Ridge, Lasso, and Elastic Net

When More Features Means Worse Predictions

Jake Anderson

UCLA

March 3, 2026

Outline

What Happens When You Have 25 Features and 200 Observations?

You are a real estate analyst predicting house prices. You have data on **200 houses** and **25 features**: square footage, bedrooms, bathrooms, distance to school, crime rate, roof age, and many more.

What Happens When You Have 25 Features and 200 Observations?

You are a real estate analyst predicting house prices. You have data on **200 houses** and **25 features**: square footage, bedrooms, bathrooms, distance to school, crime rate, roof age, and many more.

Some of these features genuinely affect price. Others are noise. But you do not know which is which.

What Happens When You Have 25 Features and 200 Observations?

You are a real estate analyst predicting house prices. You have data on **200 houses** and **25 features**: square footage, bedrooms, bathrooms, distance to school, crime rate, roof age, and many more.

Some of these features genuinely affect price. Others are noise. But you do not know which is which.

You run OLS with all 25 features. The in-sample R^2 looks great. Then you test on new houses and your test RMSE is **20% higher** than the training RMSE.

What Happens When You Have 25 Features and 200 Observations?

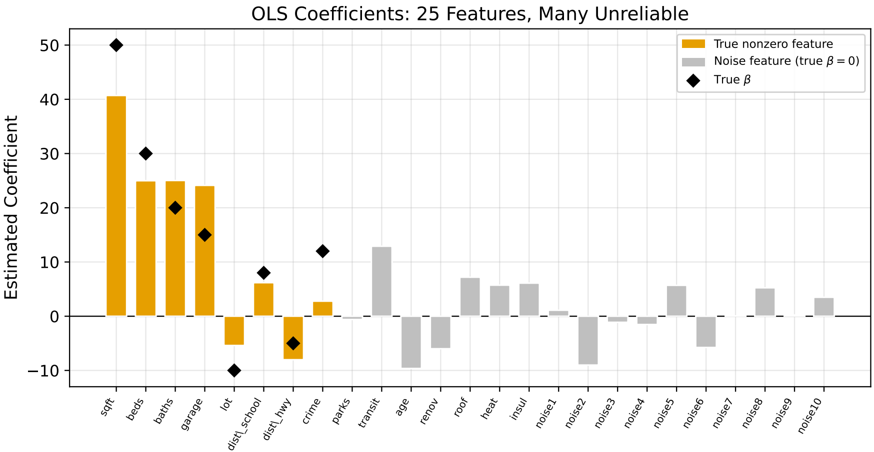
You are a real estate analyst predicting house prices. You have data on **200 houses** and **25 features**: square footage, bedrooms, bathrooms, distance to school, crime rate, roof age, and many more.

Some of these features genuinely affect price. Others are noise. But you do not know which is which.

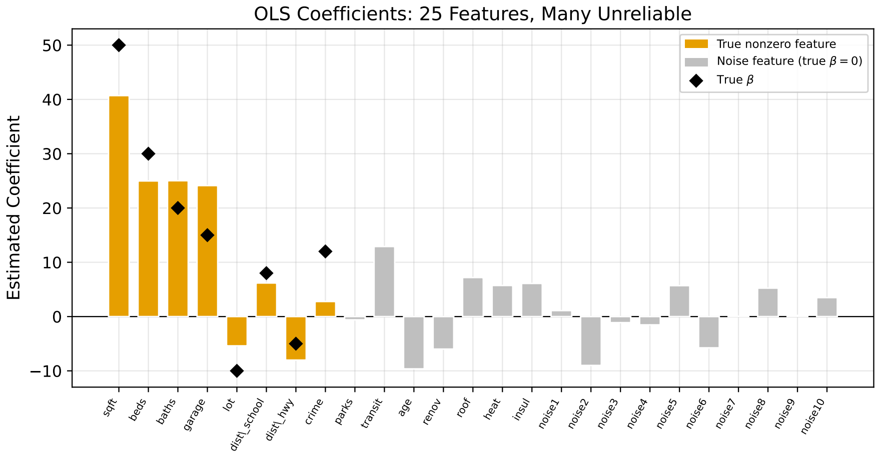
You run OLS with all 25 features. The in-sample R^2 looks great. Then you test on new houses and your test RMSE is **20% higher** than the training RMSE.

What went wrong?

OLS with 25 Features: Overfitting



OLS with 25 Features: Overfitting



OLS assigns large coefficients to many features. Some of these estimated effects are real. Others are noise that OLS is fitting by mistake. But from the OLS output alone, you cannot tell which is which.

Out-of-Sample Prediction: OLS Performs Poorly

Split the data: train on 120 houses, predict on 80.

Out-of-Sample Prediction: OLS Performs Poorly

Split the data: train on 120 houses, predict on 80.

- **In-sample** (training data): OLS fits well, R^2 is high.
- **Out-of-sample** (test data): predictions are far from actual prices.

Out-of-Sample Prediction: OLS Performs Poorly

Split the data: train on 120 houses, predict on 80.

- **In-sample** (training data): OLS fits well, R^2 is high.
- **Out-of-sample** (test data): predictions are far from actual prices.

The problem: OLS minimizes training error. It does not penalize complexity. With 25 features and only 120 training observations, OLS has enough freedom to chase noise.

Out-of-Sample Prediction: OLS Performs Poorly

Split the data: train on 120 houses, predict on 80.

- **In-sample** (training data): OLS fits well, R^2 is high.
- **Out-of-sample** (test data): predictions are far from actual prices.

The problem: OLS minimizes training error. It does not penalize complexity. With 25 features and only 120 training observations, OLS has enough freedom to chase noise.

This is **overfitting**: the model memorizes the training data instead of learning the underlying relationship.

Out-of-Sample Prediction: OLS Performs Poorly

Split the data: train on 120 houses, predict on 80.

- **In-sample** (training data): OLS fits well, R^2 is high.
- **Out-of-sample** (test data): predictions are far from actual prices.

The problem: OLS minimizes training error. It does not penalize complexity. With 25 features and only 120 training observations, OLS has enough freedom to chase noise.

This is **overfitting**: the model memorizes the training data instead of learning the underlying relationship.

$\Longrightarrow$ We need a method that deliberately sacrifices some in-sample fit to gain better out-of-sample prediction.

What Went Wrong: The Data Generating Process

It turns out only **8 of the 25** features truly affect price. The remaining 17 have true $\beta = 0$. OLS did not know this and tried to estimate all 25.

What Went Wrong: The Data Generating Process

It turns out only **8 of the 25** features truly affect price. The remaining 17 have true $\beta = 0$. OLS did not know this and tried to estimate all 25.

Feature Block	Features	Correlation	Active
Size (5 features)	sqft, beds, baths, garage, lot	$\rho = 0.7$	5 of 5
Location (5 features)	dist to school, hwy, crime, parks, transit	$\rho = 0.6$	3 of 5
Quality (5 features)	age, renovation, roof, heating, insulation	$\rho = 0.5$	0 of 5
Noise (10 features)	irrelevant variables	independent	0 of 10

What Went Wrong: The Data Generating Process

It turns out only **8 of the 25** features truly affect price. The remaining 17 have true $\beta = 0$. OLS did not know this and tried to estimate all 25.

Feature Block	Features	Correlation	Active
Size (5 features)	sqft, beds, baths, garage, lot	$\rho = 0.7$	5 of 5
Location (5 features)	dist to school, hwy, crime, parks, transit	$\rho = 0.6$	3 of 5
Quality (5 features)	age, renovation, roof, heating, insulation	$\rho = 0.5$	0 of 5
Noise (10 features)	irrelevant variables	independent	0 of 10

Within each block, features are correlated with each other. Bigger houses have more bedrooms, more bathrooms, and bigger lots. This **multicollinearity** inflates OLS standard errors and makes coefficient estimates unstable.

What Went Wrong: The Data Generating Process

It turns out only **8 of the 25** features truly affect price. The remaining 17 have true $\beta = 0$. OLS did not know this and tried to estimate all 25.

Feature Block	Features	Correlation	Active
Size (5 features)	sqft, beds, baths, garage, lot	$\rho = 0.7$	5 of 5
Location (5 features)	dist to school, hwy, crime, parks, transit	$\rho = 0.6$	3 of 5
Quality (5 features)	age, renovation, roof, heating, insulation	$\rho = 0.5$	0 of 5
Noise (10 features)	irrelevant variables	independent	0 of 10

Within each block, features are correlated with each other. Bigger houses have more bedrooms, more bathrooms, and bigger lots. This **multicollinearity** inflates OLS standard errors and makes coefficient estimates unstable.

$\Longrightarrow$ OLS failed because it treated noise features as if they were real, and multicollinearity made even the real estimates unreliable.

A Different Goal: Prediction, Not Causation

So far in this course, we have focused on **causal inference**: using IV, fixed effects, and random effects to estimate the effect of x on y.

A Different Goal: Prediction, Not Causation

So far in this course, we have focused on **causal inference**: using IV, fixed effects, and random effects to estimate the effect of x on y.

Now we shift to a different goal: **prediction**. Given a new house with known features, what price should we expect?

A Different Goal: Prediction, Not Causation

So far in this course, we have focused on **causal inference**: using IV, fixed effects, and random effects to estimate the effect of x on y.

Now we shift to a different goal: **prediction**. Given a new house with known features, what price should we expect?

The tools are different:

- Causal inference demands **unbiased, consistent** estimators. We accept higher variance to avoid bias.
- Prediction allows **some bias** if it substantially reduces variance. What counts is how well we predict new data, not whether each individual coefficient is "correct."

A Different Goal: Prediction, Not Causation

So far in this course, we have focused on **causal inference**: using IV, fixed effects, and random effects to estimate the effect of x on y.

Now we shift to a different goal: **prediction**. Given a new house with known features, what price should we expect?

The tools are different:

- Causal inference demands **unbiased, consistent** estimators. We accept higher variance to avoid bias.
- Prediction allows **some bias** if it substantially reduces variance. What counts is how well we predict new data, not whether each individual coefficient is "correct."

$\Longrightarrow$ Regularization is a prediction tool. It deliberately introduces bias to reduce variance.

Outline

Regularization: Adding a Penalty to OLS

Regularization means modifying the OLS objective to discourage large coefficients. Instead of minimizing only the sum of squared residuals, we add a **penalty term**:

Regularization: Adding a Penalty to OLS

Regularization means modifying the OLS objective to discourage large coefficients. Instead of minimizing only the sum of squared residuals, we add a **penalty term**:

$$\hat{\beta} = \arg\min_{\beta} \left\{ \underbrace{\sum_{i=1}^{n} (y_i - \hat{y}_i)^2}_{\text{fit to data}} + \underbrace{\lambda \cdot \text{Penalty}(\beta)}_{\text{shrinkage toward zero}} \right\}$$

Regularization: Adding a Penalty to OLS

Regularization means modifying the OLS objective to discourage large coefficients. Instead of minimizing only the sum of squared residuals, we add a **penalty term**:

$$\hat{\beta} = \arg\min_{\beta} \left\{ \underbrace{\sum_{i=1}^{n} (y_i - \hat{y}_i)^2}_{\text{fit to data}} + \underbrace{\lambda \cdot \text{Penalty}(\beta)}_{\text{shrinkage toward zero}} \right\}$$

Read $\arg\min_{\beta}$ as: find the value of β that makes this expression as small as possible.

Regularization: Adding a Penalty to OLS

Regularization means modifying the OLS objective to discourage large coefficients. Instead of minimizing only the sum of squared residuals, we add a **penalty term**:

$$\hat{\beta} = \arg\min_{\beta} \left\{ \underbrace{\sum_{i=1}^{n} (y_i - \hat{y}_i)^2}_{\text{fit to data}} + \underbrace{\lambda \cdot \text{Penalty}(\beta)}_{\text{shrinkage toward zero}} \right\}$$

Read $\arg\min_{\beta}$ as: find the value of β that makes this expression as small as possible.

- $\lambda \geq 0$ controls the **strength** of regularization (sometimes called the **tuning parameter**)
- $\lambda = 0$: no penalty, we recover OLS
- $\lambda \to \infty$: penalty dominates, all coefficients shrink toward zero

Regularization: Adding a Penalty to OLS

Regularization means modifying the OLS objective to discourage large coefficients. Instead of minimizing only the sum of squared residuals, we add a **penalty term**:

$$\hat{\beta} = \arg\min_{\beta} \left\{ \underbrace{\sum_{i=1}^{n} (y_i - \hat{y}_i)^2}_{\text{fit to data}} + \underbrace{\lambda \cdot \text{Penalty}(\beta)}_{\text{shrinkage toward zero}} \right\}$$

Read $\arg\min_{\beta}$ as: find the value of β that makes this expression as small as possible.

- $\lambda \geq 0$ controls the **strength** of regularization (sometimes called the **tuning parameter**)
- $\lambda = 0$: no penalty, we recover OLS
- $\lambda \to \infty$: penalty dominates, all coefficients shrink toward zero

What kind of penalty should we use? The three main choices: Ridge (ℓ_2), Lasso (ℓ_1), and Elastic Net (both). The subscript tells you the exponent: ℓ_2 squares each coefficient, ℓ_1 takes absolute values.

Why Adding a Penalty Helps: The Bias-Variance Tradeoff

Adding a penalty introduces bias (estimates are pulled toward zero), but it reduces variance (estimates are more stable across samples). Every prediction method faces this tension:

Why Adding a Penalty Helps: The Bias-Variance Tradeoff

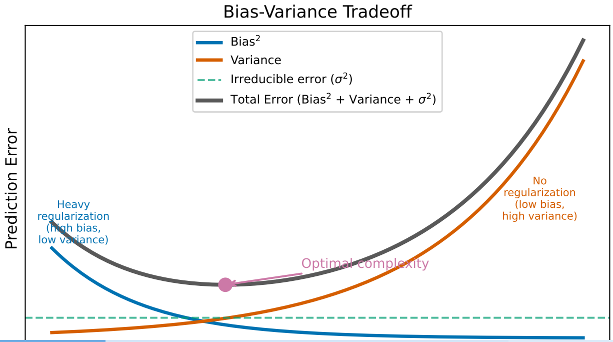
Adding a penalty introduces bias (estimates are pulled toward zero), but it reduces variance (estimates are more stable across samples). Every prediction method faces this tension:

- **Bias**: how far is the average prediction from the truth? A heavily penalized model has high bias.
- **Variance**: how much do predictions change across different training samples? An unpenalized model (OLS with many features) has high variance.

Why Adding a Penalty Helps: The Bias-Variance Tradeoff

Adding a penalty introduces bias (estimates are pulled toward zero), but it reduces variance (estimates are more stable across samples). Every prediction method faces this tension:

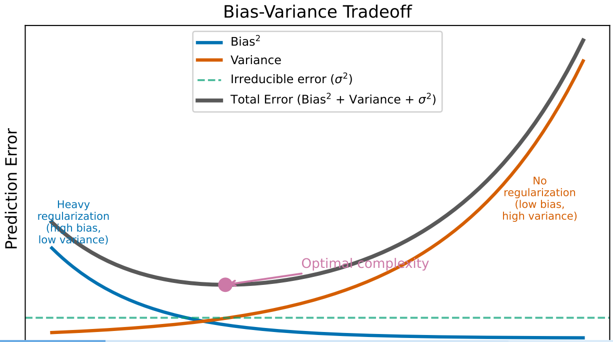
- **Bias**: how far is the average prediction from the truth? A heavily penalized model has high bias.
- **Variance**: how much do predictions change across different training samples? An unpenalized model (OLS with many features) has high variance.



Why Adding a Penalty Helps: The Bias-Variance Tradeoff

Adding a penalty introduces bias (estimates are pulled toward zero), but it reduces variance (estimates are more stable across samples). Every prediction method faces this tension:

- **Bias**: how far is the average prediction from the truth? A heavily penalized model has high bias.
- **Variance**: how much do predictions change across different training samples? An unpenalized model (OLS with many features) has high variance.



Outline

Ridge Regression: The ℓ_2 Penalty

Ridge regression (Hoerl and Kennard, 1970) adds the **sum of squared coefficients**:

Ridge Regression: The ℓ_2 Penalty

Ridge regression (Hoerl and Kennard, 1970) adds the **sum of squared coefficients**:

$$\hat{\beta}^{\text{Ridge}} = \arg\min_{\beta} \left\{ \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 + \lambda \sum_{j=1}^{p} \beta_j^2 \right\}$$

where p is the number of features.

Ridge Regression: The ℓ_2 Penalty

Ridge regression (Hoerl and Kennard, 1970) adds the **sum of squared coefficients**:

$$\hat{\beta}^{\text{Ridge}} = \arg\min_{\beta} \left\{ \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 + \lambda \sum_{j=1}^{p} \beta_j^2 \right\}$$

where p is the number of features.

The term $\sum_j \beta_j^2$ is the ℓ_2 **norm** (squared) of the coefficient vector.

Ridge Regression: The ℓ_2 Penalty

Ridge regression (Hoerl and Kennard, 1970) adds the **sum of squared coefficients**:

$$\hat{\beta}^{\text{Ridge}} = \arg\min_{\beta} \left\{ \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 + \lambda \sum_{j=1}^{p} \beta_j^2 \right\}$$

where p is the number of features.

The term $\sum_j \beta_j^2$ is the ℓ_2 **norm** (squared) of the coefficient vector.

In words: "fit the data well, but keep the coefficients small." The penalty grows with the square of each coefficient, so it penalizes large coefficients more heavily.

Ridge Regression: The ℓ_2 Penalty

Ridge regression (Hoerl and Kennard, 1970) adds the **sum of squared coefficients**:

$$\hat{\beta}^{\text{Ridge}} = \arg\min_{\beta} \left\{ \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 + \lambda \sum_{j=1}^{p} \beta_j^2 \right\}$$

where p is the number of features.

The term $\sum_j \beta_j^2$ is the ℓ_2 **norm** (squared) of the coefficient vector.

In words: "fit the data well, but keep the coefficients small." The penalty grows with the square of each coefficient, so it penalizes large coefficients more heavily.

Note: we standardize all features before fitting (subtract mean, divide by SD). This ensures the penalty treats all features equally, regardless of their original scale. The intercept is not penalized.

What Does λ Control?

Think of λ as a dial between two extremes:

What Does λ Control?

Think of λ as a dial between two extremes:

λ	**Model Behavior**	**Bias-Variance**
$\lambda = 0$	OLS (no penalty)	Low bias, high variance
λ small	Mild shrinkage	Slight bias, less variance
λ large	Heavy shrinkage	High bias, low variance
$\lambda \to \infty$	All $\hat{\beta}_j \to 0$	Maximum bias, zero variance

What Does λ Control?

Think of λ as a dial between two extremes:

λ	**Model Behavior**	**Bias-Variance**
$\lambda = 0$	OLS (no penalty)	Low bias, high variance
λ small	Mild shrinkage	Slight bias, less variance
λ large	Heavy shrinkage	High bias, low variance
$\lambda \to \infty$	All $\hat{\beta}_j \to 0$	Maximum bias, zero variance

As λ increases, Ridge **shrinks** every coefficient toward zero. It never sets any coefficient exactly *to* zero.

What Does λ Control?

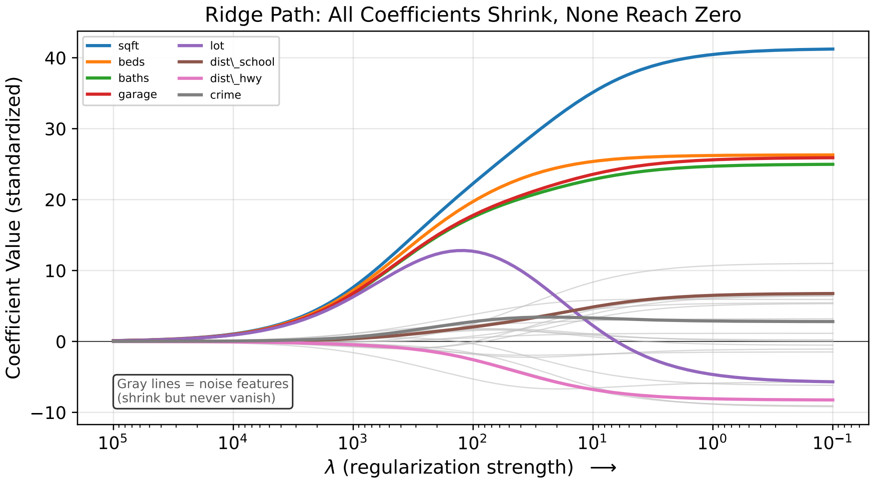
Think of λ as a dial between two extremes:

λ	**Model Behavior**	**Bias-Variance**
$\lambda = 0$	OLS (no penalty)	Low bias, high variance
λ small	Mild shrinkage	Slight bias, less variance
λ large	Heavy shrinkage	High bias, low variance
$\lambda \to \infty$	All $\hat{\beta}_j \to 0$	Maximum bias, zero variance

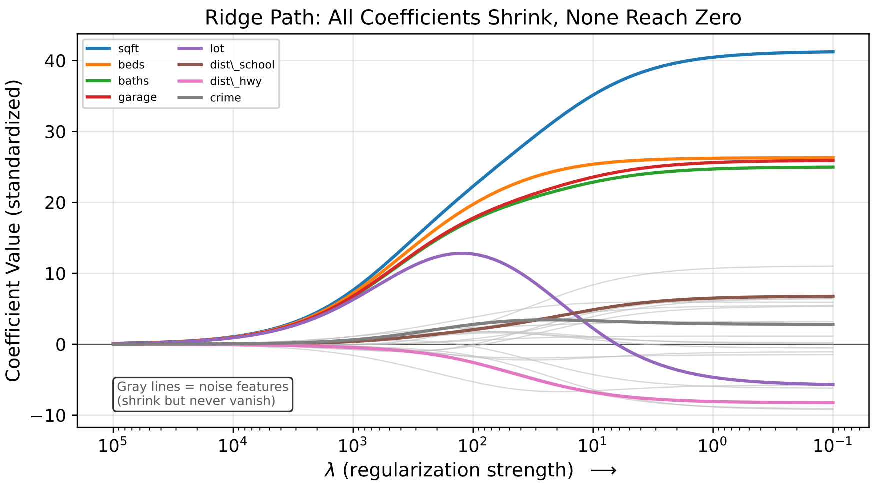
As λ increases, Ridge **shrinks** every coefficient toward zero. It never sets any coefficient exactly *to* zero.

$\Longrightarrow$ Ridge always keeps all features in the model. It reduces their influence but does not remove them.

Ridge Coefficient Path



Ridge Coefficient Path



As λ increases (left to right), every coefficient shrinks toward zero. The noise features (gray) become negligible, but no coefficient ever reaches exactly zero.

Ridge Properties

For a single-feature model (with centered data, i.e., $\bar{x} = 0$):

$$\hat{\beta}^{\text{Ridge}} = \frac{\sum_i x_i y_i}{\sum_i x_i^2 + \lambda}$$

Ridge Properties

For a single-feature model (with centered data, i.e., $\bar{x} = 0$):

$$\hat{\beta}^{\text{Ridge}} = \frac{\sum_i x_i y_i}{\sum_i x_i^2 + \lambda}$$

Compare to OLS: $\hat{\beta}^{\text{OLS}} = \frac{\sum_i x_i y_i}{\sum_i x_i^2}$. Ridge adds λ to the denominator, shrinking the estimate toward zero.

Ridge Properties

For a single-feature model (with centered data, i.e., $\bar{x} = 0$):

$$\hat{\beta}^{\text{Ridge}} = \frac{\sum_i x_i y_i}{\sum_i x_i^2 + \lambda}$$

Compare to OLS: $\hat{\beta}^{\text{OLS}} = \frac{\sum_i x_i y_i}{\sum_i x_i^2}$. Ridge adds λ to the denominator, shrinking the estimate toward zero.

When does Ridge help most?

- Many correlated features (multicollinearity inflates OLS variance)
- More features than observations ($p > n$), where OLS has no unique solution
- Goal is **prediction**, not identifying which features are active

Ridge Properties

For a single-feature model (with centered data, i.e., $\bar{x} = 0$):

$$\hat{\beta}^{\text{Ridge}} = \frac{\sum_i x_i y_i}{\sum_i x_i^2 + \lambda}$$

Compare to OLS: $\hat{\beta}^{\text{OLS}} = \frac{\sum_i x_i y_i}{\sum_i x_i^2}$. Ridge adds λ to the denominator, shrinking the estimate toward zero.

When does Ridge help most?

- Many correlated features (multicollinearity inflates OLS variance)
- More features than observations ($p > n$), where OLS has no unique solution
- Goal is **prediction**, not identifying which features are active

$\implies$ Ridge is a workhorse for prediction with correlated features, but it does not tell you which features to drop.

Can We Do Better Than "Shrink Everything"?

Ridge shrinks all 25 coefficients toward zero. That helps with variance, but we suspect most of those features are noise.

Can We Do Better Than “Shrink Everything”?

Ridge shrinks all 25 coefficients toward zero. That helps with variance, but we suspect most of those features are noise.

What if we could shrink the noise features all the way to **exactly zero**? That would give us:

- A simpler, more interpretable model
- Automatic **feature selection**: the model tells us which features to keep

Can We Do Better Than "Shrink Everything"?

Ridge shrinks all 25 coefficients toward zero. That helps with variance, but we suspect most of those features are noise.

What if we could shrink the noise features all the way to **exactly zero**? That would give us:

- A simpler, more interpretable model
- Automatic **feature selection**: the model tells us which features to keep

All we need to change is the shape of the penalty.

Outline

Lasso: The ℓ_1 Penalty

The Lasso (Tibshirani, 1996) replaces the squared penalty with an **absolute value** penalty:

Lasso: The ℓ_1 Penalty

The Lasso (Tibshirani, 1996) replaces the squared penalty with an **absolute value** penalty:

$$\hat{\beta}^{\text{Lasso}} = \arg\min_{\beta} \left\{ \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 + \lambda \sum_{j=1}^{p} |\beta_j| \right\}$$

Lasso: The ℓ_1 Penalty

The Lasso (Tibshirani, 1996) replaces the squared penalty with an **absolute value** penalty:

$$\hat{\beta}^{\text{Lasso}} = \arg\min_{\beta} \left\{ \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 + \lambda \sum_{j=1}^{p} |\beta_j| \right\}$$

Lasso stands for "Least Absolute Shrinkage and Selection Operator." The ℓ_1 penalty ($\sum_j |\beta_j|$) is the sum of the absolute values.

Lasso: The ℓ_1 Penalty

The Lasso (Tibshirani, 1996) replaces the squared penalty with an **absolute value** penalty:

$$\hat{\beta}^{\text{Lasso}} = \arg\min_{\beta} \left\{ \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 + \lambda \sum_{j=1}^{p} |\beta_j| \right\}$$

Lasso stands for "Least Absolute Shrinkage and Selection Operator." The ℓ_1 penalty ($\sum_j |\beta_j|$) is the sum of the absolute values.

Why does this small change produce a fundamentally different result?

Lasso: The ℓ_1 Penalty

The Lasso (Tibshirani, 1996) replaces the squared penalty with an **absolute value** penalty:

$$\hat{\beta}^{\text{Lasso}} = \arg\min_{\beta} \left\{ \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 + \lambda \sum_{j=1}^{p} |\beta_j| \right\}$$

Lasso stands for "Least Absolute Shrinkage and Selection Operator." The ℓ_1 penalty ($\sum_j |\beta_j|$) is the sum of the absolute values.

Why does this small change produce a fundamentally different result?

- The ℓ_2 penalty (Ridge) costs more as $|\beta_j|$ grows, so it pushes large coefficients hard but barely touches small ones.
- The ℓ_1 penalty (Lasso) charges a **constant rate** per unit of $|\beta_j|$, so it pushes small coefficients all the way to zero. If $\beta = 0.1$, the ℓ_2 cost is 0.01 (negligible) but the ℓ_1 cost is 0.1 (same rate as for $\beta = 10$).

Lasso: The ℓ_1 Penalty

The Lasso (Tibshirani, 1996) replaces the squared penalty with an **absolute value** penalty:

$$\hat{\beta}^{\text{Lasso}} = \arg\min_{\beta} \left\{ \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 + \lambda \sum_{j=1}^{p} |\beta_j| \right\}$$

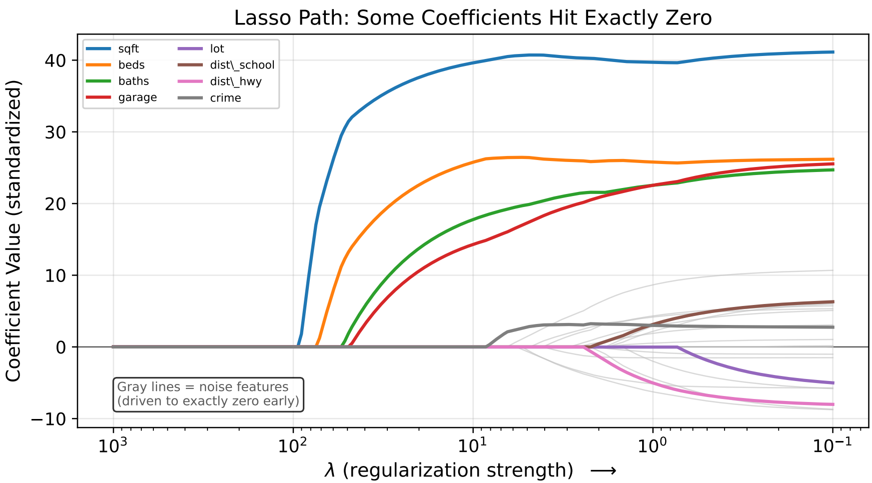
Lasso stands for "Least Absolute Shrinkage and Selection Operator." The ℓ_1 penalty ($\sum_j |\beta_j|$) is the sum of the absolute values.

Why does this small change produce a fundamentally different result?

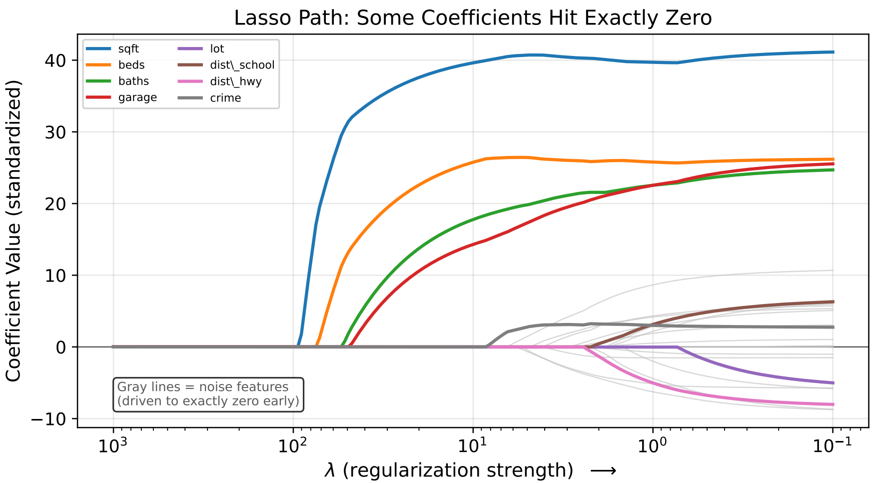
- The ℓ_2 penalty (Ridge) costs more as $|\beta_j|$ grows, so it pushes large coefficients hard but barely touches small ones.
- The ℓ_1 penalty (Lasso) charges a **constant rate** per unit of $|\beta_j|$, so it pushes small coefficients all the way to zero. If $\beta = 0.1$, the ℓ_2 cost is 0.01 (negligible) but the ℓ_1 cost is 0.1 (same rate as for $\beta = 10$).

$\Longrightarrow$ Lasso performs **feature selection**: it sets some $\hat{\beta}_j$ to exactly zero.

Lasso Coefficient Path

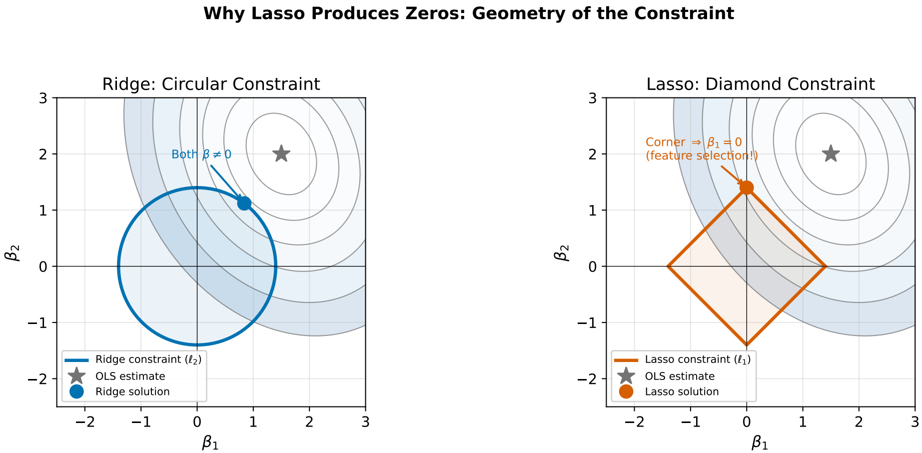


Lasso Coefficient Path

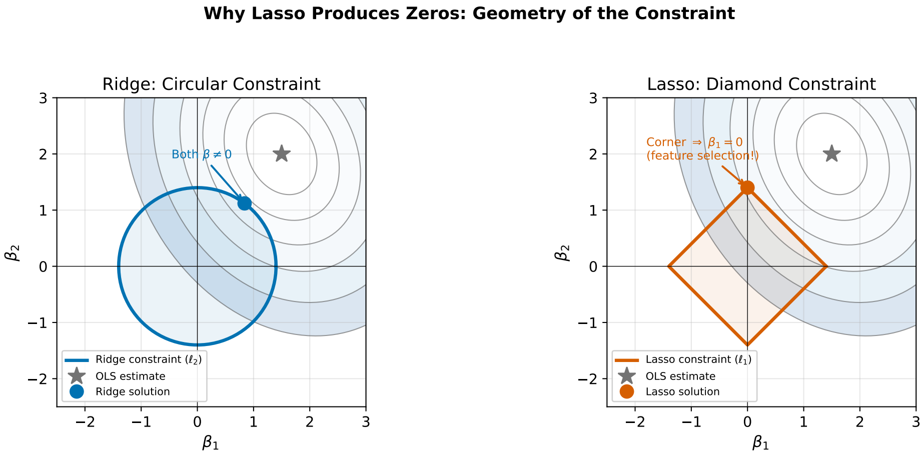


As λ increases, noise features (gray) are driven to zero **first**. The true signal features persist longer. At the right level of λ, only the genuinely predictive features remain.

Why Does Lasso Produce Zeros? The Geometry

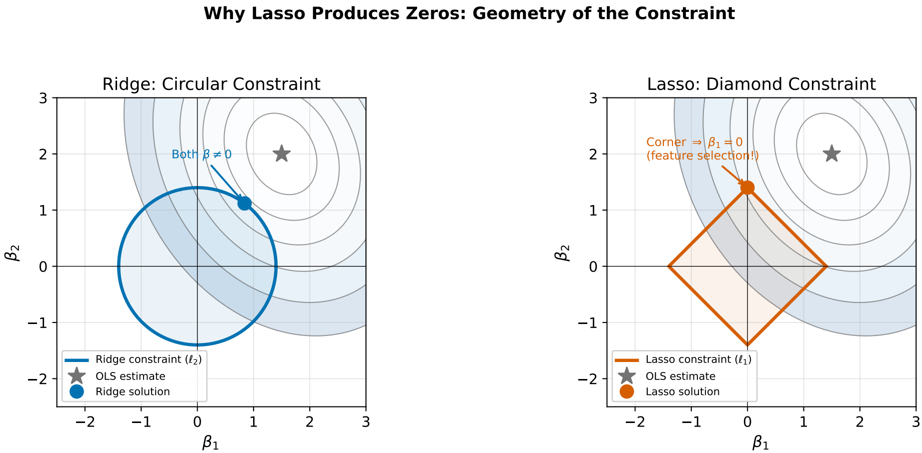


Why Does Lasso Produce Zeros? The Geometry



Each ellipse is a set of (β_1, β_2) pairs giving the same RSS. As we shrink the constraint region, the first contact point is the solution. Ridge uses a **circle** ($\beta_1^2 + \beta_2^2 \leq t$); Lasso uses a **diamond** ($|\beta_1| + |\beta_2| \leq t$).

Why Does Lasso Produce Zeros? The Geometry



Each ellipse is a set of (β_1, β_2) pairs giving the same RSS. As we shrink the constraint region, the first contact point is the solution. Ridge uses a **circle** ($\beta_1^2 + \beta_2^2 \leq t$); Lasso uses a **diamond** ($|\beta_1| + |\beta_2| \leq t$).

Lasso as Automatic Feature Selection

In our house price example, the Lasso with a well-chosen λ retains about a dozen features and sets the rest to zero.

Lasso as Automatic Feature Selection

In our house price example, the Lasso with a well-chosen λ retains about a dozen features and sets the rest to zero.

Compare the approaches:

Method	**Features Retained**	**Zeros?**
OLS	All 25	No
Ridge	All 25 (small)	No
Lasso	$\sim$10–12	Yes

Lasso as Automatic Feature Selection

In our house price example, the Lasso with a well-chosen λ retains about a dozen features and sets the rest to zero.

Compare the approaches:

Method	Features Retained	Zeros?
OLS	All 25	No
Ridge	All 25 (small)	No
Lasso	$\sim$10–12	Yes

Lasso gives you a short list of the features that the model considers most predictive. This is useful when you want an interpretable model, not just a prediction.

Lasso as Automatic Feature Selection

In our house price example, the Lasso with a well-chosen λ retains about a dozen features and sets the rest to zero.

Compare the approaches:

Method	**Features Retained**	**Zeros?**
OLS	All 25	No
Ridge	All 25 (small)	No
Lasso	$\sim$10–12	Yes

Lasso gives you a short list of the features that the model considers most predictive. This is useful when you want an interpretable model, not just a prediction.

$\Longrightarrow$ Lasso serves double duty: it improves prediction *and* identifies which variables to keep.

Lasso Properties

Strengths:

- Automatic feature selection (sparse solutions)
- Works well when only a few features are truly relevant
- Interpretable: the surviving features are the model's "short list"

Lasso Properties

Strengths:

- Automatic feature selection (sparse solutions)
- Works well when only a few features are truly relevant
- Interpretable: the surviving features are the model's "short list"

Limitations:

- With highly correlated features, Lasso tends to pick **one** from each correlated group and set the others to zero (arbitrarily)
- Cannot select more than n features when $p > n$
- The ℓ_1 penalty has no closed-form solution (requires iterative algorithms)

Lasso Properties

Strengths:

- Automatic feature selection (sparse solutions)
- Works well when only a few features are truly relevant
- Interpretable: the surviving features are the model's "short list"

Limitations:

- With highly correlated features, Lasso tends to pick **one** from each correlated group and set the others to zero (arbitrarily)
- Cannot select more than n features when $p > n$
- The ℓ_1 penalty has no closed-form solution (requires iterative algorithms)

$\Longrightarrow$ When features come in correlated groups (as in our house data), Lasso's selection can be unstable. This motivates combining the two penalties.

Can We Get Sparsity Without Instability?

Lasso gives us feature selection, but it has a specific weakness with correlated features.

Can We Get Sparsity Without Instability?

Lasso gives us feature selection, but it has a specific weakness with correlated features.

In our house data, sqft, bedrooms, and bathrooms are all correlated ($\rho = 0.7$). Run Lasso on the full sample and it might keep sqft but drop bedrooms. Run it again on a slightly different subsample and it might keep bedrooms but drop sqft.

Can We Get Sparsity Without Instability?

Lasso gives us feature selection, but it has a specific weakness with correlated features.

In our house data, sqft, bedrooms, and bathrooms are all correlated ($\rho = 0.7$). Run Lasso on the full sample and it might keep sqft but drop bedrooms. Run it again on a slightly different subsample and it might keep bedrooms but drop sqft.

The problem: among a group of correlated features, Lasso picks one **arbitrarily** and drops the rest. Which one it picks depends on small fluctuations in the training data.

Can We Get Sparsity Without Instability?

Lasso gives us feature selection, but it has a specific weakness with correlated features.

In our house data, sqft, bedrooms, and bathrooms are all correlated ($\rho = 0.7$). Run Lasso on the full sample and it might keep sqft but drop bedrooms. Run it again on a slightly different subsample and it might keep bedrooms but drop sqft.

The problem: among a group of correlated features, Lasso picks one **arbitrarily** and drops the rest. Which one it picks depends on small fluctuations in the training data.

We want a method that:

- Still sets noise features to zero (like Lasso)
- Keeps or drops correlated features **together** (like Ridge)

Can We Get Sparsity Without Instability?

Lasso gives us feature selection, but it has a specific weakness with correlated features.

In our house data, sqft, bedrooms, and bathrooms are all correlated ($\rho = 0.7$). Run Lasso on the full sample and it might keep sqft but drop bedrooms. Run it again on a slightly different subsample and it might keep bedrooms but drop sqft.

The problem: among a group of correlated features, Lasso picks one **arbitrarily** and drops the rest. Which one it picks depends on small fluctuations in the training data.

We want a method that:

- Still sets noise features to zero (like Lasso)
- Keeps or drops correlated features **together** (like Ridge)

$\Longrightarrow$ Combine the ℓ_1 and ℓ_2 penalties into a single objective.

Outline

Elastic Net: Combining ℓ_1 and ℓ_2

The Elastic Net (Zou and Hastie, 2005) uses **both** penalties:

Elastic Net: Combining ℓ_1 and ℓ_2

The Elastic Net (Zou and Hastie, 2005) uses **both** penalties:

$$\hat{\beta}^{\text{EN}} = \arg\min_{\beta} \left\{ \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 + \lambda \left[\alpha \sum_{j=1}^{p} |\beta_j| + \frac{1-\alpha}{2} \sum_{j=1}^{p} \beta_j^2 \right] \right\}$$

Elastic Net: Combining ℓ_1 and ℓ_2

The Elastic Net (Zou and Hastie, 2005) uses **both** penalties:

$$\hat{\beta}^{\text{EN}} = \arg\min_{\beta} \left\{ \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 + \lambda \left[\alpha \sum_{j=1}^{p} |\beta_j| + \frac{1-\alpha}{2} \sum_{j=1}^{p} \beta_j^2 \right] \right\}$$

Two tuning parameters:

- $\lambda \geq 0$: overall regularization strength (same role as before)
- $\alpha \in [0, 1]$: the **mixing parameter** between Lasso ($\alpha = 1$) and Ridge ($\alpha = 0$)

Note: this α is unrelated to the significance level from hypothesis testing. The $\frac{1}{2}$ in the ℓ_2 component is a standard convention; since λ is chosen by CV, this scaling is absorbed into tuning.

Elastic Net: Combining ℓ_1 and ℓ_2

The Elastic Net (Zou and Hastie, 2005) uses **both** penalties:

$$\hat{\beta}^{\text{EN}} = \arg\min_{\beta} \left\{ \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 + \lambda \left[\alpha \sum_{j=1}^{p} |\beta_j| + \frac{1-\alpha}{2} \sum_{j=1}^{p} \beta_j^2 \right] \right\}$$

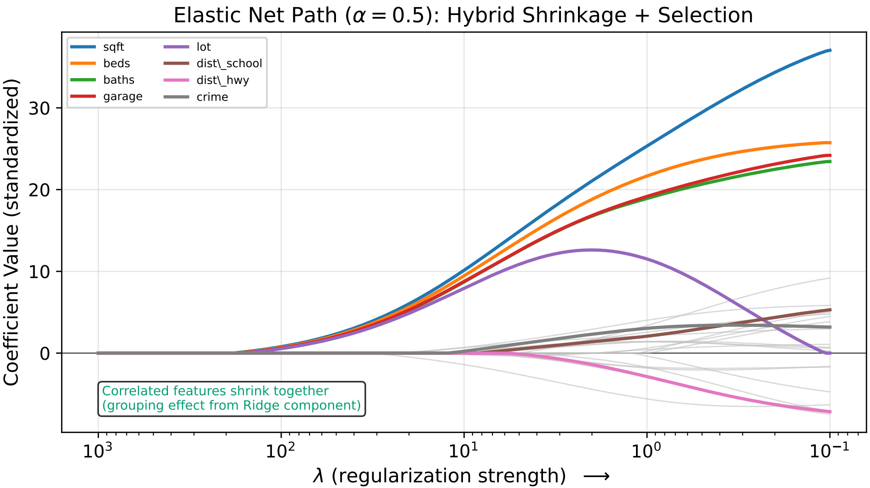
Two tuning parameters:

- $\lambda \geq 0$: overall regularization strength (same role as before)
- $\alpha \in [0, 1]$: the **mixing parameter** between Lasso ($\alpha = 1$) and Ridge ($\alpha = 0$)

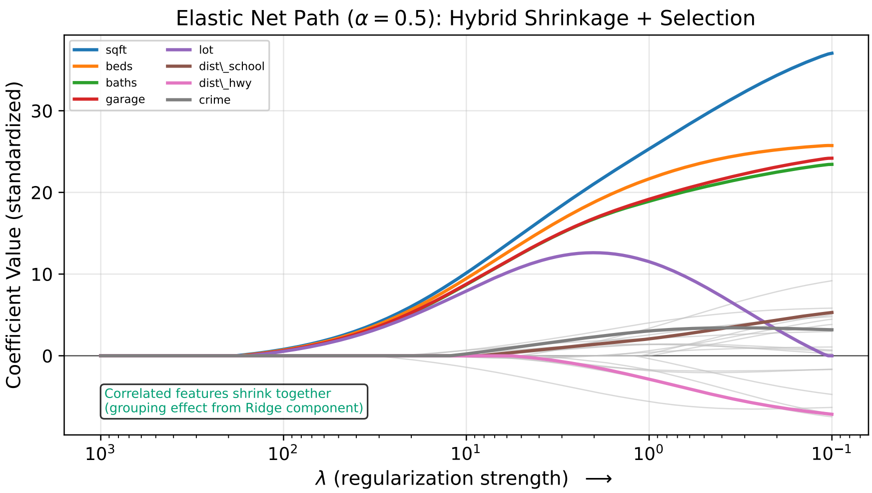
Note: this α is unrelated to the significance level from hypothesis testing. The $\frac{1}{2}$ in the ℓ_2 component is a standard convention; since λ is chosen by CV, this scaling is absorbed into tuning.

α	**Method**	**Behavior**
$\alpha = 1$	Lasso	Feature selection, no grouping
$\alpha = 0$	Ridge	Shrinkage only, no selection
$0 < \alpha < 1$	Elastic Net	Selection + grouping

Elastic Net Coefficient Path



Elastic Net Coefficient Path



Like Lasso, noise features are driven to zero. Like Ridge, correlated features within the same block

When to Use Elastic Net

Elastic Net is designed for situations where Lasso alone struggles:

When to Use Elastic Net

Elastic Net is designed for situations where Lasso alone struggles:

1. **Correlated feature groups.** Our house data has blocks of correlated features (sqft, beds, baths all measure "size"). Lasso picks one arbitrarily; Elastic Net keeps or drops them together.

When to Use Elastic Net

Elastic Net is designed for situations where Lasso alone struggles:

1. **Correlated feature groups.** Our house data has blocks of correlated features (sqft, beds, baths all measure "size"). Lasso picks one arbitrarily; Elastic Net keeps or drops them together.
2. $p > n$ **(more features than observations).** Lasso can select at most n features; Elastic Net has no such restriction.

When to Use Elastic Net

Elastic Net is designed for situations where Lasso alone struggles:

1. **Correlated feature groups.** Our house data has blocks of correlated features (sqft, beds, baths all measure "size"). Lasso picks one arbitrarily; Elastic Net keeps or drops them together.
2. **$p > n$ (more features than observations).** Lasso can select at most n features; Elastic Net has no such restriction.
3. **Stability.** Small changes in the training data can flip which correlated feature Lasso selects. The Ridge component of Elastic Net stabilizes the selection.

When to Use Elastic Net

Elastic Net is designed for situations where Lasso alone struggles:

1. **Correlated feature groups.** Our house data has blocks of correlated features (sqft, beds, baths all measure "size"). Lasso picks one arbitrarily; Elastic Net keeps or drops them together.
2. $p > n$ **(more features than observations).** Lasso can select at most n features; Elastic Net has no such restriction.
3. **Stability.** Small changes in the training data can flip which correlated feature Lasso selects. The Ridge component of Elastic Net stabilizes the selection.

In practice, many analysts default to Elastic Net with $\alpha = 0.5$ and let cross-validation choose λ.

When to Use Elastic Net

Elastic Net is designed for situations where Lasso alone struggles:

1. **Correlated feature groups.** Our house data has blocks of correlated features (sqft, beds, baths all measure "size"). Lasso picks one arbitrarily; Elastic Net keeps or drops them together.
2. $p > n$ **(more features than observations).** Lasso can select at most n features; Elastic Net has no such restriction.
3. **Stability.** Small changes in the training data can flip which correlated feature Lasso selects. The Ridge component of Elastic Net stabilizes the selection.

In practice, many analysts default to Elastic Net with $\alpha = 0.5$ and let cross-validation choose λ.

$\Longrightarrow$ Elastic Net inherits the best of both worlds: sparsity from Lasso and stability from Ridge.

Outline

How Do We Choose λ?

Now that we have three methods (Ridge, Lasso, Elastic Net), each requires choosing λ. Too small $\implies$ overfitting. Too large $\implies$ underfitting.

How Do We Choose λ?

Now that we have three methods (Ridge, Lasso, Elastic Net), each requires choosing λ. Too small $\implies$ overfitting. Too large $\implies$ underfitting.

We cannot use training error to choose λ (training error always decreases with less regularization). Instead, we estimate **out-of-sample error** using k-**fold cross-validation**:

How Do We Choose λ?

Now that we have three methods (Ridge, Lasso, Elastic Net), each requires choosing λ. Too small $\implies$ overfitting. Too large $\implies$ underfitting.

We cannot use training error to choose λ (training error always decreases with less regularization). Instead, we estimate **out-of-sample error** using k-**fold cross-validation**:

1. Split the training data into k equal folds (commonly $k = 10$).
2. For each candidate λ:
   - Hold out fold j, train on the remaining $k - 1$ folds, predict fold j.
   - Repeat for $j = 1, \ldots, k$. Average the prediction errors.
3. Pick the λ that minimizes the average CV error.

How Do We Choose λ?

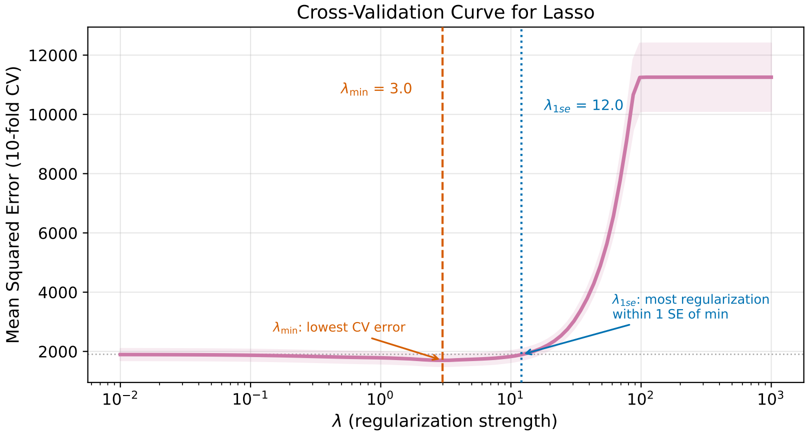
Now that we have three methods (Ridge, Lasso, Elastic Net), each requires choosing λ. Too small $\implies$ overfitting. Too large $\implies$ underfitting.

We cannot use training error to choose λ (training error always decreases with less regularization). Instead, we estimate **out-of-sample error** using k-**fold cross-validation**:

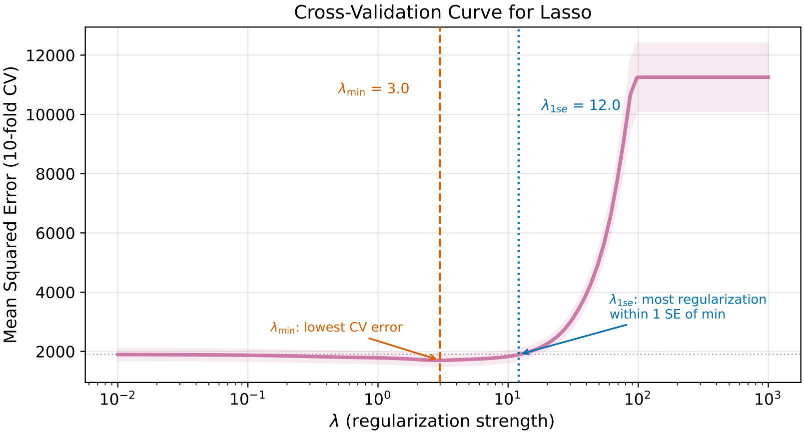
1. Split the training data into k equal folds (commonly $k = 10$).
2. For each candidate λ:
   - Hold out fold j, train on the remaining $k - 1$ folds, predict fold j.
   - Repeat for $j = 1, \ldots, k$. Average the prediction errors.
3. Pick the λ that minimizes the average CV error.

$\implies$ Cross-validation simulates out-of-sample prediction using only the training data. It selects λ without touching the test set.

The CV Curve: $\lambda_{\min}$ and λ_{1se}



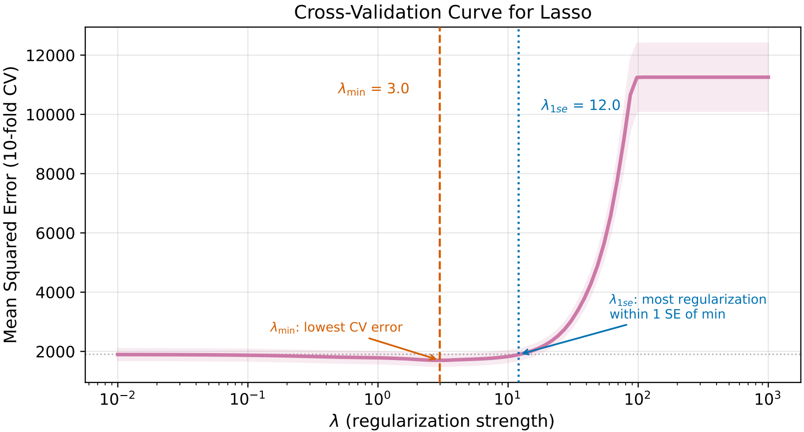
The CV Curve: λ_{min} and λ_{1se}



Two common choices:

- λ_{min}: the λ with the lowest CV error (best prediction).

The CV Curve: λ_{min} and λ_{1se}

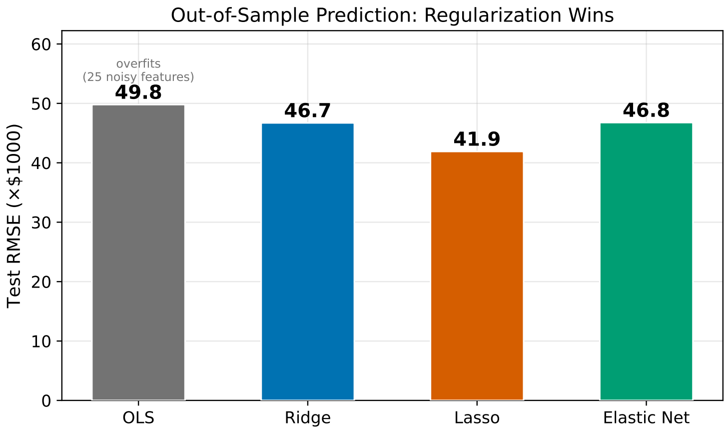


Two common choices:

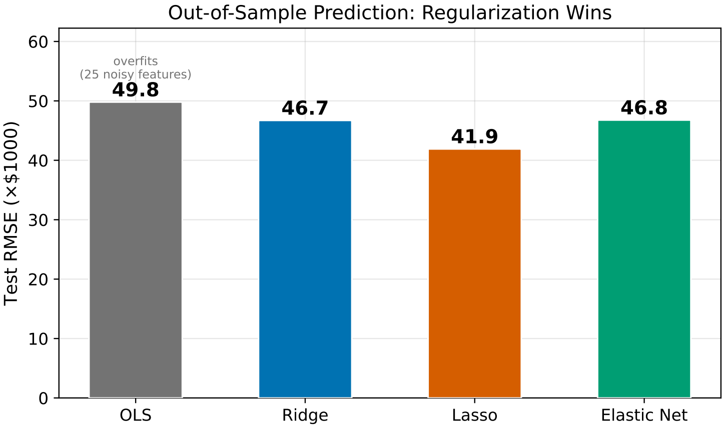
- λ_{min}: the λ with the lowest CV error (best prediction).

Outline

Prediction Comparison: Test RMSE



Prediction Comparison: Test RMSE



All three regularized methods beat OLS on out-of-sample prediction. Ridge, Lasso, and Elastic Net perform similarly here; the improvement comes from penalizing noise features.

Ridge vs. Lasso vs. Elastic Net: When to Use Which

Scenario	**Ridge**	**Lasso**	**EN**
Many correlated features	✓		✓
Want feature selection		✓	✓
Few true signals, many noise		✓	✓
$p > n$	✓	(limited)	✓
Correlated groups + sparsity			✓

Ridge vs. Lasso vs. Elastic Net: When to Use Which

Scenario	Ridge	Lasso	EN
Many correlated features	✓		✓
Want feature selection		✓	✓
Few true signals, many noise		✓	✓
$p > n$	✓	(limited)	✓
Correlated groups + sparsity			✓

Rules of thumb:

- If all features might be relevant (dense model) $\implies$ Ridge
- If only a few features are relevant (sparse model) $\implies$ Lasso
- If features come in correlated groups $\implies$ Elastic Net
- Unsure? $\implies$ Elastic Net with $\alpha = 0.5$ is a safe default

Ridge vs. Lasso vs. Elastic Net: When to Use Which

Scenario	Ridge	Lasso	EN
Many correlated features	✓		✓
Want feature selection		✓	✓
Few true signals, many noise		✓	✓
$p > n$	✓	(limited)	✓
Correlated groups + sparsity			✓

Rules of thumb:

- If all features might be relevant (dense model) $\implies$ Ridge
- If only a few features are relevant (sparse model) $\implies$ Lasso
- If features come in correlated groups $\implies$ Elastic Net
- Unsure? $\implies$ Elastic Net with $\alpha = 0.5$ is a safe default

$\implies$ In all cases, choose λ by cross-validation. Never choose λ by hand or by in-sample fit.

Regularization Is Not Causal Inference

Suppose your Lasso model drops “distance to school.” Does that mean distance has no effect on price?
No. It means distance was not useful for *prediction* given the other 24 features.

Regularization Is Not Causal Inference

Suppose your Lasso model drops "distance to school." Does that mean distance has no effect on price? No. It means distance was not useful for *prediction* given the other 24 features.

	Prediction	**Causal Inference**
Goal	Predict y for new observations	Estimate the effect of x on y
Bias	Acceptable (bias-variance tradeoff)	Not acceptable (consistency needed)
Feature selection	Useful (simpler model)	Dangerous (may drop confounders)
Regularization	Designed for this	Not appropriate

Regularization Is Not Causal Inference

Suppose your Lasso model drops "distance to school." Does that mean distance has no effect on price? No. It means distance was not useful for *prediction* given the other 24 features.

	Prediction	**Causal Inference**
Goal	Predict y for new observations	Estimate the effect of x on y
Bias	Acceptable (bias-variance tradeoff)	Not acceptable (consistency needed)
Feature selection	Useful (simpler model)	Dangerous (may drop confounders)
Regularization	Designed for this	Not appropriate

Regularization **deliberately introduces bias** to reduce variance. This helps prediction but means the coefficients are **not consistent estimates** of causal effects.

Regularization Is Not Causal Inference

Suppose your Lasso model drops "distance to school." Does that mean distance has no effect on price? No. It means distance was not useful for *prediction* given the other 24 features.

	Prediction	**Causal Inference**
Goal	Predict y for new observations	Estimate the effect of x on y
Bias	Acceptable (bias-variance tradeoff)	Not acceptable (consistency needed)
Feature selection	Useful (simpler model)	Dangerous (may drop confounders)
Regularization	Designed for this	Not appropriate

Regularization **deliberately introduces bias** to reduce variance. This helps prediction but means the coefficients are **not consistent estimates** of causal effects.

$\Longrightarrow$ Use regularization for prediction. Use IV, FE, or other causal methods for inference. Do not mix them up.

Outline

Summary

We started with 200 houses and 25 features. OLS overfit the training data, chasing noise in 17 irrelevant features and producing predictions that were 20% worse on new houses.

Summary

We started with 200 houses and 25 features. OLS overfit the training data, chasing noise in 17 irrelevant features and producing predictions that were 20% worse on new houses.

1. **Ridge** (ℓ_2 penalty) shrinks all coefficients but sets none to zero. It tamed the variance but kept all 25 features.

Summary

We started with 200 houses and 25 features. OLS overfit the training data, chasing noise in 17 irrelevant features and producing predictions that were 20% worse on new houses.

1. **Ridge** (ℓ_2 penalty) shrinks all coefficients but sets none to zero. It tamed the variance but kept all 25 features.
2. **Lasso** (ℓ_1 penalty) shrinks some coefficients to exactly zero. It identified most of the genuine features and discarded much of the noise, retaining roughly 10–12 features total.

Summary

We started with 200 houses and 25 features. OLS overfit the training data, chasing noise in 17 irrelevant features and producing predictions that were 20% worse on new houses.

1. **Ridge** (ℓ_2 penalty) shrinks all coefficients but sets none to zero. It tamed the variance but kept all 25 features.
2. **Lasso** (ℓ_1 penalty) shrinks some coefficients to exactly zero. It identified most of the genuine features and discarded much of the noise, retaining roughly 10–12 features total.
3. **Elastic Net** combines both penalties. It selected features like Lasso but kept correlated groups together like Ridge.

Summary

We started with 200 houses and 25 features. OLS overfit the training data, chasing noise in 17 irrelevant features and producing predictions that were 20% worse on new houses.

1. **Ridge** (ℓ_2 penalty) shrinks all coefficients but sets none to zero. It tamed the variance but kept all 25 features.
2. **Lasso** (ℓ_1 penalty) shrinks some coefficients to exactly zero. It identified most of the genuine features and discarded much of the noise, retaining roughly 10–12 features total.
3. **Elastic Net** combines both penalties. It selected features like Lasso but kept correlated groups together like Ridge.
4. **Cross-validation** chose λ by simulating out-of-sample prediction. The one-standard-error rule (λ_{1se}) favored a simpler model with nearly identical accuracy.

Summary

We started with 200 houses and 25 features. OLS overfit the training data, chasing noise in 17 irrelevant features and producing predictions that were 20% worse on new houses.

1. **Ridge** (ℓ_2 penalty) shrinks all coefficients but sets none to zero. It tamed the variance but kept all 25 features.
2. **Lasso** (ℓ_1 penalty) shrinks some coefficients to exactly zero. It identified most of the genuine features and discarded much of the noise, retaining roughly 10–12 features total.
3. **Elastic Net** combines both penalties. It selected features like Lasso but kept correlated groups together like Ridge.
4. **Cross-validation** chose λ by simulating out-of-sample prediction. The one-standard-error rule (λ_{1se}) favored a simpler model with nearly identical accuracy.

All three regularized methods cut the test RMSE substantially compared to OLS.

Summary

We started with 200 houses and 25 features. OLS overfit the training data, chasing noise in 17 irrelevant features and producing predictions that were 20% worse on new houses.

1. **Ridge** (ℓ_2 penalty) shrinks all coefficients but sets none to zero. It tamed the variance but kept all 25 features.
2. **Lasso** (ℓ_1 penalty) shrinks some coefficients to exactly zero. It identified most of the genuine features and discarded much of the noise, retaining roughly 10–12 features total.
3. **Elastic Net** combines both penalties. It selected features like Lasso but kept correlated groups together like Ridge.
4. **Cross-validation** chose λ by simulating out-of-sample prediction. The one-standard-error rule (λ_{1se}) favored a simpler model with nearly identical accuracy.

All three regularized methods cut the test RMSE substantially compared to OLS.

$\Longrightarrow$ Regularization is a prediction tool: it trades bias for lower variance. For causal inference, use IV, FE, or RE instead.

Thank you!

jakeanderson@g.ucla.edu