Lesson 22: Regularization

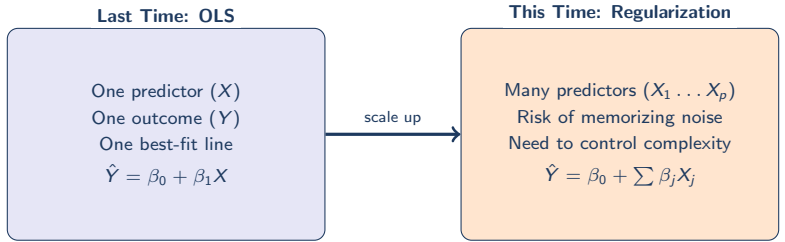
Data Science with Python – BSc Course

Data Science Program

BSc Course

45 Minutes

Previously on L21...



OLS finds the best-fit line. But what happens when we have 50 features instead of 1?

β = market sensitivity. More features $\rightarrow$ more risk of overfitting.

Learning Objectives

The Problem: With many predictors, our model can memorize noise instead of learning patterns.

After this lesson, you will be able to:

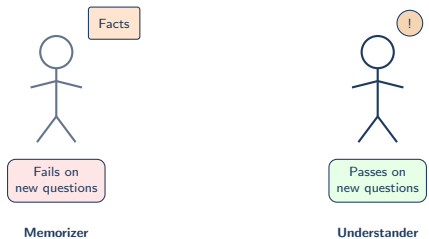
- Recognize overfitting and the bias-variance tradeoff
- Apply Ridge (L2) regularization to shrink coefficients
- Apply Lasso (L1) for automatic feature selection
- Tune the regularization strength λ with cross-validation

Finance Application: Building robust factor models with many correlated predictors

The Goldilocks Problem

Your model memorized the noise.

Every dataset contains **signal** (true patterns) and **noise** (random variation). A model that is too complex fits both—perfect on training data, terrible on new data.

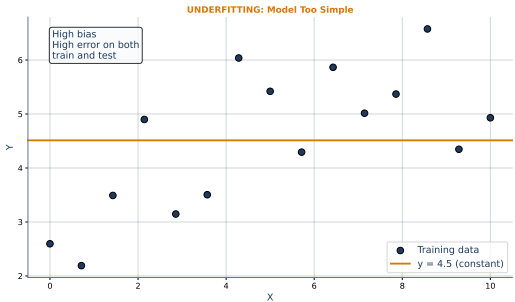


Analogy: studying by memorizing answers vs. understanding concepts

Too Simple: Underfitting

The model is too simple to capture the true pattern.

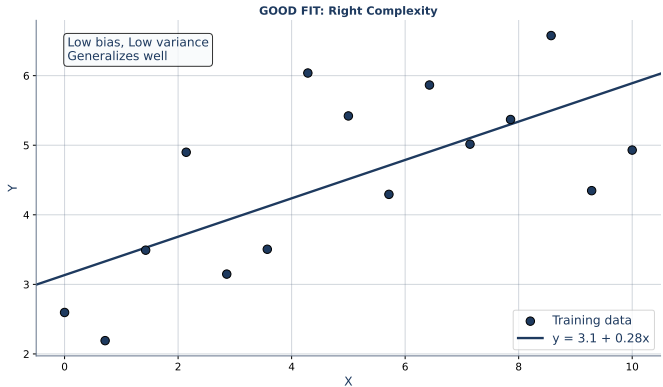
- Straight line through curved data
- High bias – consistently wrong in the same direction
- High error on both training and test data



Underfitting = high bias. Adding more data won't help – the model itself is too rigid.

Just Right: Good Fit

Captures the pattern, ignores the noise.

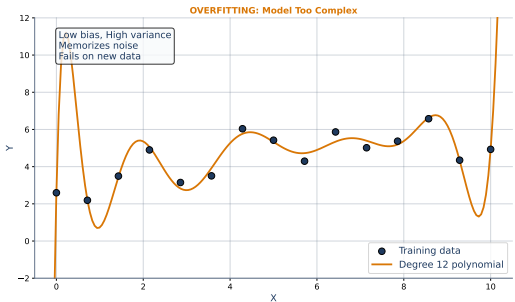


The sweet spot: complex enough to learn, simple enough to generalize

Too Complex: Overfitting

Perfect on training data, terrible on new data.

- Wiggly line chases every data point
- High variance – model changes drastically with different samples

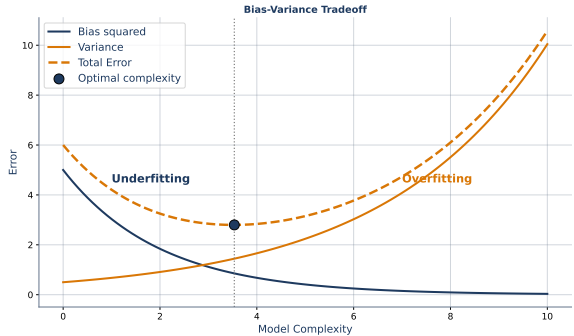


Overfitting = high variance. The model memorized noise that won't repeat.

The Bias-Variance Tradeoff

$$\text{Total Error} = \text{Bias}^2 + \text{Variance} + \text{Irreducible Noise}$$

Simple = high bias. Complex = high variance. Regularization controls this tradeoff.



Goal: minimize total error, not just training error

Deep Dive: Ridge vs OLS Mathematics

OLS Closed-Form:

$$\hat{\beta}_{\text{OLS}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$$

- Requires $\mathbf{X}^T\mathbf{X}$ to be invertible (fails with multicollinearity)

Ridge Closed-Form:

$$\hat{\beta}_{\text{Ridge}} = (\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^T\mathbf{y}$$

- Adding $\lambda\mathbf{I}$ to diagonal **always** makes the matrix invertible
- This is why Ridge handles multicollinearity – it stabilizes the solution

Important: Standardize First!

- The penalty $\lambda \sum \beta_j^2$ treats all coefficients equally
- Features on different scales $\rightarrow$ unfair penalization
- Always standardize features (mean=0, std=1) before regularization

---

$\lambda\mathbf{I}$ adds a "ridge" on the matrix diagonal – that's the name origin

The Solution: Penalize Complexity

What if we add a COST for large coefficients?

- Original goal: minimize prediction errors only
- New goal: minimize errors **plus** keep coefficients small

Two Penalty Types:

- **L2 (Ridge)**: Penalty $= \lambda \sum \beta_j^2$ (sum of squared coefficients)
- **L1 (Lasso)**: Penalty $= \lambda \sum |\beta_j|$ (sum of absolute values)

Lambda (λ) controls penalty strength:

- $\lambda = 0$: no penalty (plain OLS)
- λ large: strong shrinkage toward zero

Note: In sklearn, λ is called `alpha` – same concept, different name.

Larger penalty = smaller coefficients = simpler model

Ridge: The Volume Knob

Analogy: Each coefficient is a speaker volume dial. OLS cranks every dial to fit best. Ridge adds: "minimize total volume." All dials turn down, none turn off.

Ridge = OLS + L2 penalty:

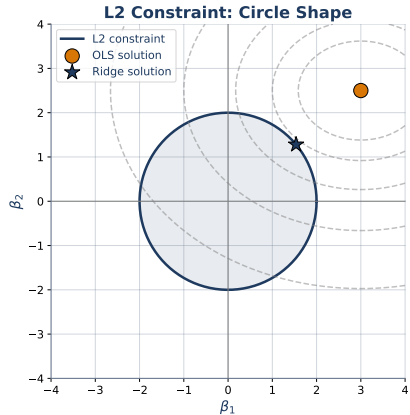
$$\text{Ridge minimizes: } \text{RSS} + \lambda \sum_{j=1}^{p} \beta_j^2$$

Name origin: $\lambda\mathbf{I}$ adds a "ridge" on the matrix diagonal. **Always standardize features first!**

Ridge shrinks ALL coefficients toward zero but never sets any to exactly zero

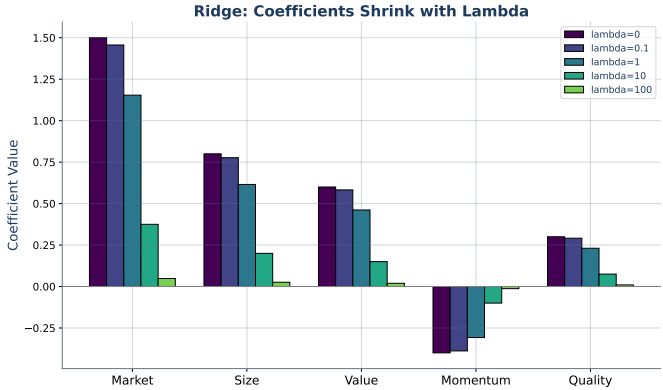
Ridge: Constraint Region

Coefficients must stay inside a **circle** (Euclidean ball).



Where the error contours first touch the circle = Ridge solution

Ridge: Lambda Effect on Coefficients



Large lambda = stronger shrinkage toward zero, simpler model

Lasso: The Suitcase Packer

Analogy: 20 items, small suitcase. Ridge: take all 20 but miniature. Lasso: leave some home, bring important ones full-size.

Lasso = OLS + L1 penalty:

$$\text{Lasso minimizes: } \text{RSS} + \lambda \sum_{j=1}^{p} |\beta_j|$$

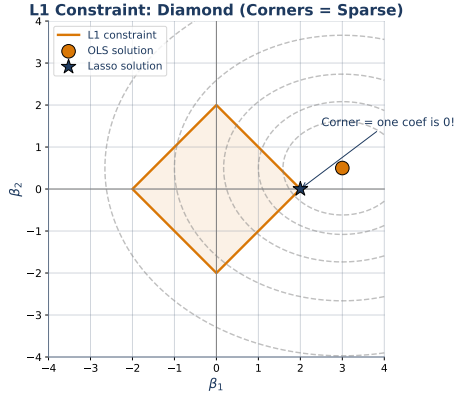
Key: Some coefficients become **exactly zero** $\rightarrow$ automatic feature selection!

LASSO = "Least Absolute Shrinkage and Selection Operator"

Lasso eliminates irrelevant features – simpler, more interpretable models

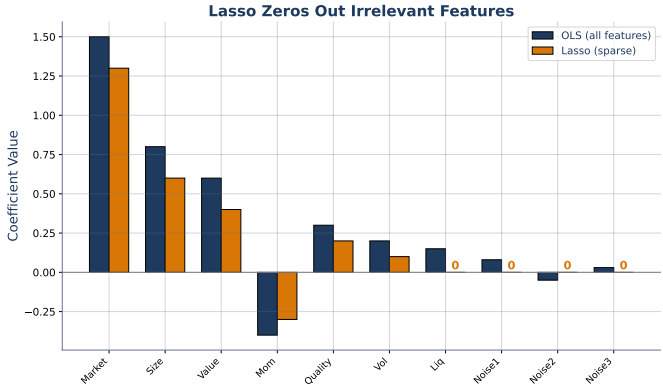
Lasso: Diamond Constraint

Diamond corners sit on axes (where $\beta_j = 0$). Contours touch corners first $\rightarrow$ coefficients become exactly zero.



L1's diamond shape is WHY Lasso does automatic feature selection

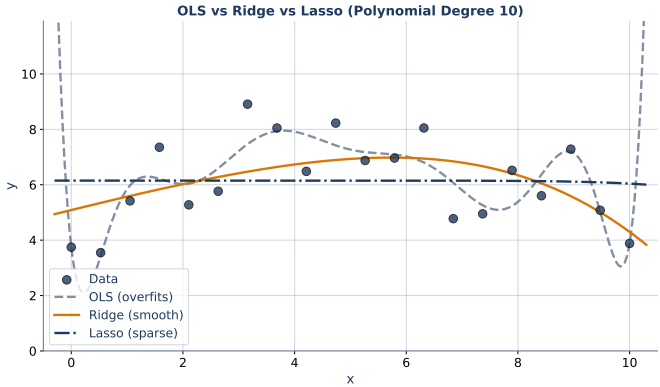
Lasso: Automatic Feature Selection



L1 penalty sets some coefficients to exactly zero – automatic feature selection

OLS vs Ridge vs Lasso: Visual Showdown

Polynomial degree 10 fitted to noisy data: OLS overfits (wiggly), Ridge smooths, Lasso zeros out weak terms.

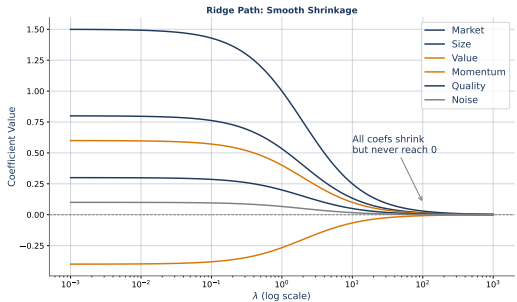


Regularization tames polynomial overfitting. Both Ridge and Lasso produce smoother fits.

Coefficient Paths: How Shrinkage Works

Ridge path: smooth shrinkage, coefficients approach but never reach zero.

Lasso path: coefficients hit exactly zero at different λ values.



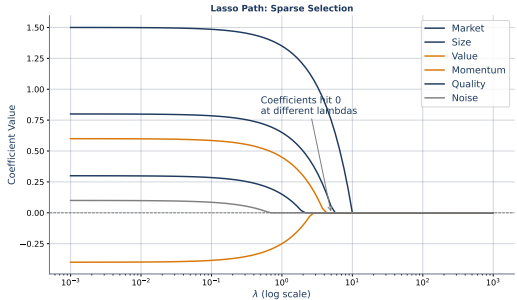
Mnemonic: Ridge keeps all features. Lasso **L**oses features (L for Lose).

Ridge: smooth decay. Lasso: sharp elimination. Both controlled by λ.

Lasso Coefficient Path

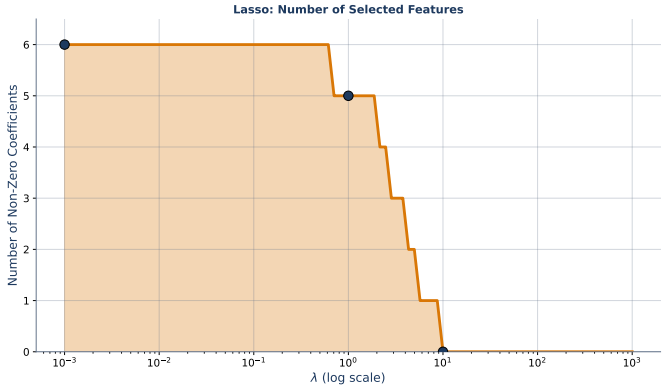
Sparse Feature Selection

- Coefficients hit exactly zero at different λ values
- Weaker predictors are eliminated first



Lasso automatically selects the most important features

Lasso Feature Count

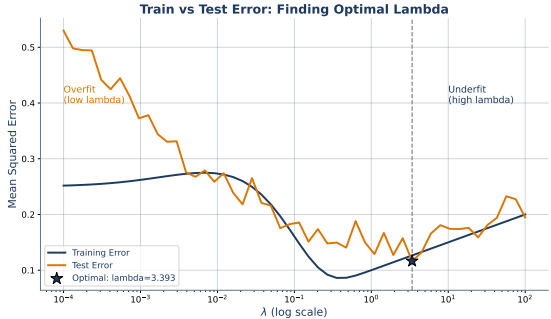


As λ increases, fewer features remain – use CV to find optimal sparsity

Tuning Lambda: How Much Penalty?

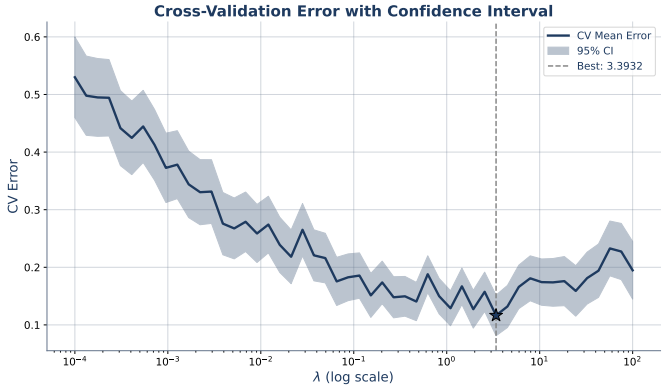
$\lambda = 0$ (no penalty, OLS) $\longrightarrow$ $\lambda \to \infty$ (all coefficients zero)

- No teacher: coefficients do whatever they want
- Moderate: discipline, reasonable coefficients
- Overbearing: coefficients do almost nothing



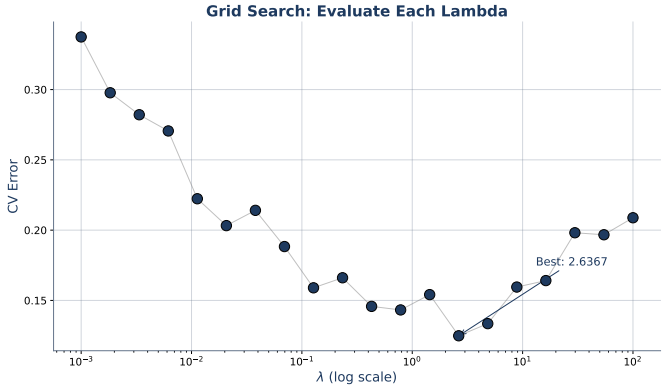
Optimal λ minimizes test error – neither underfitting nor overfitting

Tuning Lambda: Cross-Validation Confidence



CV provides error bars – pick lambda within 1 SE of minimum for simpler model

Tuning Lambda: Grid Search



Search logarithmically: alphas=[0.001, 0.01, 0.1, 1, 10, 100]

Cross-Validation: The Gold Standard

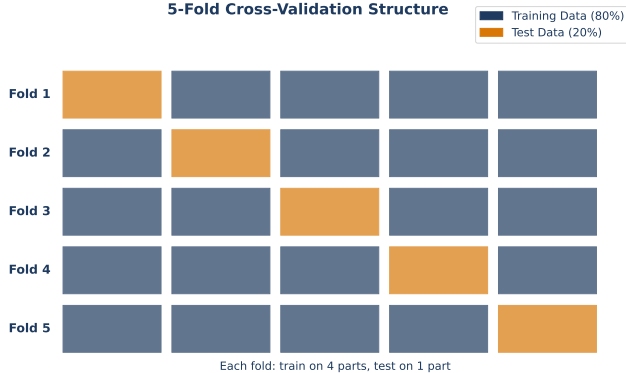
One practice test may not reflect your true ability. Five different tests and averaging gives a reliable estimate.

K-Fold CV: split into K folds, train on $K-1$, test on 1, rotate, average. **sklearn:** `RidgeCV`, `LassoCV` do this automatically.

Cross-validation: the gold standard for choosing hyperparameters like λ

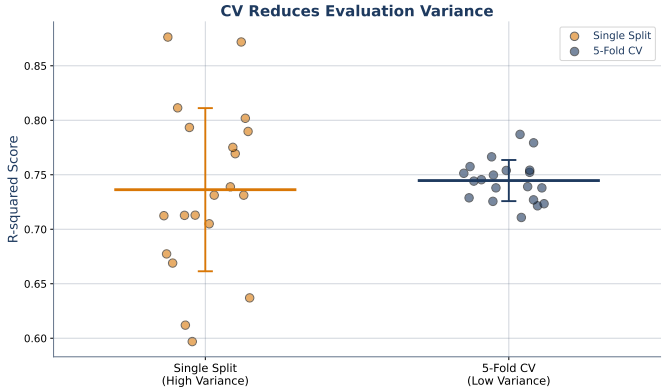
K-Fold Structure

Each observation is in the test set exactly once.



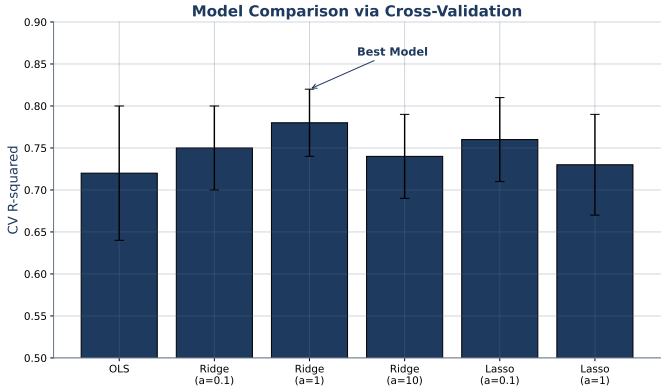
K-fold CV uses ALL data for both training and testing – no wasted observations

Cross-Validation: Reduces Variance



Averaging across folds gives more stable estimate of test performance

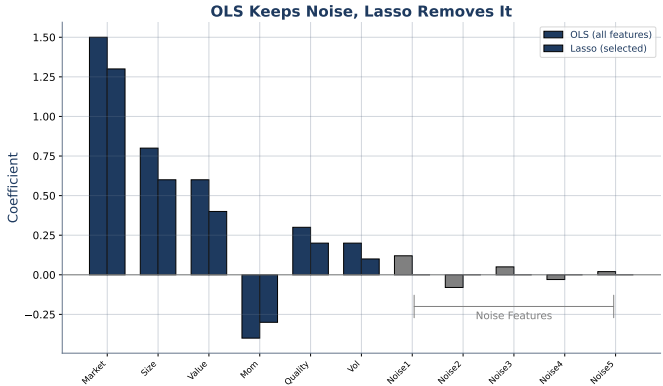
Cross-Validation: Model Comparison



CV enables fair comparison of different models and hyperparameters

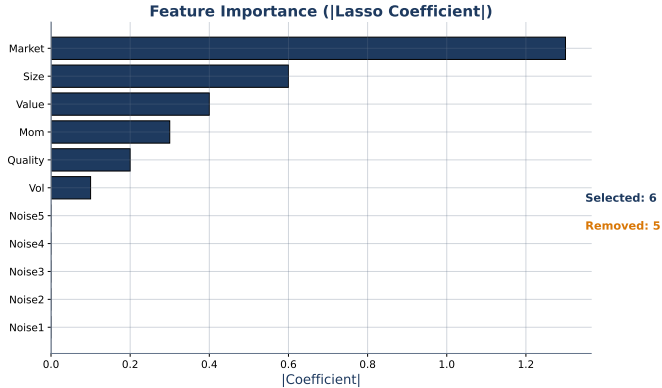
Feature Selection with Lasso

OLS keeps all features (even noisy ones). Lasso zeros out irrelevant ones.



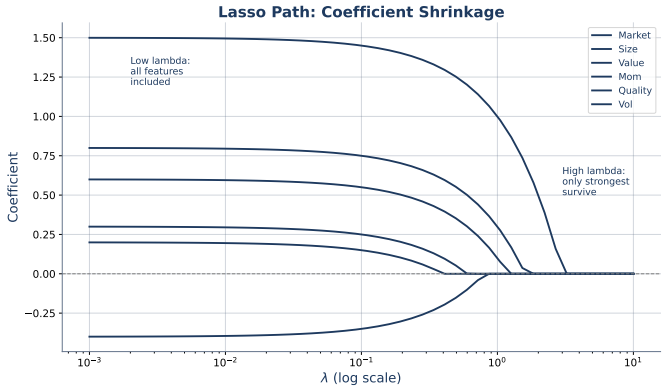
Like hiring: OLS hires all 20 candidates, Lasso keeps only the 5 productive ones

Feature Selection: Importance Ranking



Non-zero Lasso coefficients indicate selected features

Feature Selection: Lasso Path



Path plot shows order of feature elimination as lambda increases

Checkpoint: Pick the Right Tool

Which regularization method for each scenario?

1. 100 correlated features, all may matter
   → **Ridge** – keeps all features, handles multicollinearity
2. 50 features, suspect only 5 are useful
   → **Lasso** – zeros out the 45 irrelevant ones
3. Correlated features + want some selection
   → **ElasticNet** – best of both worlds

ElasticNet in Brief:

- Penalty $= \lambda_1 \sum |\beta_j| + \lambda_2 \sum \beta_j^2$
- Controlled by `l1_ratio`: 0 = pure Ridge, 1 = pure Lasso
- `ElasticNet(alpha=0.1, l1_ratio=0.5)`

When in doubt, ElasticNet with CV can find the optimal L1/L2 mix

Common Mistakes in Regularization

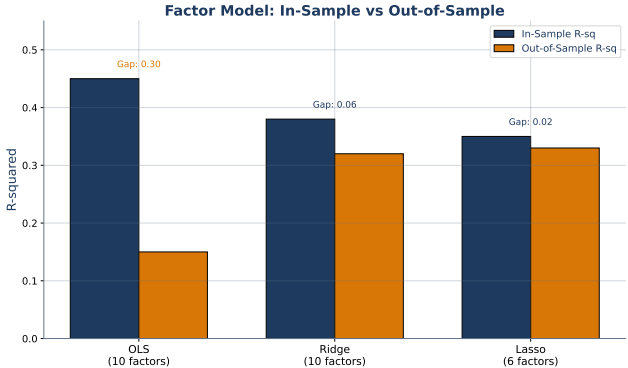
- ✗ Forgetting to standardize features before regularization
- ✗ Using R^2 on training data to pick λ instead of CV
- ✗ Not doing cross-validation at all – just guessing λ
- ✗ Comparing models trained on different feature scales

Fix: Always standardize first, always use cross-validation, always compare fairly.

These mistakes are the most common source of poor regularization results

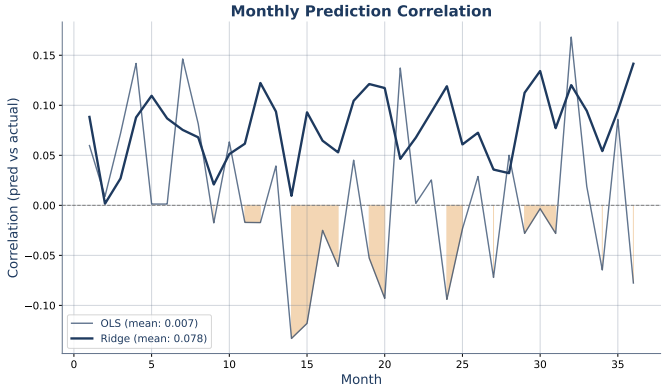
Finance: Regularized Factor Model

Many correlated predictors in factor investing. Lasso selects relevant factors; regularization prevents extreme allocations.



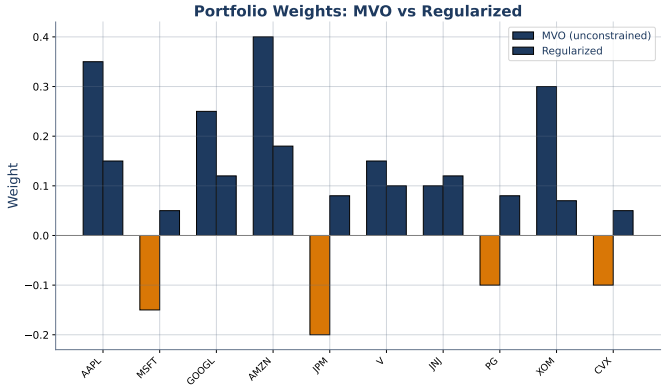
Regularized factor model vs OLS on out-of-sample data

Finance: Monthly Prediction



Rolling predictions from regularized model

Finance: Portfolio Weights



Regularization prevents extreme factor allocations

Hands-On Exercise (25 min)

Task: Compare Ridge vs Lasso on Multi-Factor Data

1. Create synthetic data with 20 features (only 5 truly predictive)
2. Fit OLS, Ridge, and Lasso models
3. Compare test set R^2 for each model
4. Plot Lasso coefficients – which features were selected?

Deliverable: Bar chart of coefficients comparing OLS vs Ridge vs Lasso.

Reference: `inclass/L22_inclass_lasso_selection.ipynb`

Extension: Use LassoCV to find optimal lambda and report selected features

Lesson Summary + Preview

Problem Solved: Regularization prevents overfitting when we have many predictors.

Key Takeaways:

- **Ridge** (L2, circle): shrinks all coefficients – handles multicollinearity
- **Lasso** (L1, diamond): sets some to zero – automatic feature selection
- Cross-validation (`RidgeCV`, `LassoCV`) tunes λ

Mnemonic:

- Ridge = Ridge keeps all features
- Lasso = Lasso **L**oses features (L for Lose)

Next Lesson: *How do we KNOW our model is good?*
L23: Regression Metrics – Trust But Verify.

Ridge keeps all. Lasso loses some. CV picks the best λ.