# Background: Why We Need Regularization

When you fit a **Linear Regression model**, it tries to minimize:

But if your model has:

* Many correlated features, or
* Features with large values, or
* Too many parameters compared to data size

Then the model may **overfit** (fit noise instead of true trend).  
To fix this, we add a **penalty term** to the cost function → **Regularization**.

# Ridge Regression (L2 Regularization)

**Definition**

Ridge Regression adds a penalty on the **square of coefficients** (L2 penalty).

Here:

* → regularization strength (λ in some books)
* Larger α ⇒ more penalty ⇒ coefficients shrink toward 0 (but not exactly 0)

**Why Use Ridge**

* To reduce **overfitting**.
* To handle **multicollinearity** (correlated predictors).
* Keeps all features but shrinks their impact.

**How It Works**

Ridge tries to balance:

* Minimizing prediction error (fit)
* Keeping coefficients small (simplicity)

When α = 0 → it becomes normal linear regression.  
When α → ∞ → coefficients approach zero.

**Example Code (Ridge Regression)**

# Ridge Regression Example

from sklearn.linear\_model import Ridge

from sklearn.model\_selection import train\_test\_split

from sklearn.metrics import mean\_squared\_error, r2\_score

from sklearn.datasets import load\_diabetes

import pandas as pd

# Load sample dataset

data = load\_diabetes()

X = pd.DataFrame(data.data, columns=data.feature\_names)

y = data.target

# Split data

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Train Ridge Regression model

ridge = Ridge(alpha=1.0)

ridge.fit(X\_train, y\_train)

# Predictions

y\_pred = ridge.predict(X\_test)

# Evaluation

print("R^2 Score:", r2\_score(y\_test, y\_pred))

print("MSE:", mean\_squared\_error(y\_test, y\_pred))

print("Coefficients:", ridge.coef\_)

**Interpretation**

* **α (alpha)** controls how much regularization to apply.
* Increasing α → smaller coefficients → simpler model → less overfitting.
* ridge.coef\_ shows how much each feature contributes.

# Lasso Regression (L1 Regularization)

**Definition**

Lasso Regression adds a penalty on the **absolute values of coefficients** (L1 penalty):

**Why Use Lasso**

* To **reduce overfitting** *and*
* **Select important features** automatically (sets some coefficients exactly to 0).

So, Lasso can perform **feature selection**.

**How It Works**

Because of the absolute value term, Lasso can drive some coefficients **exactly to zero**.  
This means:

* Irrelevant features are removed.
* Model becomes simpler and more interpretable.

**Example Code (Lasso Regression)**

# Lasso Regression Example

from sklearn.linear\_model import Lasso

from sklearn.model\_selection import train\_test\_split

from sklearn.metrics import mean\_squared\_error, r2\_score

from sklearn.datasets import load\_diabetes

import pandas as pd

# Load sample dataset

data = load\_diabetes()

X = pd.DataFrame(data.data, columns=data.feature\_names)

y = data.target

# Split data

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Train Lasso Regression model

lasso = Lasso(alpha=0.1)

lasso.fit(X\_train, y\_train)

# Predictions

y\_pred = lasso.predict(X\_test)

# Evaluation

print("R^2 Score:", r2\_score(y\_test, y\_pred))

print("MSE:", mean\_squared\_error(y\_test, y\_pred))

print("Coefficients:", lasso.coef\_)

**Interpretation**

* **α (alpha)** controls how many coefficients are zeroed.
* Small α → behaves like linear regression.
* Large α → more coefficients become 0.
* Great for identifying the most important features.

# Ridge vs Lasso (Comparison)

| **Feature** | **Ridge (L2)** | **Lasso (L1)** |
| --- | --- | --- |
| Penalty Type | Sum of squares (β²) | Sum of absolute values ( |
| Shrinks Coefficients | Yes | Yes |
| Sets Coefficients to Zero | ❌ No | ✅ Yes |
| Feature Selection | ❌ No | ✅ Yes |
| When to Use | Many small/medium effects | Only few important features |
| Stability | More stable | May be unstable if features are highly correlated |

**When to Use Which**

* Use **Ridge**:  
  When you have many correlated variables, and you don’t want to remove any feature.
* Use **Lasso**:  
  When you suspect only a few features are truly important, and you want automatic feature selection.