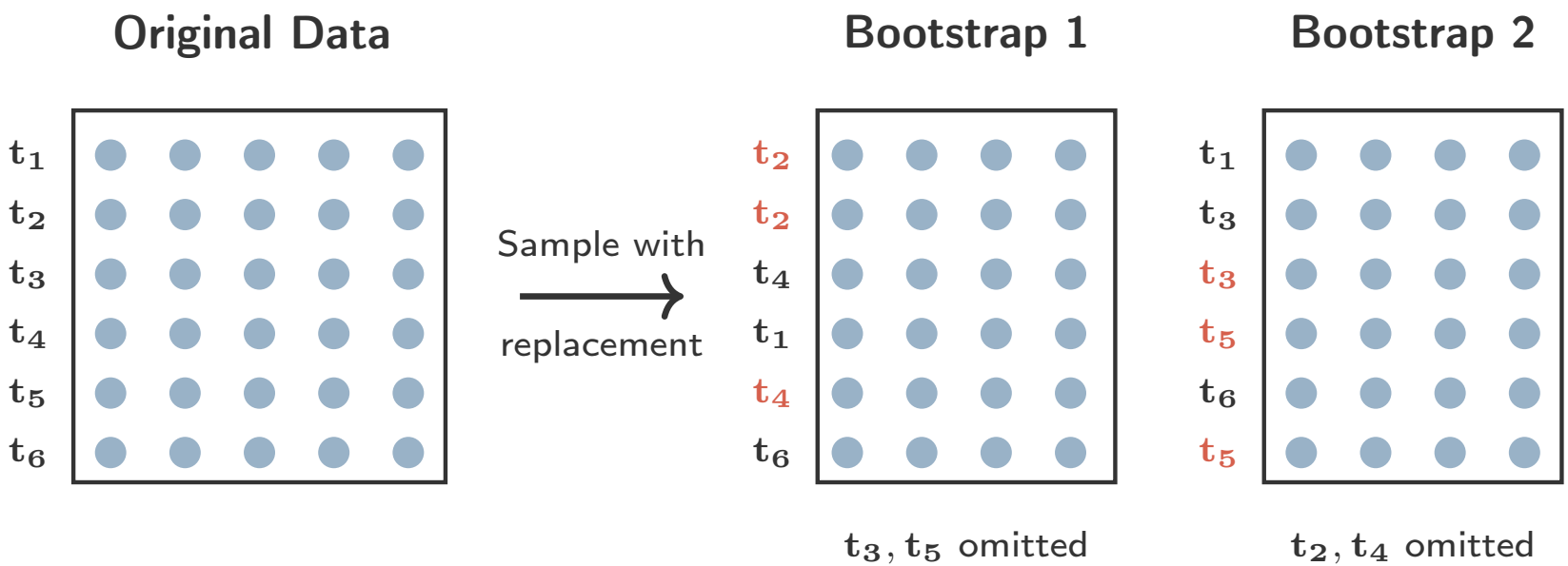
- Ideally, we'd train trees on B different training datasets
- Then average their predictions: $\widehat{f}_{\text{bag}}(x) = \frac{1}{B}\sum_{b=1}^{B}\widehat{f}^{b}(x)$
- But we only have one dataset!

The Solution: Bootstrap Sampling

- Take repeated samples **with replacement** from our training dataset
- Each bootstrap sample has the same size as the original data
- This creates the variation we need for ensemble diversity

What Does Bootstrap Sampling Do?

Bootstrap: Sample n rows (time periods) *with replacement* from original data



What happens:

- Some periods appear multiple times
- Some periods are omitted entirely
- All p predictors remain

Why it helps:

- Each tree sees different data
- Trees make different errors
- Averaging reduces variance

Why Does Averaging Reduce Variance?

Setup: B estimators, each with variance σ^2 and pairwise correlation ρ

Variance of average:

$$\text{Var}\left(\widehat{f}_{\text{bag}}(x)\right) = \text{Var}\left(\frac{1}{B}\sum_{b=1}^{B}\widehat{f}^{b}(x)\right) = \rho\sigma^2 + \frac{1-\rho}{B}\sigma^2$$

As $B \to \infty$ Variance $\to \rho\sigma^2$

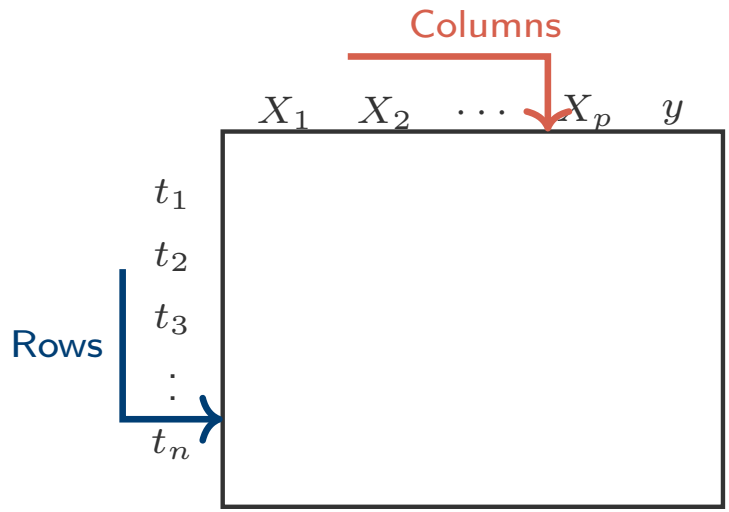
Implications:

- First term depends on **correlation between trees**
- If $\rho = 0$: variance $\to 0$ (ideal)
- If $\rho = 1$: no benefit from averaging
- **Issue:** Bootstrap trees are correlated (same predictors across trees)

The Challenge: How do we ensure our bagged trees are sufficiently different from each other?

Two Sources of Randomness in Random Forests

Random Forests combine **two** randomization strategies:



1. Bootstrap rows (Bagging)

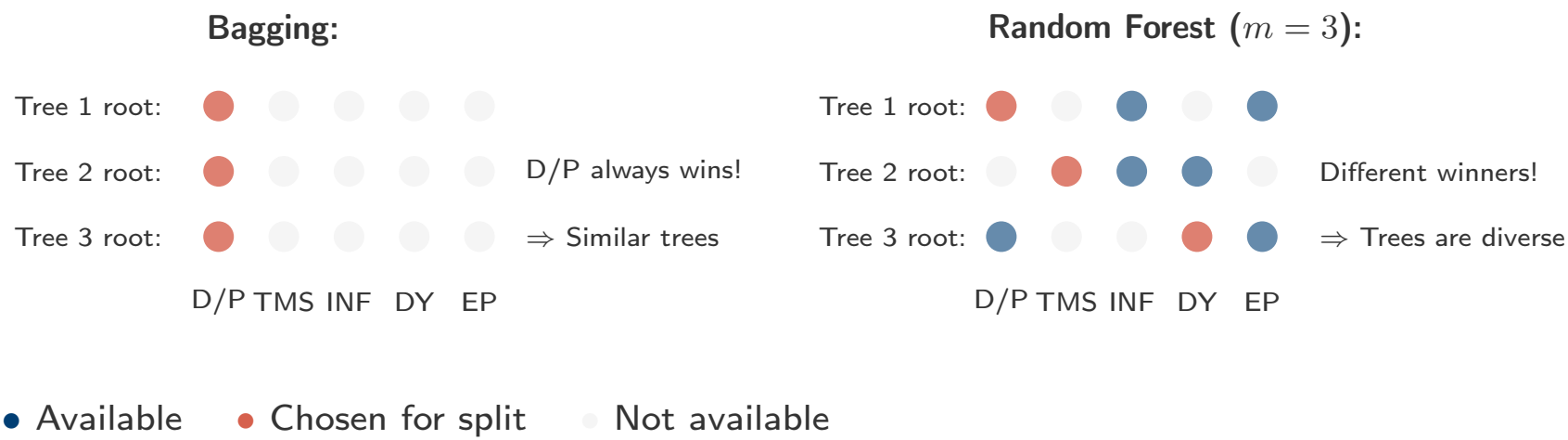
- Sample n time periods with replacement
- Some periods repeated, others omitted
- $\approx 37\%$ left out ("out-of-bag")
- *Goal:* Create diverse training sets

2. Subsample columns (Random Forest)

- At each split, consider only $m < p$ predictors
- Different m predictors each tree
- All rows in node still used
- *Purpose:* Decorrelate trees

Visualization: The Random Forest Innovation

Problem: Even with bootstrap sampling, trees are correlated

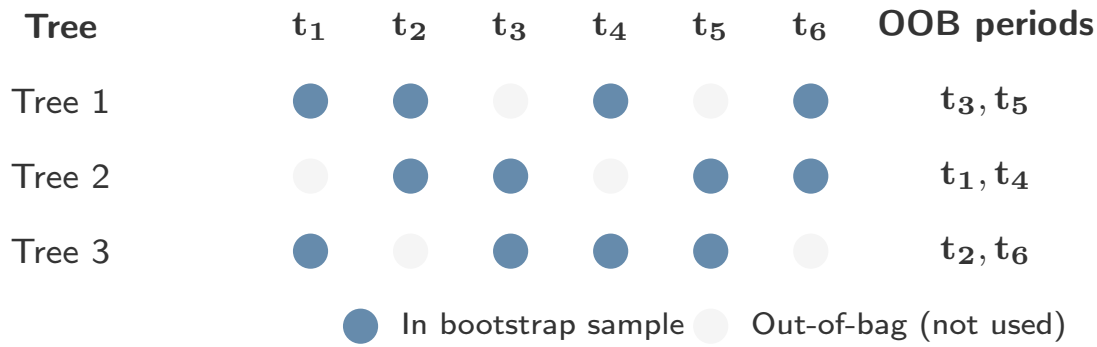


Result: Lower correlation ρ between trees $\Rightarrow$ lower $\mathrm{Var}\left(\widehat{f}_{\text{bag}}(x)\right)$

Typical: $m \approx \sqrt{p}$

Out-of-Bag (OOB) Error: From Bootstrap Sampling

Recall: We bootstrap time periods (rows) with replacement



OOB prediction for observation i:

$$\widehat{y}_i^{\text{OOB}} = \frac{1}{|B_i|} \sum_{b \in B_i} T_b(x_i)$$

where $B_i = \{b : \text{observation } i \text{ was OOB for tree } b\}$.

Example: Only Tree 2 and 3 see observation $\mathbf{t_3} \Rightarrow$ OOB prediction using Tree 1 (which did not use $\mathbf{t_3}$).

Free Cross-Validation!

OOB error $\approx$ test error, no need for separate validation set

Summary: The Key Innovation of Random Forest

Problem: Bagged trees are correlated (few strong predictors can dominate the splits)

Solution: **Decorrelate** the trees (Breiman, 2001)

Random Forest modification:
At each split, consider only a **random subset** of m predictors

- Before: choose best split among all p predictors
- After: choose best split among random $m < p$ predictors
- Different random subset at each split

Recall: $\text{Var}\left(\widehat{f}_{\text{bag}}(x)\right) = \rho\sigma^2 + \frac{1-\rho}{B}\sigma^2$

- Reducing ρ directly reduces variance
- May slightly increase individual tree variance σ^2 (weaker splits)
- Net effect: substantial variance reduction

Two Ensemble Strategies

Bagging / Random Forest

- Build deep trees **in parallel**
- One tree does not depend on another
- Combine by averaging
- **Goal:** Variance reduction

Boosting

- Build shallow trees **sequentially**
- Each tree depends on previous
- Combine by weighted sum
- **Goal:** Bias reduction

Key Insight

Bagging: "Average many overfitting trees to reduce variance"
Boosting: "Sequentially correct errors of underfitting trees to reduce bias"

Boosting: The Core Idea

Intuition: Fit trees to **residuals** from previous trees

1. Fit a simple tree $\widehat{f}_1(x)$ to data
2. Compute residuals: $r_i^{(1)} = y_i - \widehat{f}_1(x_i)$
3. Fit next tree $\widehat{f}_2(x)$ to residuals $r^{(1)}$
4. Update: $\widehat{f}(x) = \widehat{f}_1(x) + \widehat{f}_2(x)$
5. Repeat: fit trees to remaining residuals

Final prediction:

$$\widehat{f}(x) = \sum_{b=1}^{B} \widehat{f}_b(x)$$

Each tree "boosts" the model by correcting mistakes of all previous trees

Gradient Boosting: Formal Framework

View as gradient descent in function space (Friedman, 2001)

Objective: Minimize loss $L(y, f(x))$, e.g., squared error

$$\min_f \sum_{i=1}^{n} L(y_i, f(x_i)) = \min_f \sum_{i=1}^{n} (y_i - f(x_i))^2$$

For squared error, the negative gradient is just the **residual**:
$r_i = y_i - f(x_i)$

Concrete example: Each tree tries to predict the previous residual

Iter	Tree target	Tree prediction	Cumulative $\widehat{f}(x_i)$	New residual
0	—	—	2% (mean)	$5\% - 2\% = 3\%$
1	3%	$h_1(x_i) = 1.5\%$	$2\% + 1.5\% = 3.5\%$	$5\% - 3.5\% = 1.5\%$
2	1.5%	$h_2(x_i) = 0.7\%$	$3.5\% + 0.7\% = 4.2\%$	$5\% - 4.2\% = 0.8\%$
3	0.8%	$h_3(x_i) = 0.4\%$	$4.2\% + 0.4\% = 4.6\%$	$5\% - 4.6\% = 0.4\%$

Each tree corrects remaining errors $\rightarrow$ residuals shrink toward zero

The Gradient Boosting Algorithm

Algorithm 1 Gradient Boosting for Regression

1: Initialize $\widehat{f}^{(0)}(x) = \bar{y}$ (mean of training targets)
2: **for** $b = 1$ to B **do**
3: Compute residuals: $r_i = y_i - \widehat{f}^{(b-1)}(x_i)$ for all i
4: Fit a shallow tree $h_b(x)$ to residuals $\{(x_i, r_i)\}_{i=1}^n$
5: Update: $\widehat{f}^{(b)}(x) = \widehat{f}^{(b-1)}(x) + \nu \cdot h_b(x)$
6: **end for**
7: **Output:** $\widehat{f}(x) = \widehat{f}^{(B)}(x) = \bar{y} + \nu \sum_{b=1}^{B} h_b(x)$

Key hyperparameters:

- B: number of trees (iterations)
- $\nu \in (0, 1]$: learning rate (shrinkage) — typically 0.01–0.1
- Tree depth: typically 3–8 (shallow!)

Why Shrink? ($\nu < 1$)

Small steps prevent overshooting—like turning down the volume on each correction so we do not over-correct and start fitting noise

Why Shallow Trees in Boosting?

Recall bias-variance:

- Random Forest: deep trees (low bias) + averaging (reduces variance)
- Boosting: shallow trees (high bias) + sequential correction (reduces bias)

Shallow trees ("stumps" to depth 4–6):

- High bias individually
- But sequential fitting systematically reduces bias
- Low variance $\rightarrow$ stable at each step

Tree depth controls interaction order:

- Depth 1 (stump): only main effects
  - Example: "If D/P > 2, predict $+2\%$" — no interactions
- Depth 2: pairwise interactions
  - Example: "If D/P > 2 **and** TMS > 1, predict $+4\%$"
- Depth d: up to d-way interactions

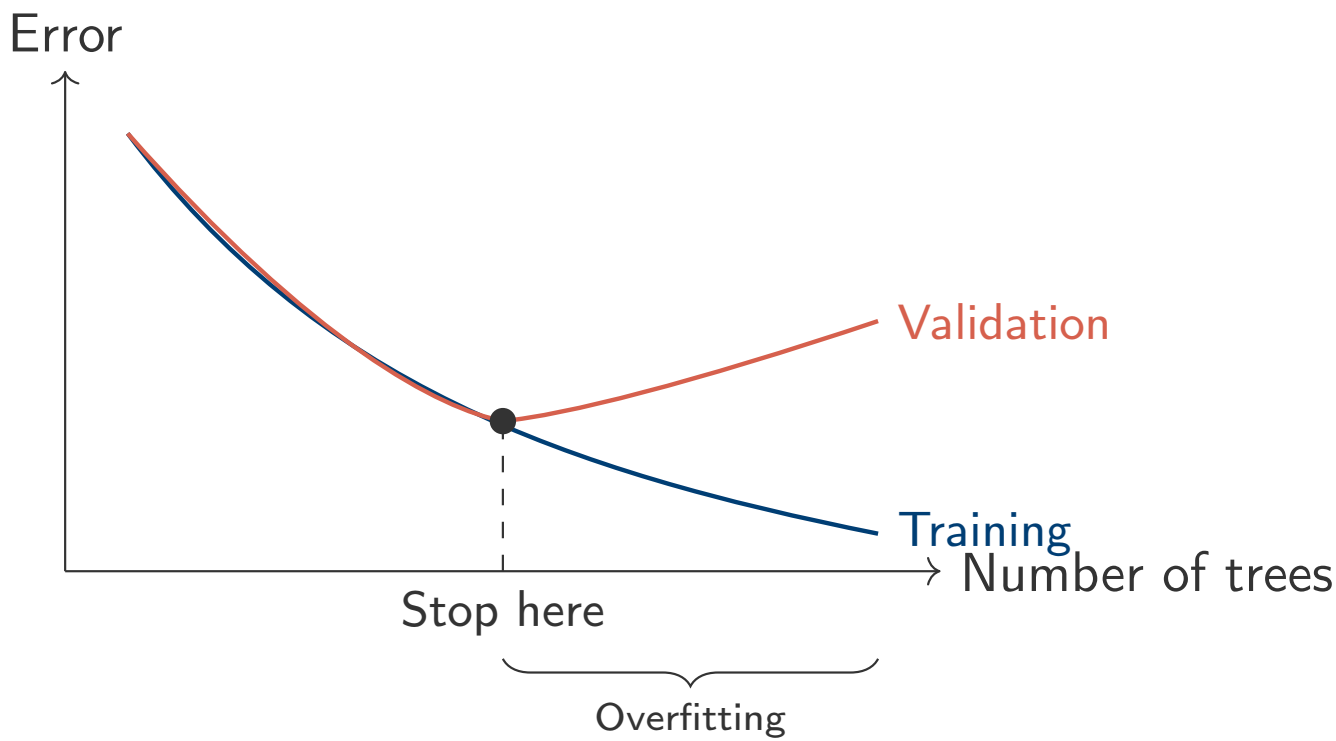
Early Stopping: Preventing Overfitting

Key difference from Random Forest:

- RF: more trees $\rightarrow$ always helps (or neutral)
- Boosting: more trees $\rightarrow$ **can eventually overfit**

Why? Each boosting tree is designed to fix remaining errors—eventually it starts fitting noise rather than signal

Solution: Monitor validation error, stop when it increases



RF vs. Gradient Boosting: When to Use Which?

	Random Forest	**Gradient Boosting**
Tree construction	Parallel (independent)	Sequential (dependent)
Individual trees	Deep (overfit)	Shallow (underfit)
Ensemble goal	Variance reduction	Bias reduction
Overfitting risk	Low	Higher (use early stopping)
Tuning difficulty	Easier	Harder

Practical guidance:

Limited tuning time:	Start with RF (robust to default parameters)
Maximizing performance:	Try Boosted trees with careful tuning
Very noisy data:	RF (harder to overfit)
Clear signal, many features:	Gradient boosting often wins

Hint: Try both, compare out-of-sample performance

Revisiting Market Timing

Recap: What We Learned in Lecture 2

Market timing with linear methods:

- OLS overfits badly with many predictors (negative R^2_{OS})
- Ridge provided best statistical accuracy (R^2_{OS} closest to zero)
- Lasso/Elastic Net behaved like historical mean (always bullish)
- Economic performance $\neq$ statistical accuracy

Key Question for Today

Can tree-based methods capture **non-linearities and interactions** that linear models miss?

Motivation: Maybe D/P predicts differently at extremes, or the effect of TMS depends on inflation regime

The Promise of Tree-Based Methods

Potential advantages over linear models:

1. **Automatic non-linearity:** No need to specify polynomial terms
2. **Automatic interactions:** Discovers D/P $\times$ TMS effects without manual coding
3. **No scaling required:** Trees are scale-invariant

Potential disadvantages:

- Higher variance (especially single trees)
- Hard to extrapolate beyond training data
- More hyperparameters to tune
- Not particularly effective with noisy data

Empirical question: Do these advantages outweigh disadvantages for market timing?

Implementation: Methods Compared

Linear methods (from Lecture 2):

- OLS, Ridge, Lasso, Elastic Net

Tree-based methods (new):

- **Decision Tree:** Single pruned tree
- **Random Forest:** Bagging + feature subsampling
- **Gradient Boosting:** Sequential residual fitting

Common setup:

- Walk-forward prediction (expanding window)
- Initial training: 120 months (10 years)
- Hyperparameters: Time-series cross-validation

Data: GWZ (2024) — 25 predictors, monthly excess returns 1953–2021

Implementing Regression Trees

Single Tree Hyperparameters

Key hyperparameters for regularization:

Parameter	Effect	Grid
`max_depth`	Limits tree depth	{2, 3, 4, 5, 6}
`min_samples_leaf`	Min. observations per leaf	{5, 10, 20}

Why constrain depth?

- The deeper the tree, the more the model overfits
- Shallow trees = simpler, more stable rules
- Market timing: probably only need 2–3 key fundamental interactions

Selection: Time-series cross-validation (no shuffling!)

Random Forest Hyperparameters

Key hyperparameters:

Parameter	Effect	Grid
`n_estimators`	Number of trees	100 (fixed)
`max_depth`	Individual tree depth	{3, 5, 7, None}
`max_features`	Features per split	$\{\sqrt{p}, 0.33, 0.5\}$
`min_samples_leaf`	Min. observations per leaf	{5, 10, 20}

Key insight: More trees never hurts (just slower)

Focus tuning on:

- `max_features`: controls tree correlation (lower $\rightarrow$ more diverse)
- `max_depth`: controls individual tree complexity (deeper tree $\rightarrow$ overfitting)

Gradient Boosting Hyperparameters

Key hyperparameters:

Parameter	Effect	Grid
`n_estimators`	Number of boosting rounds	{50, 100, 200}
`max_depth`	Tree depth (interaction order)	{2, 3, 4}
`learning_rate`	Shrinkage factor ν	{0.01, 0.05, 0.1}

Critical trade-off: `learning_rate` $\times$ `n_estimators`

- Small ν + many trees = careful, gradual learning
- Large ν + few trees = aggressive, faster learning
- Small ν generally better but slower

Depth for GBM: Keep shallow individual trees (2–4) since boosting reduces bias

Out-of-Sample Statistical Performance

Out-of-Sample R^2: Linear vs. Tree Methods

Recall: $R^2_{OS} = 1 - \frac{\sum_t (r_t - \widehat{r}_t)^2}{\sum_t (r_t - \bar{r}_t)^2}$

Negative R^2_{OS} means worse than predicting the historical mean.

Model	R^2_{OS} **(%)**	**Assessment**
OLS	-13.85	**Severe overfitting**
Ridge	-0.50	Slight underperformance
Lasso	$+0.10$	$\approx$ Historical mean
Elastic Net	$+0.14$	$\approx$ Historical mean
Decision Tree	-10.68	**Severe overfitting**
Random Forest	$\mathbf{+0.15}$	**Best**
Gradient Boosting	-0.88	Moderate overfitting

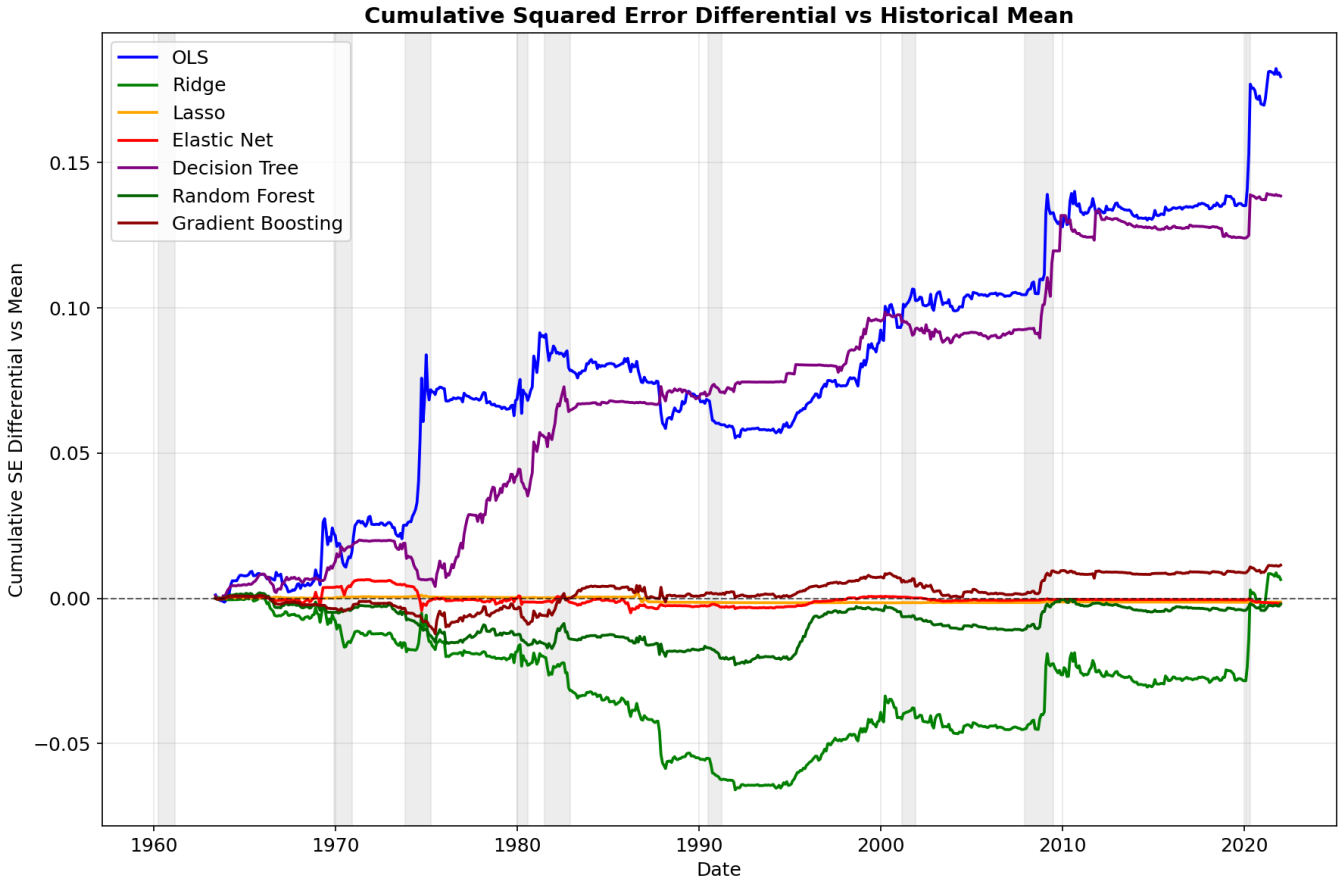
Surprise: Random Forest achieves the **only positive** R^2_{OS} among complex models!

Interpreting the Results

Key observations:

1. **Single trees overfit dramatically** ($R^2_{OS} = -10.7\%$)
   - Worse than OLS in relative terms
   - Flexibility without ensemble averaging $\rightarrow$ noise fitting
2. **Random Forest achieves best statistical accuracy**
   - Only non-linear model with positive R^2_{OS} (+0.15%)
   - Averaging + decorrelation effectively controls variance
3. **Gradient Boosting disappoints** ($R^2_{OS} = -0.88\%$)
   - Sequential bias reduction less useful when signal is weak
   - Easier to overfit than Random Forest
4. **Lasso and E-Net behave like historical mean**
   - Shrink most coefficients to zero $\rightarrow$ predict near-constant

Cumulative Squared Error Differential



The chart shows the **cumulative Sum of Squared Error (SSE)** difference: $\sum_{t=1}^{T}[(y_t - \widehat{y}_t^{\text{model}})^2 - (y_t - \bar{y}_t)^2]$

- **Below zero** = model beats historical mean
- **Above zero** = model worse than historical mean

Cumulative SSE Differential: What We Learn

Key patterns in the figure:

1. **OLS & Decision Tree**: Steadily accumulate errors, especially during crises
2. **Ridge & Random Forest**: Fluctuate but consistently below zero
3. **Lasso/Elastic Net/GBM**: Flat near zero (predictions $\approx$ historical mean)

Timing insight: Large jumps occur during volatile periods (gray bands = recessions)

- 2008–2009 crisis: OLS predictions particularly bad

Sign Accuracy: Getting the Direction Right

Sign accuracy: % of months where predicted sign = actual sign

Model	Sign Acc. (%)	When Up (%)	When Down (%)
Historical Mean	59.4	100.0	0.0
OLS	58.4	68.9	43.0
Ridge	58.1	73.2	36.0
Lasso	**59.5**	100.0	0.3
Elastic Net	57.8	92.3	7.3
Decision Tree	56.5	74.2	30.8
Random Forest	58.0	82.8	21.7
Gradient Boosting	56.3	86.1	12.6

Key insight: Lasso achieves highest sign accuracy but is **always bullish**—same strategy as historical mean.

Why Trees Have More Balanced Predictions

Sparse linear models (Lasso/Elastic Net):

- Shrink most/all coefficients toward zero
- Predictions collapse to near-constant (historical mean)
- "When Down" accuracy ≈ 0–7% (almost never predict negative)

Tree models:

- "When Down" accuracy 13–31% (do predict some negative months)

OLS/Ridge (unconstrained):

- Most balanced: "When Down" accuracy 36–43%
- But also most volatile predictions $\rightarrow$ more trading

Trade-off

More balanced predictions $\rightarrow$ potentially catch downturns
But also $\rightarrow$ higher turnover and more false alarms

Economic Performance

From Predictions to Portfolios

Market timing strategy: Same as Lecture 2

Mean-variance optimal weight:

$$\omega_t = \frac{1}{\gamma} \frac{\widehat{r}_{t+1}}{\widehat{\sigma}_t^2}$$

- $\gamma = 5$ (risk aversion)
- $\widehat{\sigma}_t^2$ = rolling 60-month variance
- Constraints: $-1 \leq \omega_t \leq 1.5$

Portfolio return:

$$r_{t+1}^{\text{portfolio}} = \omega_t \cdot r_{t+1}^{\text{market}} + (1 - \omega_t) \cdot r_f$$

Transaction costs: 50 bps per unit weight change

Performance Comparison: Sharpe Ratios

Strategy	Ann. Ret. (%)	Vol. (%)	Sharpe	Sharpe Net
Buy & Hold	6.9	14.8	0.46	0.46
Historical Mean	4.3	11.1	0.39	0.38
OLS	11.2	17.5	0.64	0.45
Ridge	**11.9**	17.1	**0.69**	**0.53**
Lasso	4.6	11.6	0.40	0.39
Elastic Net	5.0	11.8	0.43	0.37
Decision Tree	6.3	16.0	0.39	0.29
Random Forest	7.0	14.4	0.49	0.35
Gradient Boosting	5.1	14.4	0.36	0.22

Key finding: Ridge dominates economically despite worse R^2_{OS} than RF!

Puzzle: How can Ridge have negative R^2_{OS} but best returns?

Impact of Transaction Costs

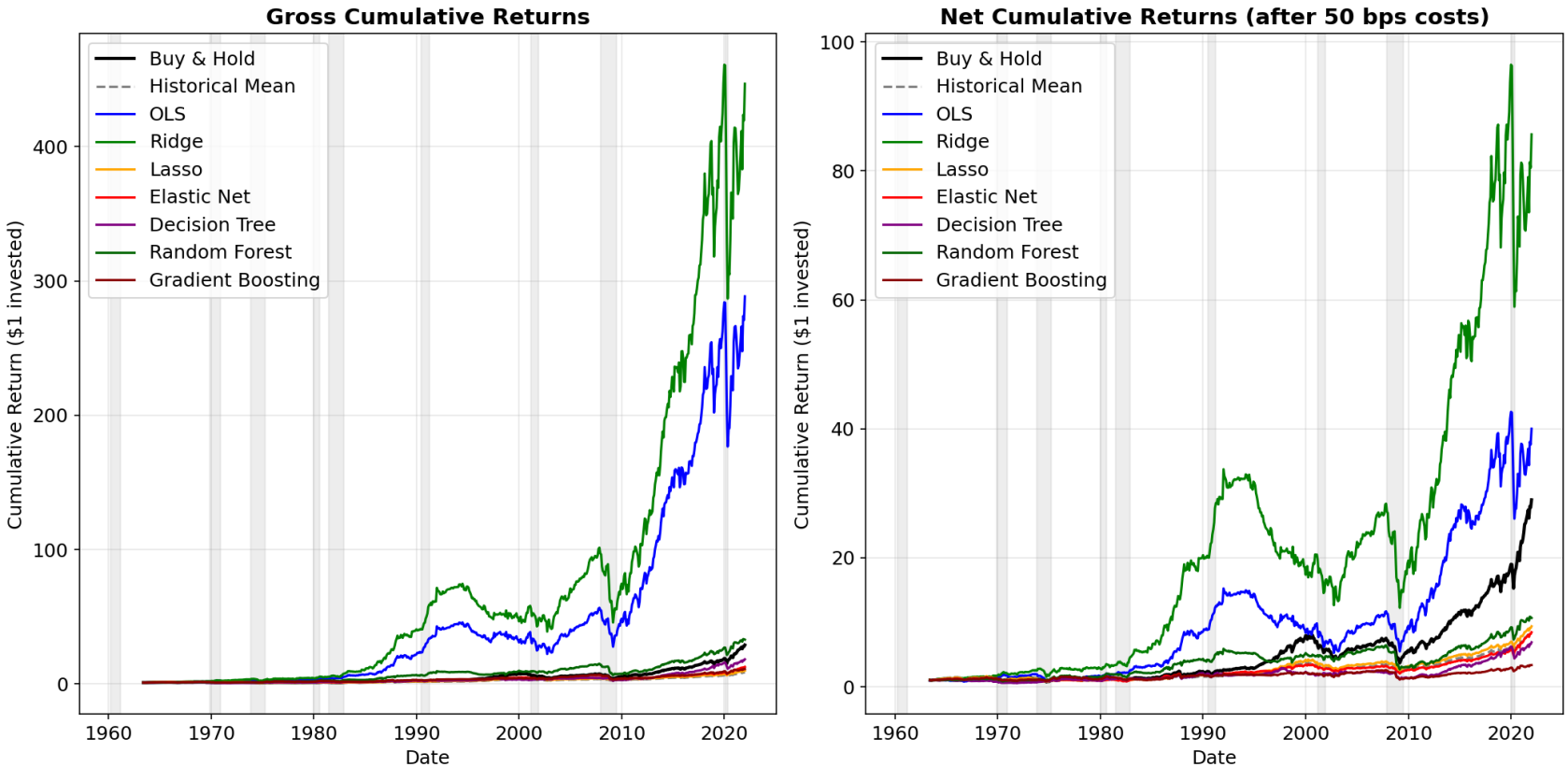
Turnover comparison:

Model	Avg. Annual Turnover	Sharpe Gross	Sharpe Net
Historical Mean	0.13×	0.39	0.38
Lasso	0.19×	0.40	0.39
Elastic Net	1.40×	0.43	0.37
Decision Tree	3.32×	0.39	0.29
Random Forest	3.85×	0.49	0.35
Gradient Boosting	4.08×	0.36	**0.22**
Ridge	5.67×	0.69	**0.53**
OLS	6.79×	0.64	0.45

Observations:

- Tree methods have 3–4× annual turnover
- Ridge has highest turnover but still best net Sharpe!
- GBM hurt most by costs (Sharpe drops from 0.36 to 0.22)

Cumulative Returns



Left panel: Gross returns (no transaction costs)
Right panel: Net returns (50 bps costs)

Cumulative Returns: What the Figure Shows

Gross returns (left panel):

- **Ridge**: Dominates dramatically, reaches \$400+ by 2021
- **OLS**: Second best gross (∼\$290), but very volatile path
- **Random Forest**: Solid growth (∼\$32), much smoother than OLS
- **All others**: Cluster near bottom (\$5–\$20)

Net returns after 50 bps costs (right panel):

- **Ridge**: Still best (∼\$85), but gap narrows substantially
- **Buy & Hold**: Catches up to ∼\$40 (no trading costs)
- **OLS**: Falls to ∼\$40 (high turnover penalized)
- **Random Forest**: ∼\$10, penalized by 3.85× turnover

Key Insight

Ridge's *economic* dominance persists after costs, but the margin shrinks. Transaction costs matter more for tree methods than linear methods.

The R^2 vs. Economic Performance Puzzle

Puzzle: Random Forest has best R^2_{OS} (+0.15%), but Ridge has best returns. Why?

Resolution: R^2_{OS} measures *average* squared error, but economic value depends on:

1. **Magnitude of correct predictions**
   - Ridge makes bigger bets when confident
   - RF predictions are more muted (closer to zero)
2. **Timing of errors**
   - Being wrong in calm markets costs less than in volatile markets
   - Ridge may time the big moves better
3. **Portfolio construction**
   - Mean-variance weights amplify prediction differences
   - Small differences in predictions $\rightarrow$ big differences in positions

Lesson

Statistical accuracy $\neq$ **Economic value**. Always evaluate both!

Why Don't Trees Dominate?

We expected trees to help because:

- They capture non-linearities automatically
- They discover interactions without specification

Results show: RF has best R^2_{OS}, but Ridge has best economic performance. Why?

1. **RF predictions are too conservative**
   - Averaging many trees $\rightarrow$ predictions shrink toward zero
   - Good for R^2 (small errors), bad for returns (small bets)
2. **Ridge takes bigger positions**
   - More aggressive forecasts when predictors are extreme
   - Higher volatility but also higher returns
3. **Variance reduction may be too much**
   - RF succeeds at variance reduction
   - But in low signal-to-noise settings, you need some variance to generate returns

The Market Timing Puzzle

We expected trees to help because:

- They capture non-linearities automatically
- They discover interactions without specification

But results show trees don't clearly outperform Ridge. Why?

1. **Signal is too weak**
   - Monthly return volatility ≈ 4–5%
   - Predictable component $\approx 0.5\%$ (if any)
   - Non-linearities in the noise, not the signal
2. **Sample size limitations**
   - $\sim$800 months may not be enough for complex patterns
   - Trees need more data than linear models
3. **Linear approximation may be sufficient**
   - True non-linearities might be small
   - Occam's razor: simpler models generalize better

When Do Trees Excel?

Trees tend to outperform when:

1. **True interactions exist and are strong**
   - Credit risk: income $\times$ debt ratio
   - Trading: volume $\times$ price momentum
2. **Non-linearities are pronounced**
   - Threshold effects (default if ratio $> x$)
   - Saturation effects
3. **Sample size is large**
   - Cross-sectional prediction: 1000s of stocks per month
   - More data to detect complex patterns
4. **Signal-to-noise ratio is higher**
   - Default prediction: $R^2 \approx 10$–30%
   - Return prediction: $R^2 \approx 1$–2%

Gu, Kelly & Xiu (2020): What the Literature Says

Cross-sectional return prediction:

- Trees perform well with firm characteristics
- Neural networks slightly better
- All ML methods beat linear models

Time-series (market timing):

- Much harder problem
- Trees "competitive but not dominant"
- Gains modest compared to cross-section

Key Insight

The value of ML depends on the application:

- High signal + large cross-section $\rightarrow$ ML helps a lot
- Low signal + small time series $\rightarrow$ simpler methods suffice

Summary and Next Steps

Key Takeaways from Today

1. Market timing is very difficult even with non-linear regression models
   - Random Forest only modest positive R^2_{OS} (+0.15%)
   - Averaging + decorrelation helps to control overfitting
   - Yet ridge achieves a statistical accuracy (R^2_{OS}) just as good as RF
2. Single trees and OLS catastrophically overfit
   - Decision Tree: $R^2_{OS} = -10.7\%$, OLS: $R^2_{OS} = -13.8\%$
   - Flexibility without regularization $\rightarrow$ noise fitting
3. The method should match the problem
   - Market timing: low signal, small sample $\rightarrow$ simple methods OK
   - Cross-section/credit risk: higher signal, large sample $\rightarrow$ trees help (see next lectures)

Readings and Next Steps

Readings for Today's Lecture:

- James et al. (2023), Chapter 8 *[Recomm.]*
- Goyal, Welch, & Zafirov (2024). "A comprehensive 2022 look at the empirical performance of equity premium prediction." *The Review of Financial Studies*. *[Supp.]*
- Gu, Kelly & Xiu (2020), Sections on tree methods *[Supp.]*

Coming up in Week 4:

- Classification methods for credit risk
  - Logistic regressions
  - Classification trees and ensemble learning
  - Credit risk assessment (LendingClub data), evaluation metrics, calibration.
- Why conventional regressions cannot be used for classification?
- Would non-linearity help in a classification context?

Questions?