Regularization

Regularization

- Regularization seeks to solve a few common model issues by:
    - Minimizing model complexity
    - Penalizing the loss function
    - Reducing model overfitting (add more bias to reduce model variance)

Regularization

- In general, we can think of regularization as a way to reduce model overfitting and variance.
  - Requires some additional bias
  - Requires a search for optimal penalty hyperparameter.

Regularization

- Three main types of Regularization:
  - L1 Regularization
    - LASSO Regression
  - L2 Regularization
    - Ridge Regression
  - Combining L1 and L2
    - Elastic Net

Regularization

- L1 regularization adds a penalty equal to the **absolute value** of the magnitude of coefficients.
    - Limits the size of the coefficients.
    - Can yield sparse models where some coefficients can become zero.

Regularization

- L1 regularization adds a penalty equal to the **absolute value** of the magnitude of coefficients.

$$\sum_{i=1}^{n}\left(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij}\right)^2 + \lambda\sum_{j=1}^{p}|\beta_j| = \text{RSS} + \boxed{\lambda\sum_{j=1}^{p}|\beta_j|}$$

Regularization

- L2 regularization adds a penalty equal to the **square** of the magnitude of coefficients.
    - All coefficients are shrunk by the same factor.
    - Does not necessarily eliminate coefficients.

Regularization

- L2 regularization adds a penalty equal to the **square** of the magnitude of coefficients.

$$\sum_{i=1}^{n}\left(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij}\right)^2 + \lambda\sum_{j=1}^{p}\beta_j^2 = \text{RSS} + \boxed{\lambda\sum_{j=1}^{p}\beta_j^2}$$

Regularization

- Elastic net combines L1 and L2 with the addition of an alpha parameter deciding the ratio between them:

$$\frac{\sum_{i=1}^{n}(y_i - x_i^J \hat{\beta})^2}{2n} + \lambda \left(\frac{1-\alpha}{2} \sum_{j=1}^{m} \hat{\beta}_j^2 + \alpha \sum_{j=1}^{m} |\hat{\beta}_j| \right)$$

Regularization

* These regularization methods do have a cost:
    * Introduce an additional hyperparameter that needs to be tuned.
    * A multiplier to the penalty to decide the “strength” of the penalty.

Regularization

* Later on, we will actually cover L2 regularization (Ridge Regression) first, due to the intuition behind the squared term being easier to understand.

Regularization

- Before we dive straight into coding regularization with Scikit-Learn, we need to discuss a few more relevant topics:
  - Feature Scaling
  - Cross Validation

Feature Scaling

Feature Scaling

- Feature scaling provides many benefits to our machine learning process!
- Some machine learning models that rely on distance metrics (e.g. KNN) **require** scaling to perform well.
- Let's discuss the main ideas behind feature scaling...

Feature Scaling

- Feature scaling improves the convergence of steepest descent algorithms, which do not possess the property of scale invariance.
- If features are on different scales, certain weights may update faster than others since the feature values $\mathbf{x_j}$ play a role in the weight updates.

Feature Scaling

- Critical benefit of feature scaling related to gradient descent.
- There are some ML Algos where scaling won’t have an effect (e.g. CART based methods).

Feature Scaling

- Scaling the features so that their respective ranges are uniform is important in comparing measurements that have different units.
- Allows us directly compare model coefficients to each other.

Feature Scaling

- Feature scaling caveats:
  - Must always scale new unseen data before feeding to model.
  - Effects direct interpretability of feature coefficients
    - Easier to compare coefficients to one another, harder to relate back to original unscaled feature.

Feature Scaling

* Feature scaling benefits:
    * Can lead to great increases in performance.
    * Absolutely necessary for some models.
    * Virtually no “real” downside to scaling features.

Feature Scaling

- Two main ways to scale features:
  - Standardization
    - Rescales data to have a mean (**μ**) of 0 and standard deviation (**σ**) of 1.
  - Normalization
    - Rescales all data values to be between 0-1.

Feature Scaling

- Standardization:
  - Rescales data to have a mean (**μ**) of 0 and standard deviation (**σ**) of 1 (unit variance).

$$X_{changed} = \frac{X - \mu}{\sigma}$$

Feature Scaling

- Standardization:
  - Namesake can be confusing since this is also referred to as “Z-score normalization”.

$$X_{changed} = \frac{X - \mu}{\sigma}$$

Feature Scaling

- Normalization:
  - Scales all data values to be between 0 and 1.

$$X_{changed} = \frac{X - X_{min}}{X_{max} - X_{min}}$$

Feature Scaling

- Normalization:
  - Simple and easy to understand.

$$X_{changed} = \frac{X - X_{min}}{X_{max} - X_{min}}$$

Feature Scaling

- There are many more methods of scaling features and Scikit-Learn provides easy to use classes that “fit” and “transform” feature data for scaling.
- Let’s quickly discuss the fit and transform calls in more detail when it comes to scaling.

Feature Scaling

- A .fit() method call simply calculates the necessary statistics (Xmin,Xmax,mean, standard deviation).
- A .transform() call actually scales data and returns the new scaled version of data.
- Previously saw a similar process for polynomial feature conversion.

Feature Scaling

- Very important consideration for fit and transform:
  - We only **fit** to training data.
  - Calculating statistical information should only come from training data.
  - Don’t want to assume prior knowledge of the test set!

Feature Scaling

- Using the full data set would cause **data leakage**:
  - Calculating statistics from full data leads to some information of the test set leaking into the training process upon transform() conversion.

Feature Scaling

- Feature scaling process:
  - Perform train test split
  - Fit to training feature data
  - Transform training feature data
  - Transform test feature data

Feature Scaling

- Do we need to scale the label?
  - In general it is not necessary nor advised.
  - Normalising the output distribution is altering the definition of the target.
  - Predicting a distribution that doesn't mirror your real-world target.

Feature Scaling

- Do we need to scale the label?
    - Can negatively impact stochastic gradient descent.
- **stats.stackexchange.com/questions/111467**

Feature Scaling

- Now that we understand the benefits of feature scaling, let’s move on to understanding the benefits of cross-validation!

Cross Validation

Cross Validation

- Cross validation is a more advanced set of methods for splitting data into training and testing sets.
- Cross Validation Relevant Reading:
  - Section 5.1 of ISLR

Cross Validation

- We understand the intuition behind performing a train test split, we want to fairly evaluate our model's performance on unseen data.
- Unfortunately this means we are not able to tune hyperparameters to the **entire** dataset.

Cross Validation

- Is there a way we can achieve the following:
  - Train on all the data
  - Evaluate on all the data
- While it sounds impossible, we can achieve this with cross validation!
- Let’s have an overview of the concept…

Cross Validation

- Imagine our data set:

Area m^2	Bedrooms	Bathrooms	Price
200	3	2	$500,000
190	2	1	$450,000
230	3	3	$650,000
180	1	1	$400,000
210	2	2	$550,000

X (Area m^2, Bedrooms, Bathrooms) y (Price)

Cross Validation

- Let’s convert this data into colored blocks for cross-validation

X			y
Area m^2	Bedrooms	Bathrooms	Price
200	3	2	$500,000
190	2	1	$450,000
230	3	3	$650,000
180	1	1	$400,000
210	2	2	$550,000

Cross Validation

- Convert to generalized form

	X		**y**
$\mathbf{x_1}$	$\mathbf{x_2}$	$\mathbf{x_3}$	$\mathbf{y}$
$x^1{}_1$	$x^1{}_1$	$x^1{}_1$	y_1
$x^2{}_1$	$x^2{}_1$	$x^2{}_1$	y_2
$x^3{}_1$	$x^3{}_1$	$x^3{}_1$	y_3
$x^4{}_1$	$x^4{}_1$	$x^4{}_1$	y_4
$x^5{}_1$	$x^5{}_1$	$x^5{}_1$	y_5

Cross Validation

- Color based off train vs. test set.

	X		**y**
$\mathbf{x_1}$	$\mathbf{x_2}$	$\mathbf{x_3}$	$\mathbf{y}$
$x^1{}_1$	$x^1{}_1$	$x^1{}_1$	y_1
$x^2{}_1$	$x^2{}_1$	$x^2{}_1$	y_2
$x^3{}_1$	$x^3{}_1$	$x^3{}_1$	y_3
$x^4{}_1$	$x^4{}_1$	$x^4{}_1$	y_4
$x^5{}_1$	$x^5{}_1$	$x^5{}_1$	y_5

Cross Validation

- Color based off train vs. test set.

	X			y
	x_1	x_2	x_3	y
TRAIN	x^1_1	x^1_1	x^1_1	y_1
	x^2_1	x^2_1	x^2_1	y_2
	x^3_1	x^3_1	x^3_1	y_3
TEST	x^4_1	x^4_1	x^4_1	y_4
	x^5_1	x^5_1	x^5_1	y_5

Cross Validation

- Color based off train vs. test set.

	X			y
	x_1	x_2	x_3	y
TRAIN	x^1_1	x^1_1	x^1_1	y_1
TRAIN	x^2_1	x^2_1	x^2_1	y_2
TRAIN	x^3_1	x^3_1	x^3_1	y_3
TEST	x^4_1	x^4_1	x^4_1	y_4
TEST	x^5_1	x^5_1	x^5_1	y_5

Cross Validation

- For now just consider training vs testing:

	X			y
	x_1	x_2	x_3	y
TRAIN	x^1_1	x^1_1	x^1_1	y_1
TRAIN	x^2_1	x^2_1	x^2_1	y_2
TRAIN	x^3_1	x^3_1	x^3_1	y_3
TEST	x^4_1	x^4_1	x^4_1	y_4
TEST	x^5_1	x^5_1	x^5_1	y_5

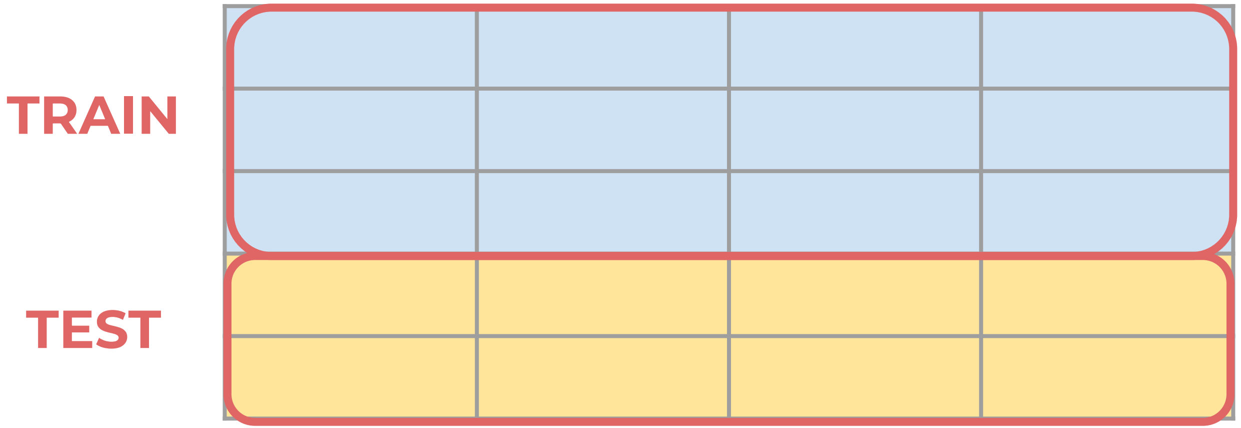
Cross Validation

- For now just consider training vs testing:

TRAIN	$x^1{}_1$	$x^1{}_1$	$x^1{}_1$	y_1
	$x^2{}_1$	$x^2{}_1$	$x^2{}_1$	y_2
	$x^3{}_1$	$x^3{}_1$	$x^3{}_1$	y_3
TEST	$x^4{}_1$	$x^4{}_1$	$x^4{}_1$	y_4
	$x^5{}_1$	$x^5{}_1$	$x^5{}_1$	y_5

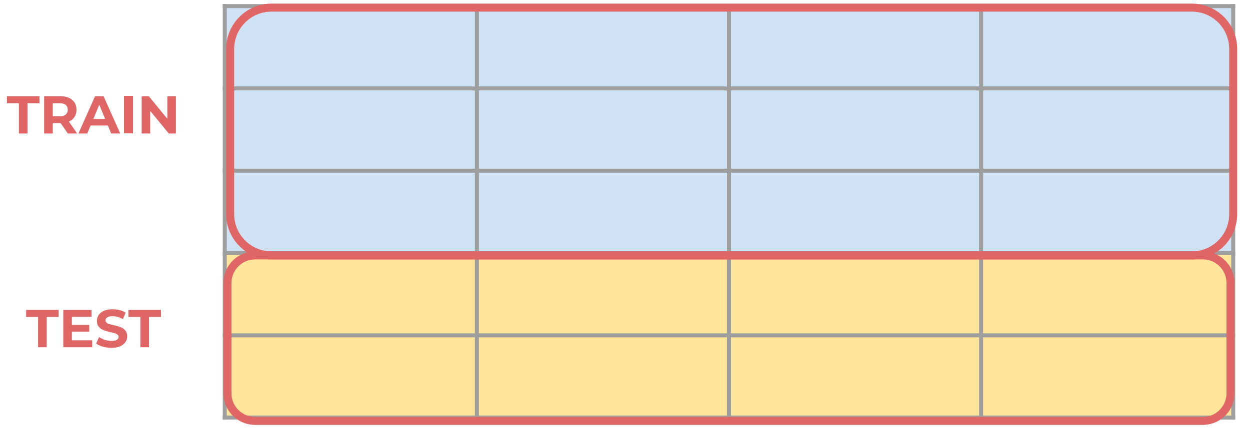
Cross Validation

- Now we have all data, colored by training set versus test set.



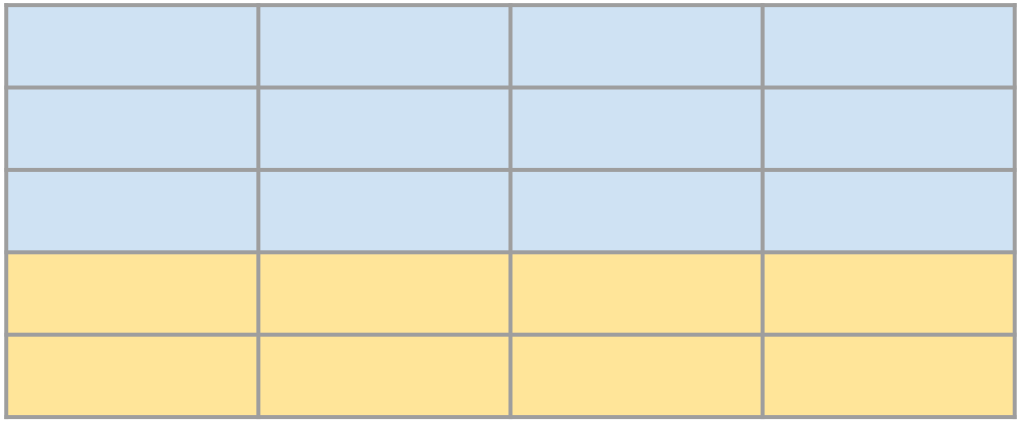
Cross Validation

- Rotate and resize:



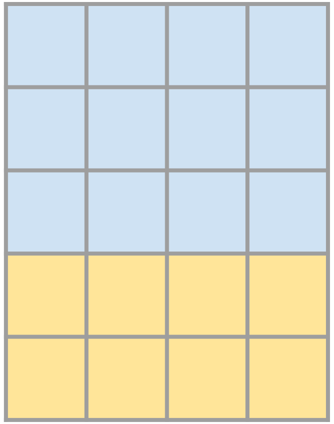
Cross Validation

- Rotate and resize:



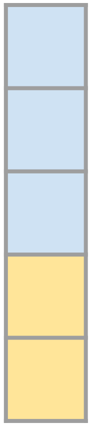
Cross Validation

- Rotate and resize:



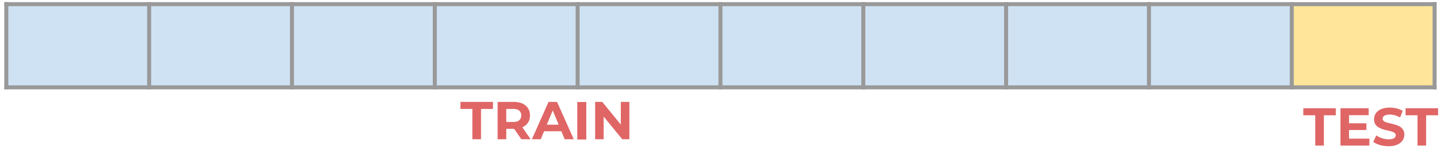
Cross Validation

- Rotate and resize:



Cross Validation

- Now we can represent full data and splits:



Cross Validation

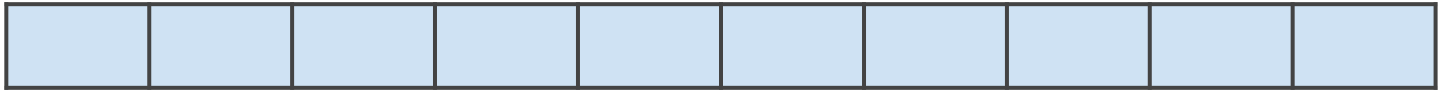
- Let’s start with the entire original data:

Cross Validation

- How does cross validation work?

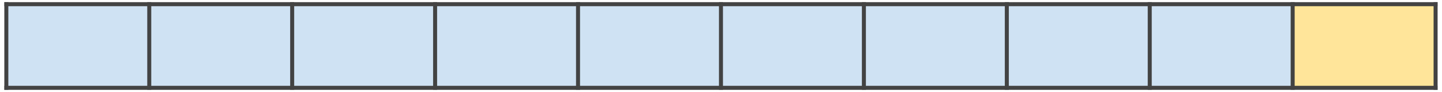
Cross Validation

- Split data into K equal parts:



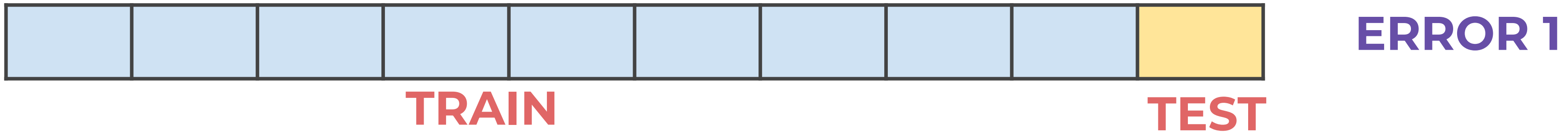
Cross Validation

- 1/K left as test set



Cross Validation

- Train model and get error metric for split:



Cross Validation

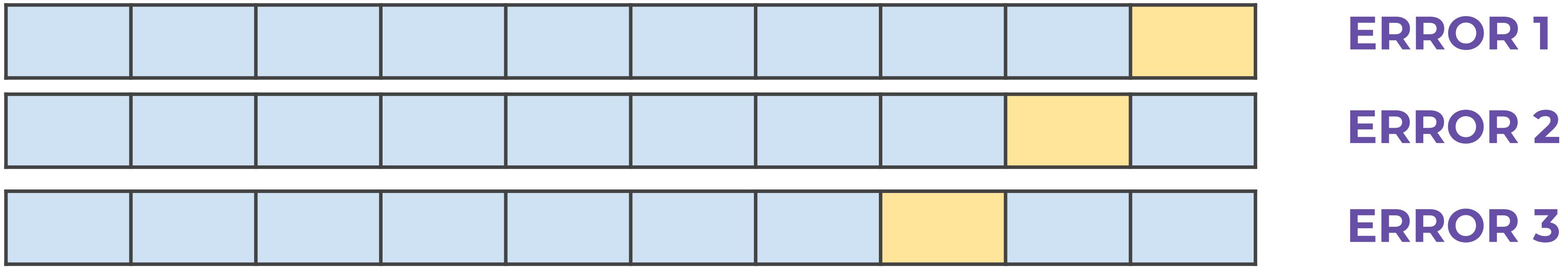
- Repeat for another 1/K split

ERROR 1

ERROR 2

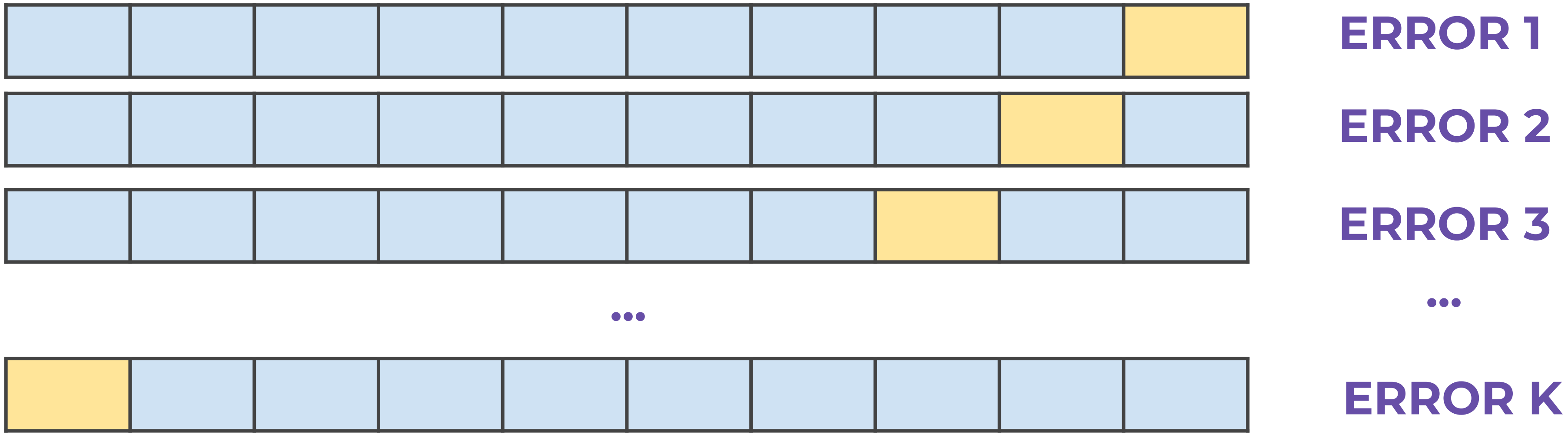
Cross Validation

- Keep repeating for all possible splits



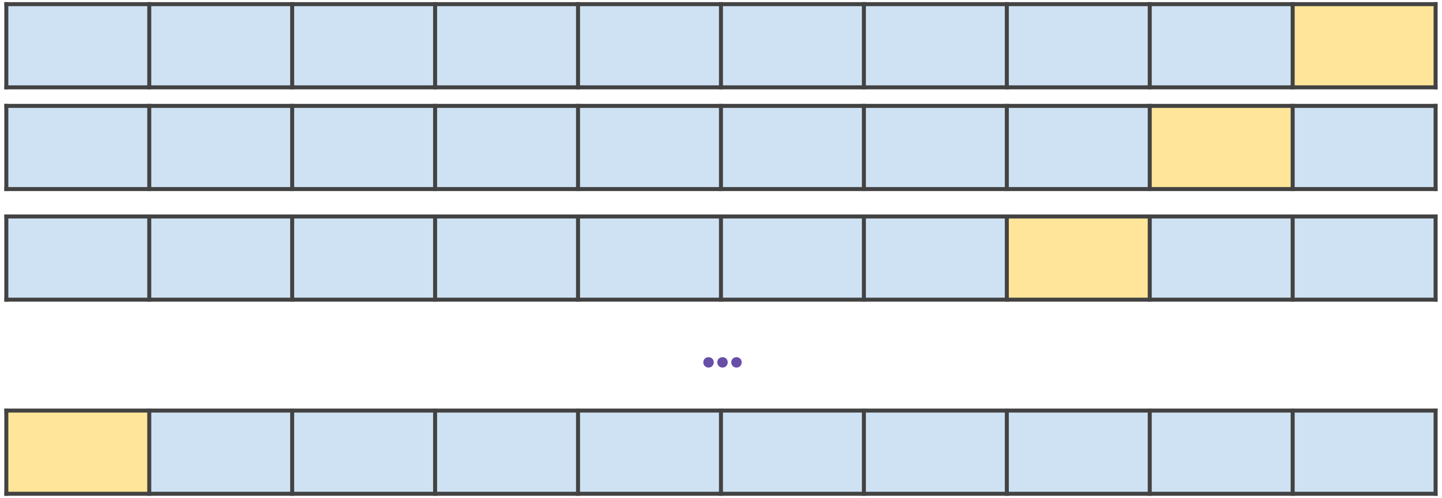
Cross Validation

- Keep repeating for all possible splits



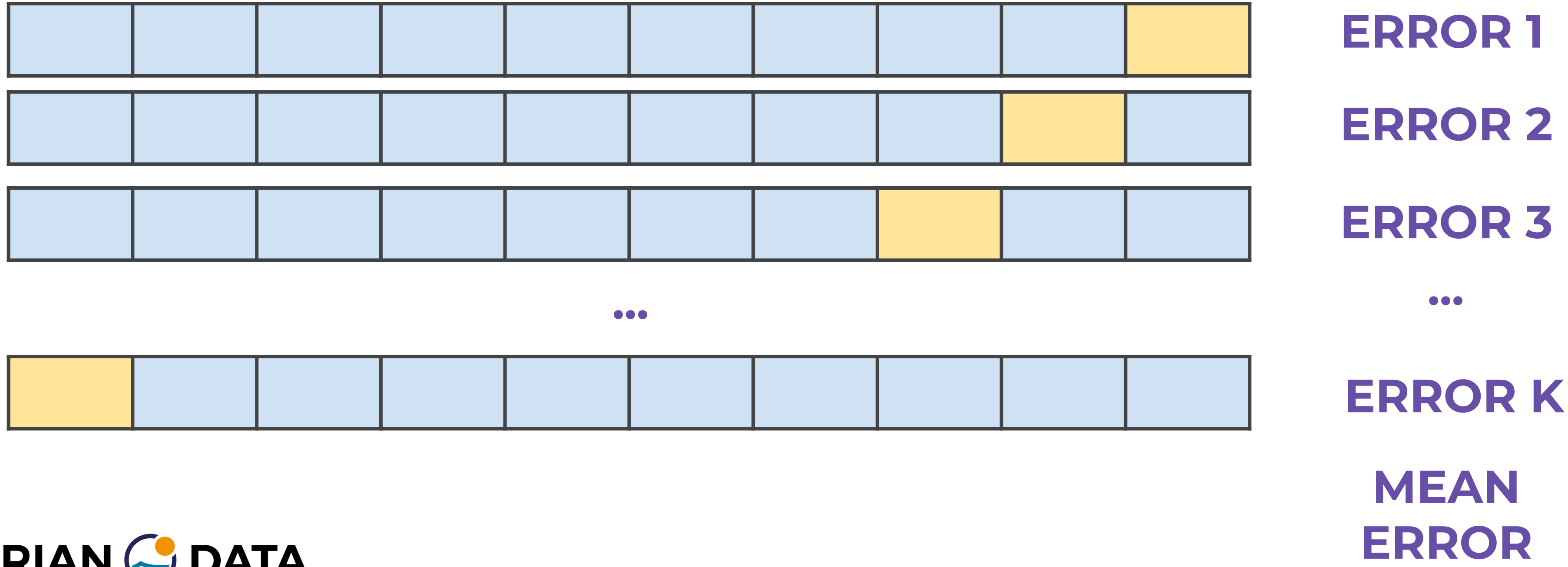
Cross Validation

- Get average error



Cross Validation

- Average error is the expected performance



Cross Validation

- We were able to train on all data **and** evaluate on all data!
- We get a better sense of true performance across multiple potential splits.
- What is the cost of this?
  - We have to repeat computations K number of times!

Cross Validation

- This is known as K-fold cross-validation.
- Common choice for K is 10 so each test set is 10% of your total data.
- Largest K possible would be K equal to the number of number of rows.
  - This is known as **leave one out** cross validation.
  - Computationally expensive!

Cross Validation

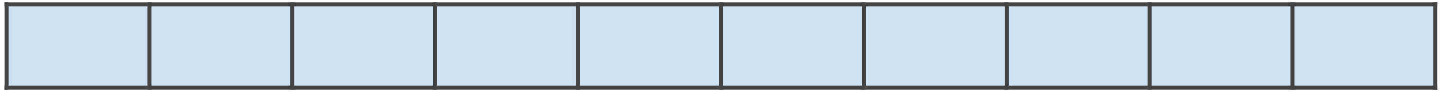
- One consideration to note with K-fold cross validation and a standard train test split is fairly tuning hyperparameters.
- If we tune hyperparameters to test data performance, are we ever fairly getting performance metrics?

Cross Validation

- How can we understand how the model behaves for data that is has not seen **and** not been influenced by for hyperparameter tuning?
- For this we can use a **hold out** test set.
- Let's explore what this looks like...

Cross Validation

- Start with entire data set:



Cross Validation

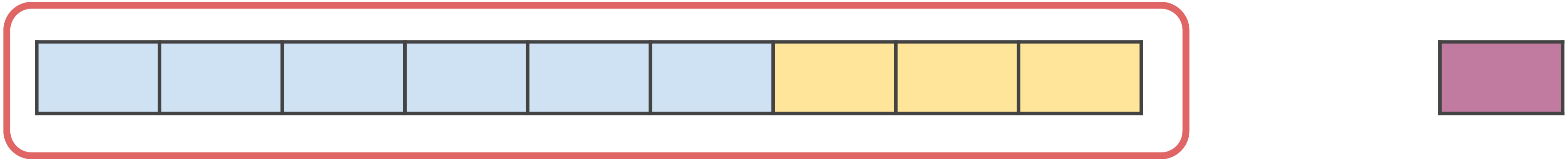
- Remove a hold out test set

Cross Validation

- Perform “classic” train test split:

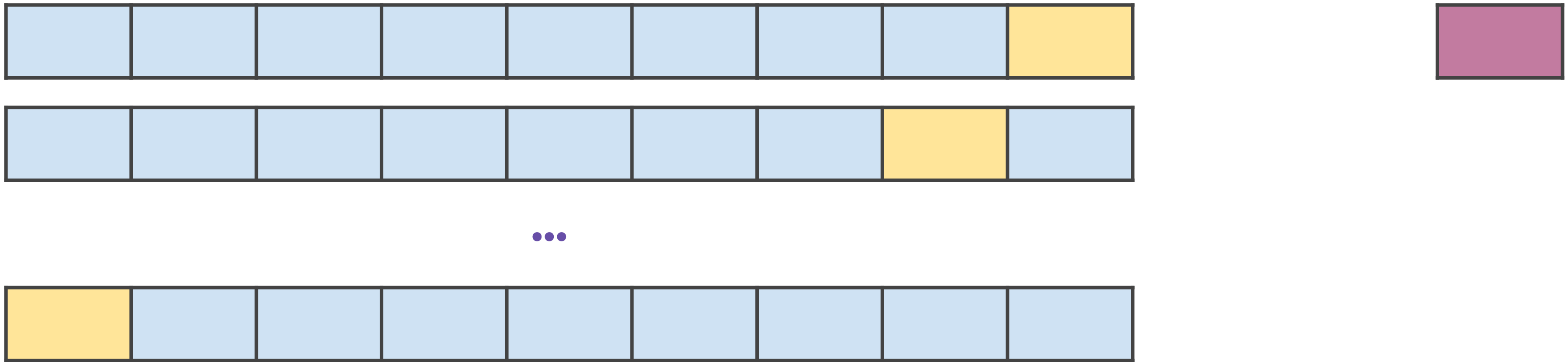
Cross Validation

* Train and tune on this data:



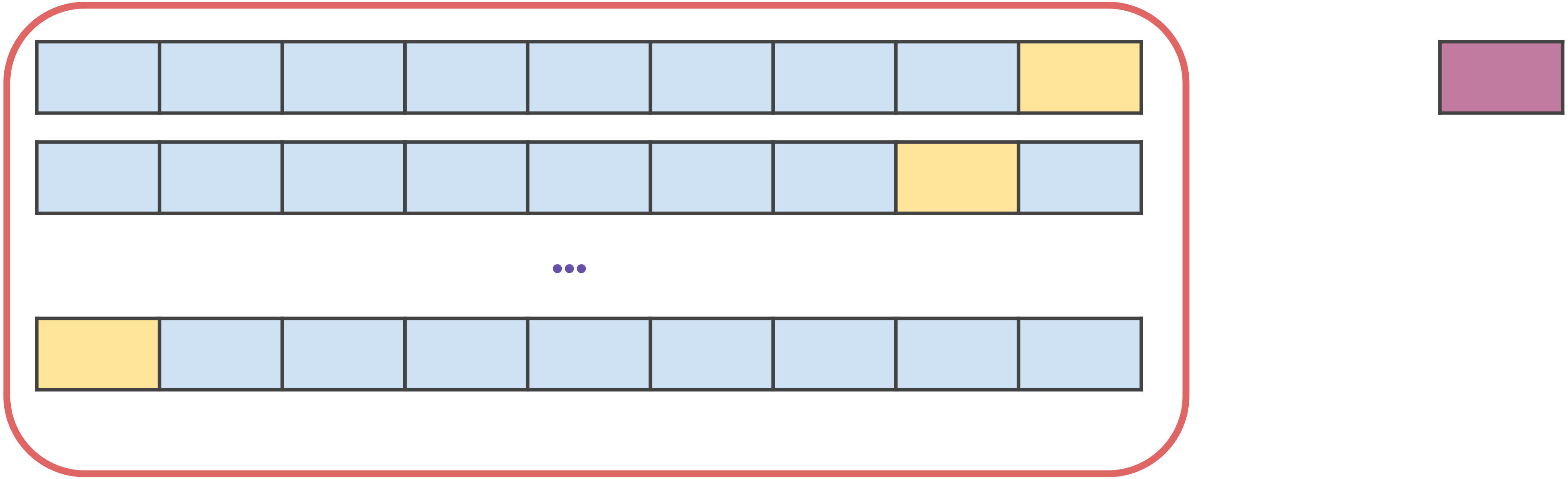
Cross Validation

- Or K-Fold cross validation



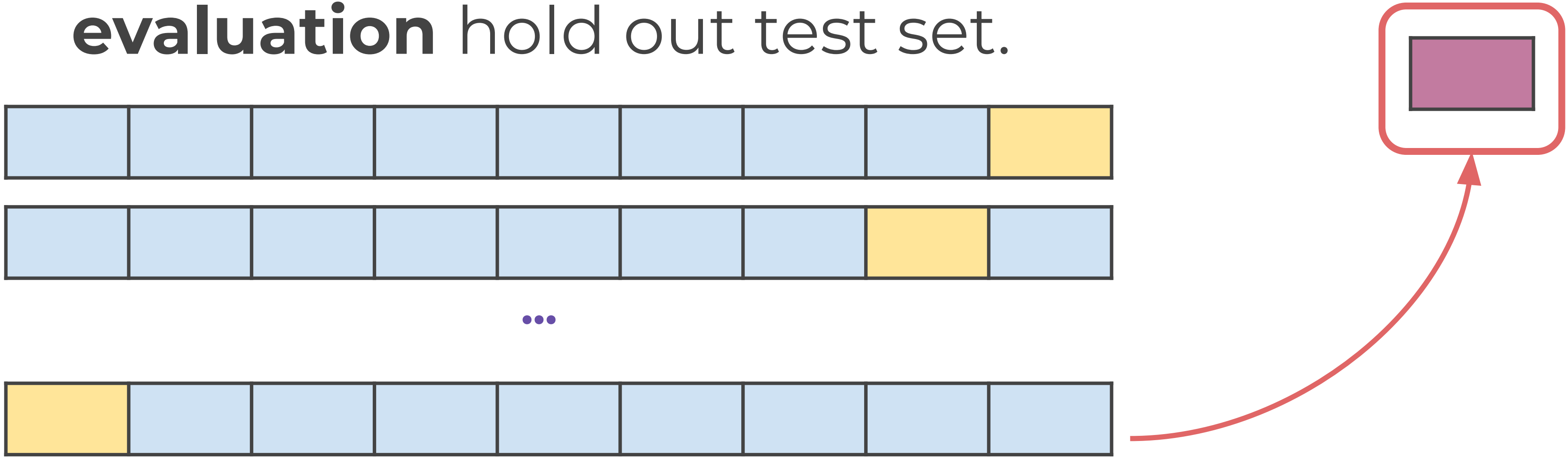
Cross Validation

- Train **and** tune on this data:



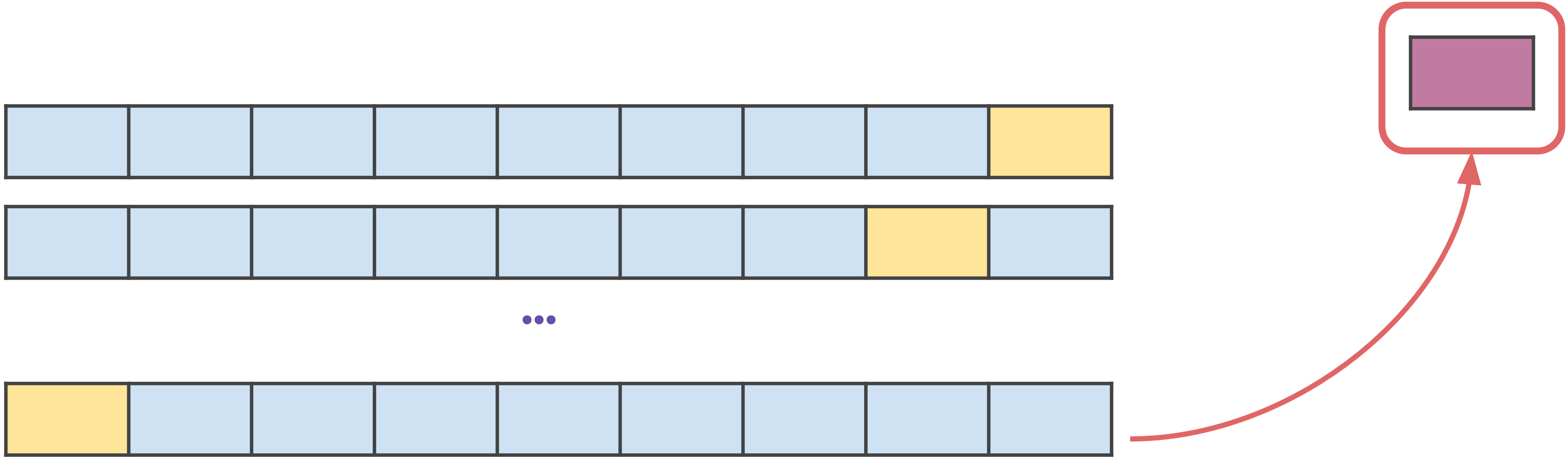
Cross Validation

- **After** training and tuning perform **final evaluation** hold out test set.



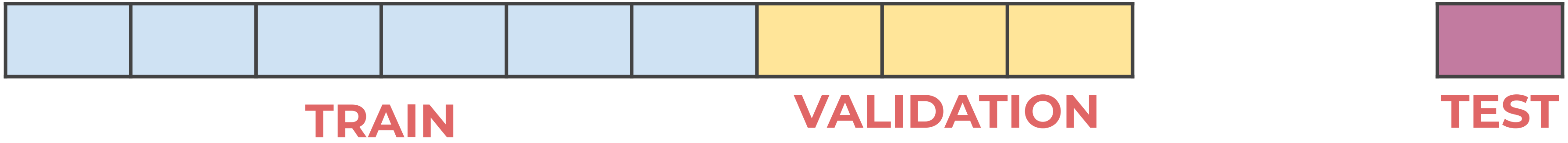
Cross Validation

- Can **not** tune after this **final** test evaluation!



Cross Validation

- Train | Validation | Test Split



- Allows us to get a true final performance metric to report.
- No editing model after this!

Cross Validation

- All these approaches are valid, each situation is unique!
- Keep in mind:
    - Previous modeling work
    - Reporting requirements
    - Fairness of evaluation
    - Context of data and model

Cross Validation

- Many regularization methods have tunable parameters we can adjust based on cross-validation techniques.
- For simplicity, there are times in the course we will opt for a simple two part train test split.

Regularization for Linear Regression

Data Set Up

Ridge Regression

Theory and Intuition

Ridge Regression

- Ridge Regression is a regularization technique that works by helping reduce the potential for overfitting to the training data.
- It does this by adding in a penalty term to the error that is based on the squared value of the coefficients.

Ridge Regression

- Ridge Regression is a regularization method for Linear Regression.
- Relevant Reading in ISLR:
    - Section 6.2.1
- Let’s explore the main concepts behind how Ridge Regression works...

Ridge Regression

- Recall the general formula for the regression line:

$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x_1 + \hat{\beta}_2 x_2 + \cdots + \hat{\beta}_p x_p$$

Ridge Regression

- These Beta coefficients were solved by minimizing the residual sum of squares (RSS).

$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x_1 + \hat{\beta}_2 x_2 + \cdots + \hat{\beta}_p x_p$$

Ridge Regression

- These Beta coefficients were solved by minimizing the residual sum of squares (RSS).

$$\mathrm{RSS} = \sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

Ridge Regression

- We could substitute our regression equation for **ŷ:**

$$\mathrm{RSS} = \sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

Ridge Regression

- We could substitute our regression equation for **ŷ:**

$$
\begin{aligned}
\text{RSS} &= \sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \\
&= \sum_{i=1}^{n}(y_i - \hat{\beta}_0 - \hat{\beta}_1 x_{i1} - \hat{\beta}_2 x_{i2} - \cdots - \hat{\beta}_p x_{ip})^2
\end{aligned}
$$

Ridge Regression

- We can then summarize RSS as:

$$\text{RSS} = \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij} \right)^2$$

Ridge Regression

- The goal of Ridge Regression is to help prevent overfitting by adding an additional penalty term.

$$\mathrm{RSS} = \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij} \right)^2$$

Ridge Regression

- Ridge Regression adds a **shrinkage penalty**:

$$\text{Error} = \sum_{i=1}^{n}\left(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij}\right)^2 \boxed{+\lambda\sum_{j=1}^{p}\beta_j^2}$$

Ridge Regression

- Ridge Regression seeks to minimize this entire error term **RSS + Penalty**.

$$\text{Error} = \sum_{i=1}^{n}\left(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij}\right)^2 \boxed{+ \lambda \sum_{j=1}^{p}\beta_j^2}$$

Ridge Regression

- **Shrinkage penalty** based off the squared coefficient:

$$\text{Error} = \sum_{i=1}^{n}\left(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij}\right)^2 + \lambda\sum_{j=1}^{p}\boxed{\beta_j^2}$$

Ridge Regression

- **Shrinkage penalty** has a **tunable lambda parameter!**

$$\text{Error} = \sum_{i=1}^{n}\left(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij}\right)^2 + \boxed{\lambda}\sum_{j=1}^{p}\beta_j^2$$

Ridge Regression

- Lambda determines how severe the penalty is.

$$\text{Error} = \sum_{i=1}^{n}\left(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij}\right)^2 + \boxed{\lambda}\sum_{j=1}^{p}\beta_j^2$$

Ridge Regression

- In theory it can be any value from 0 to positive infinity.

$$\text{Error} = \sum_{i=1}^{n}\left(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij}\right)^2 + \boxed{\lambda}\sum_{j=1}^{p}\beta_j^2$$

Ridge Regression

- If it is zero, then it is simply back to RSS.

$$\text{Error} = \sum_{i=1}^{n}\left(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij}\right)^2 + \boxed{\lambda}\sum_{j=1}^{p}\beta_j^2$$

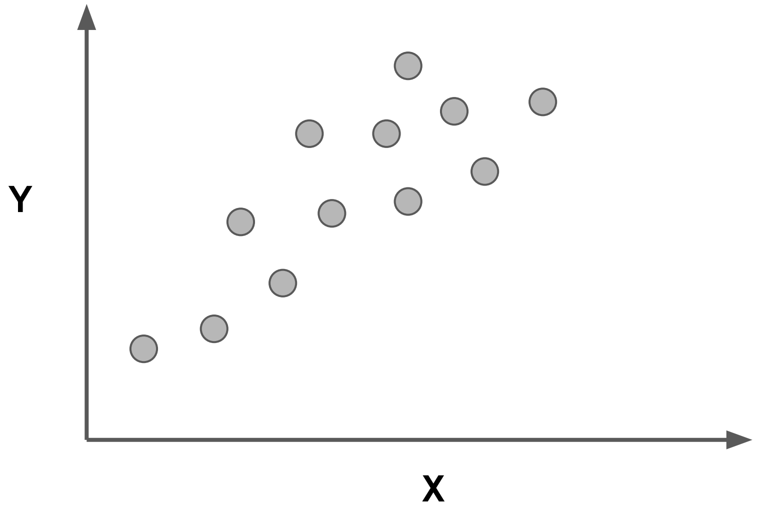
Ridge Regression

- Let's explore a simple thought experiment to get an intuition behind Ridge Regression...

$$\text{Error} = \sum_{i=1}^{n}\left(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij}\right)^2 + \lambda\sum_{j=1}^{p}\beta_j^2$$

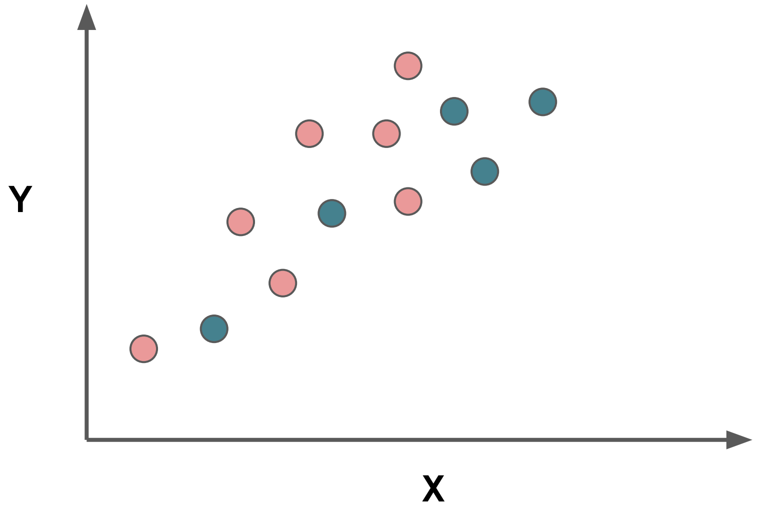
Ridge Regression

* Imagine the following data set.



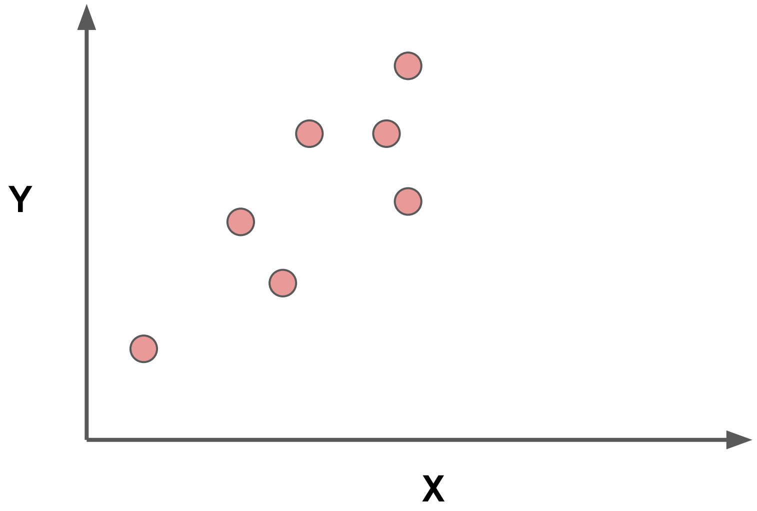
Ridge Regression

- We can split it into a training set and test set:



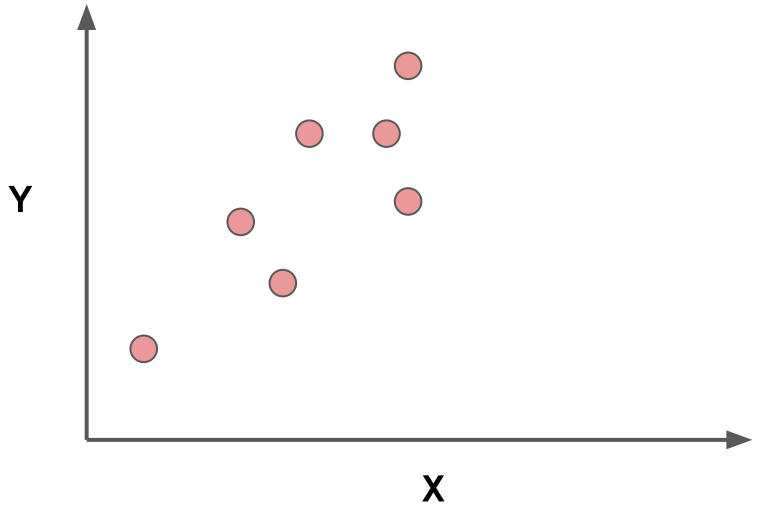
Ridge Regression

- Now we can fit on the training data to produce the line: $\hat{y} = \beta_1 x + \beta_0$



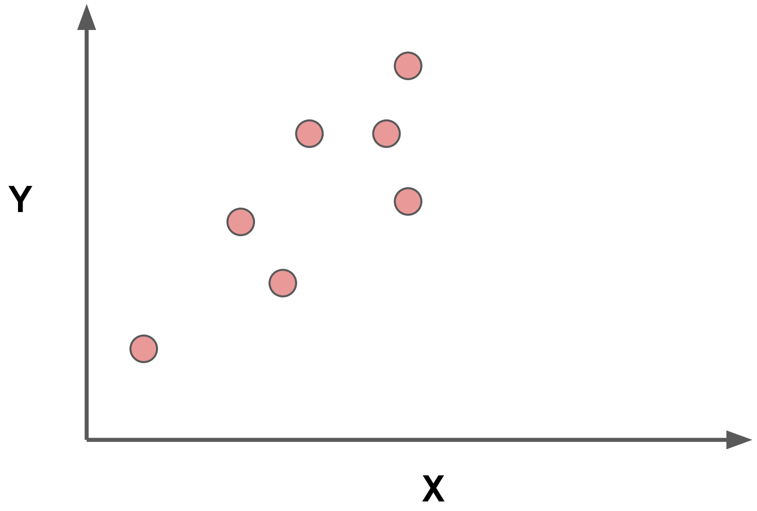
Ridge Regression

- Regardless of RSS or Ridge error, we're still trying to create a line: $\hat{y} = \beta_1 x + \beta_0$



Ridge Regression

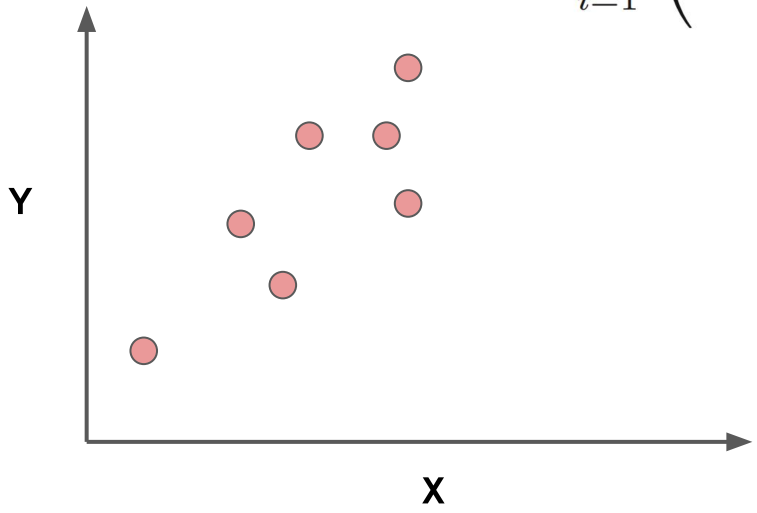
- The only difference would be the coefficients found.



Ridge Regression

- First let's fit using only RSS...

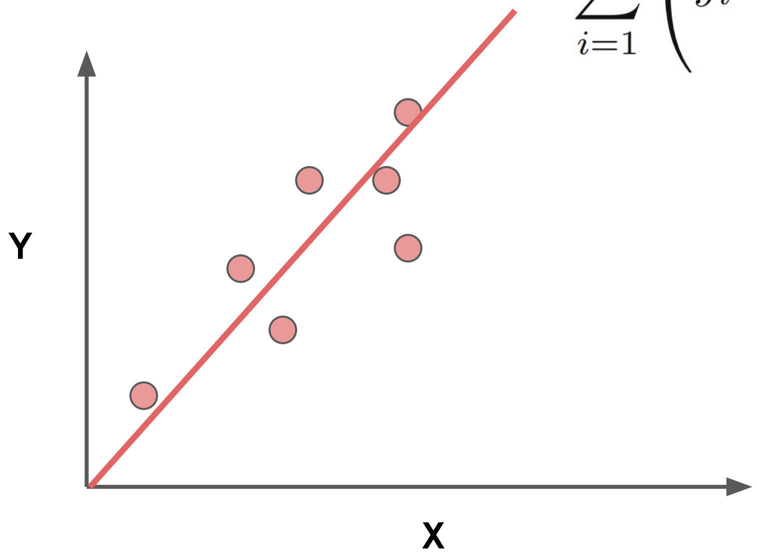
$$\sum_{i=1}^{n}\left(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij}\right)^2$$



Ridge Regression

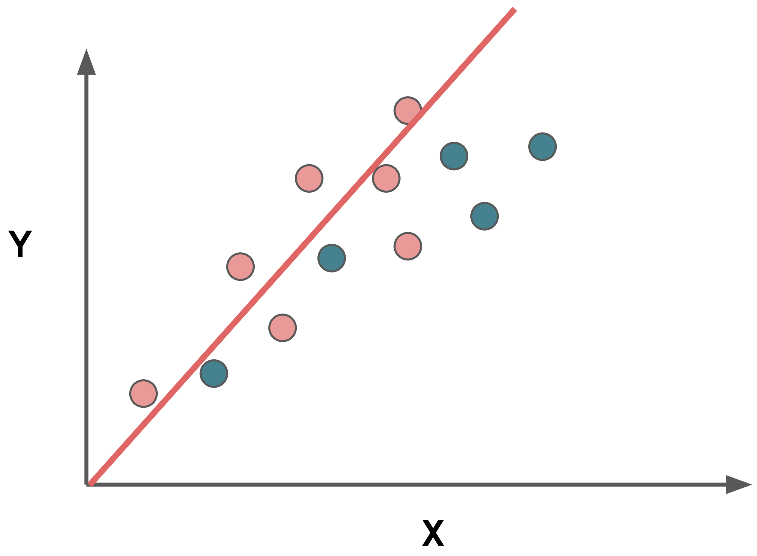
- Our fitted $\hat{y} = \beta_1 x + \beta_0$

$$\sum_{i=1}^{n}\left(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij}\right)^2$$



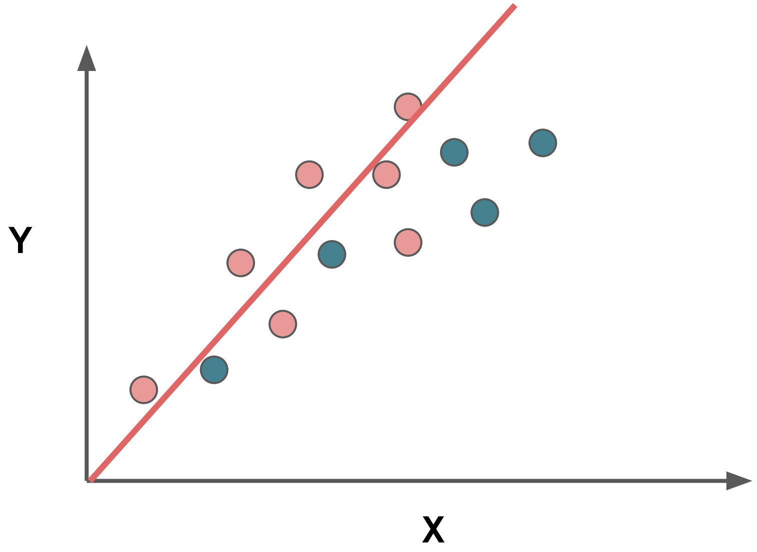
Ridge Regression

- Appears to have over fit to training data.



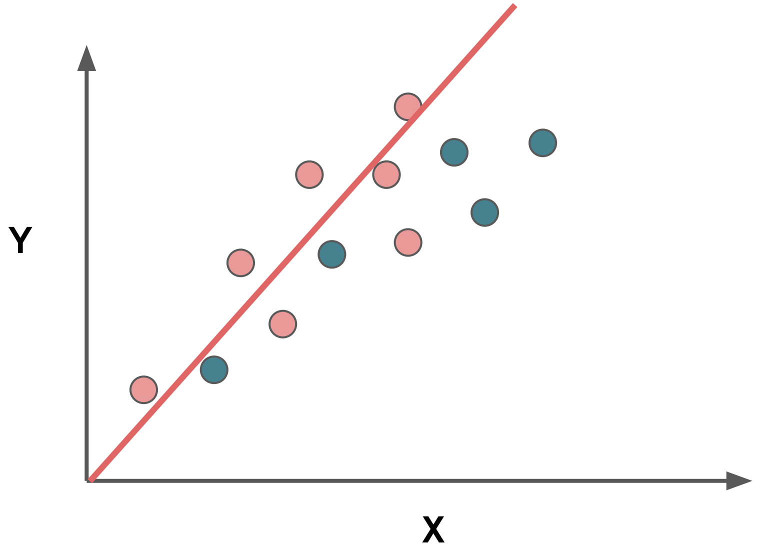
Ridge Regression

- This means we have high **variance.**



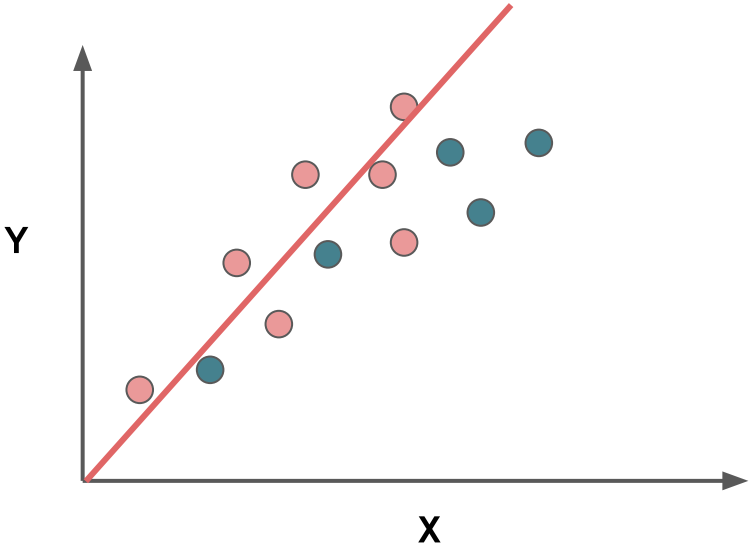
Ridge Regression

- We know there is a **bias-variance** trade-off.



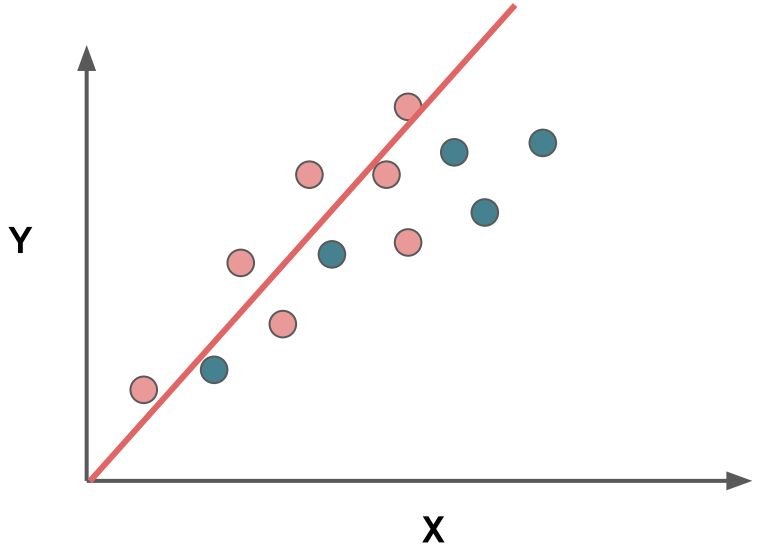
Ridge Regression

- But could we introduce a little more **bias** to significantly **reduce** variance?



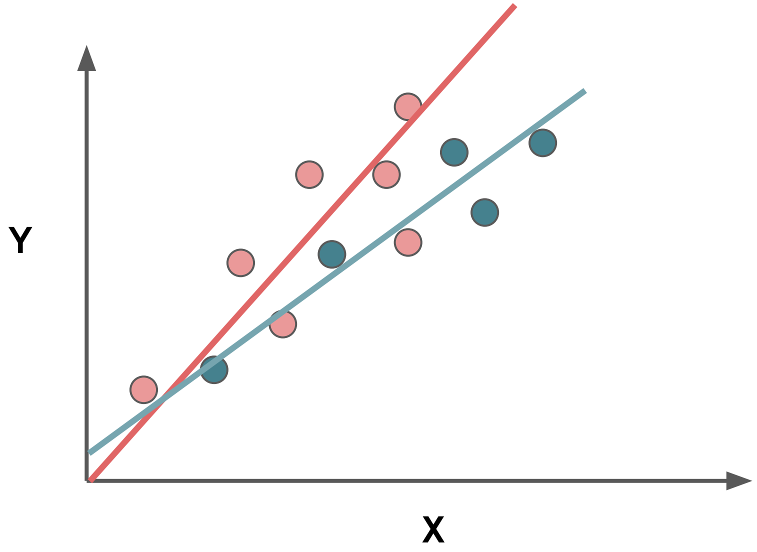
Ridge Regression

- Would adding the penalty term help generalize with more **bias**?



Ridge Regression

- Adding bias can help generalize $\hat{y} = \beta_1 x + \beta_0$

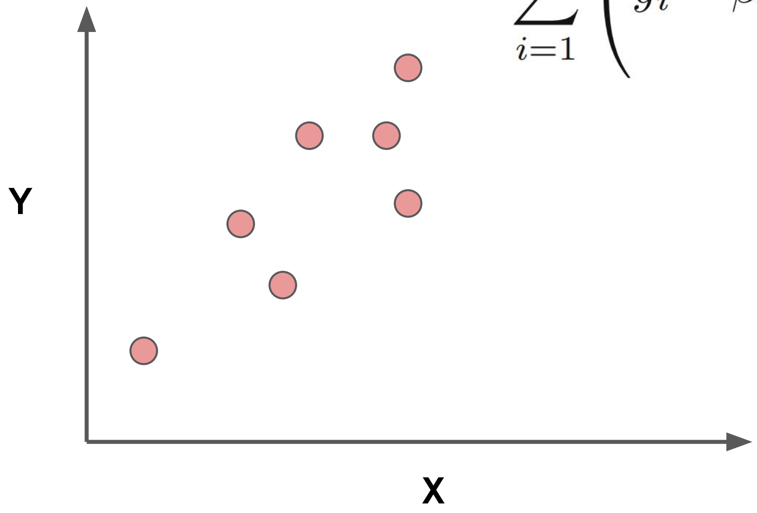


PIERIAN DATA

Ridge Regression

- Let's imagine trying to reduce the Ridge Regression error term:

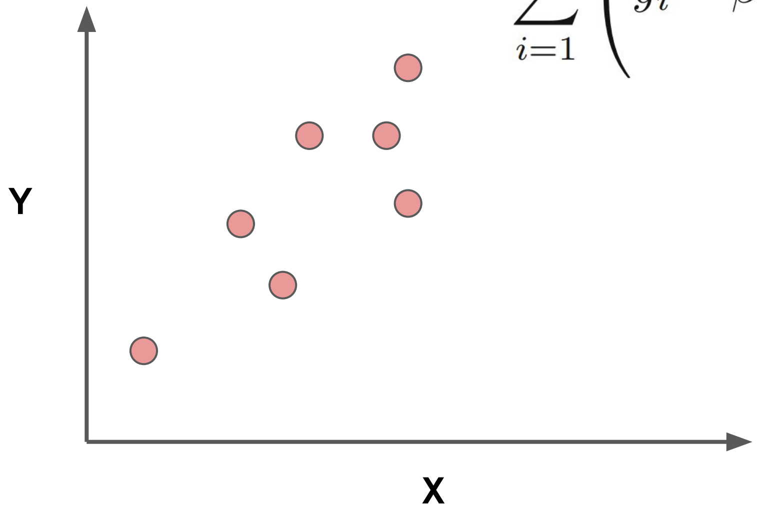
$$\sum_{i=1}^{n}\left(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij}\right)^2 + \lambda\sum_{j=1}^{p}\beta_j^2$$



Ridge Regression

- There is **λ** and the squared slope coefficient.

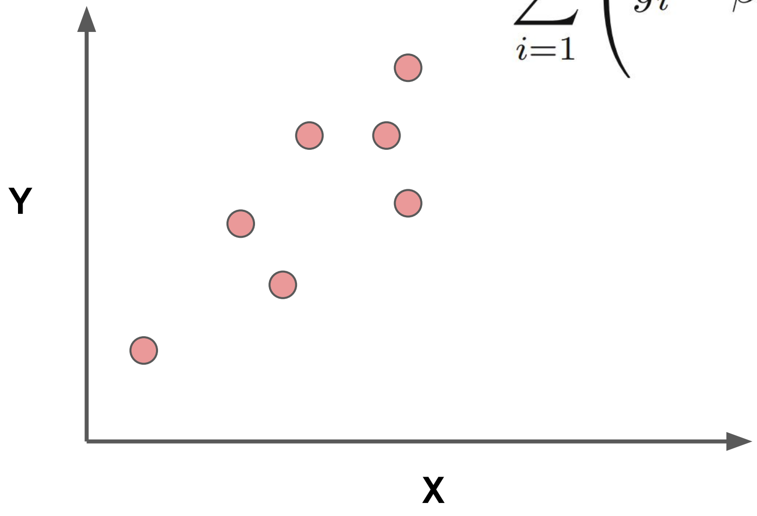
$$\sum_{i=1}^{n}\left(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij}\right)^2 + \boxed{\lambda\sum_{j=1}^{p}\beta_j^2}$$



Ridge Regression

- In the case of $\hat{y} = \beta_1 x + \beta_0$

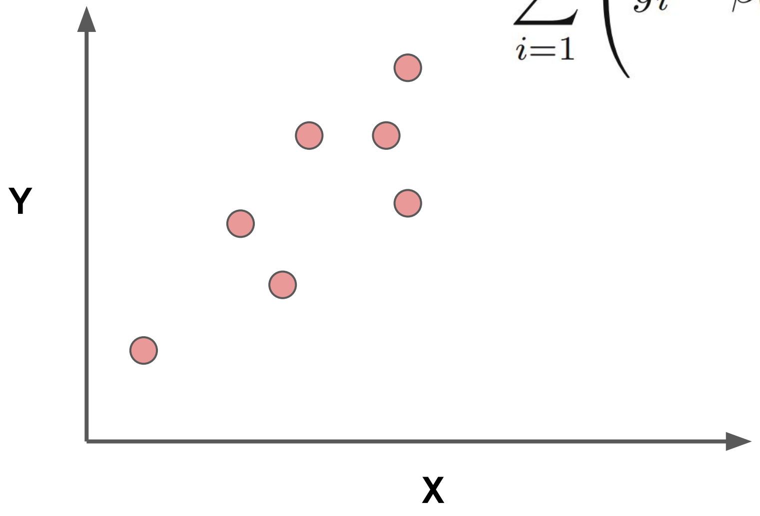
$$\sum_{i=1}^{n}\left(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij}\right)^2 + \lambda \sum_{j=1}^{p}\beta_j^2$$



Ridge Regression

- In the case of $\hat{\mathbf{y}} = \boxed{\boldsymbol{\beta}_1}\mathbf{x} + \boldsymbol{\beta}_0$

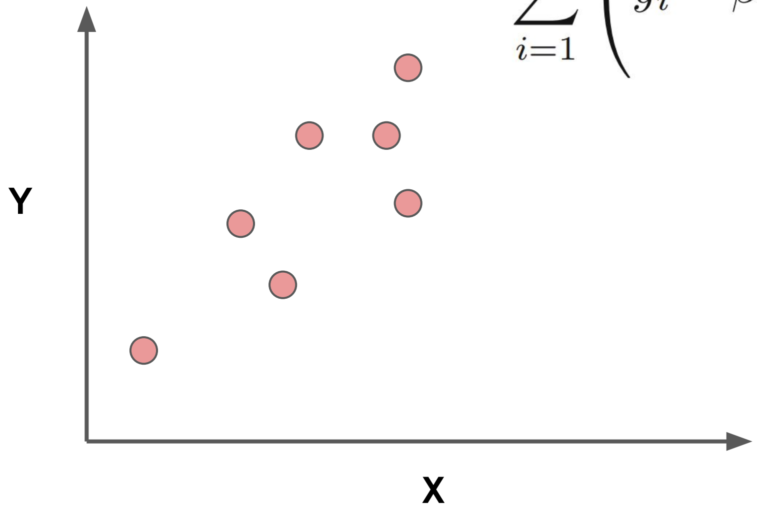
$$\sum_{i=1}^{n}\left(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij}\right)^2 + \boxed{\lambda\sum_{j=1}^{p}\beta_j^2}$$



Ridge Regression

- Let's assume **λ = 1**

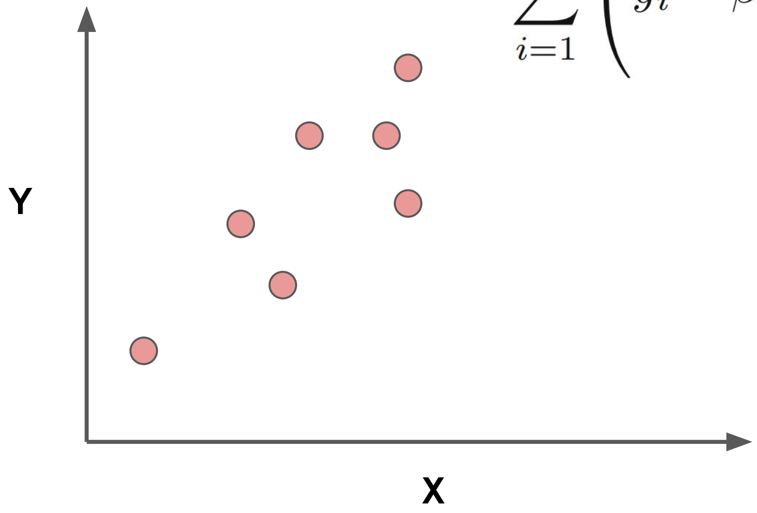
$$\sum_{i=1}^{n}\left(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij}\right)^2 + \lambda\sum_{j=1}^{p}\beta_j^2$$



Ridge Regression

- This punishes a large slope for $\hat{\mathbf{y}} = \boxed{\boldsymbol{\beta}_1}\mathbf{x} + \boldsymbol{\beta}_0$

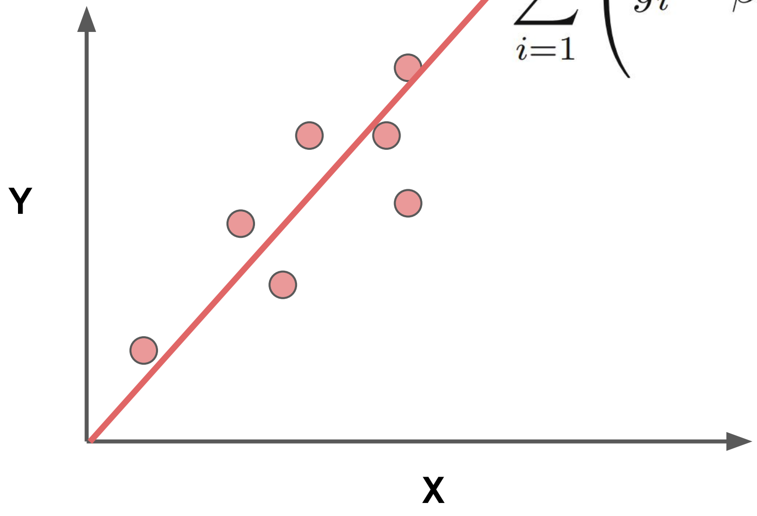
$$\sum_{i=1}^{n}\left(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij}\right)^2 + \boxed{\lambda\sum_{j=1}^{p}\beta_j^2}$$



Ridge Regression

- For single feature this lowers slope

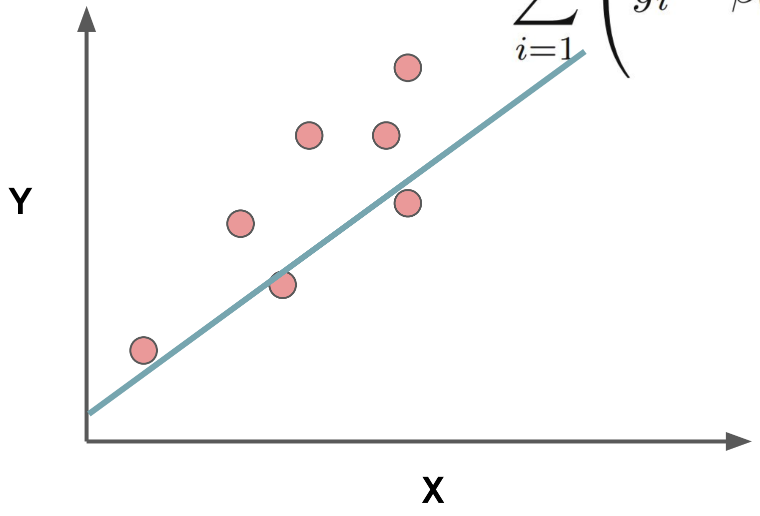
$$\sum_{i=1}^{n}\left(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij}\right)^2 + \boxed{\lambda \sum_{j=1}^{p}\beta_j^2}$$



Ridge Regression

- For single feature this lowers slope

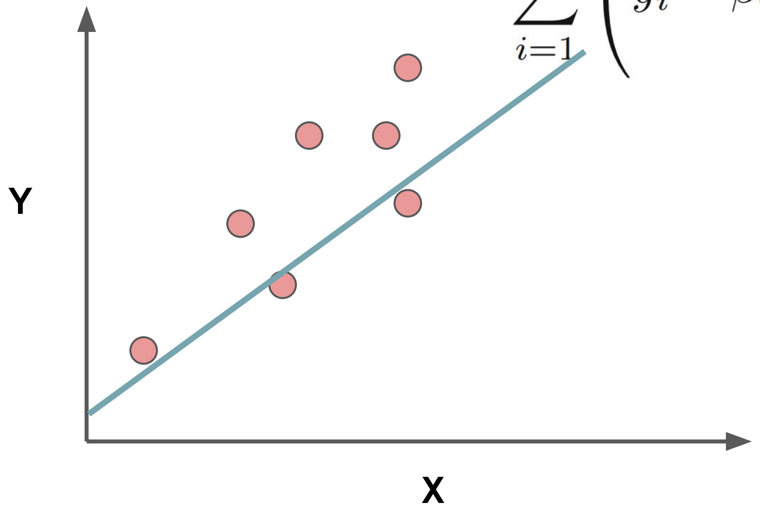
$$\sum_{i=1}^{n}\left(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij}\right)^2 + \boxed{\lambda \sum_{j=1}^{p}\beta_j^2}$$



Ridge Regression

- At the cost of some additional bias (error in training set)

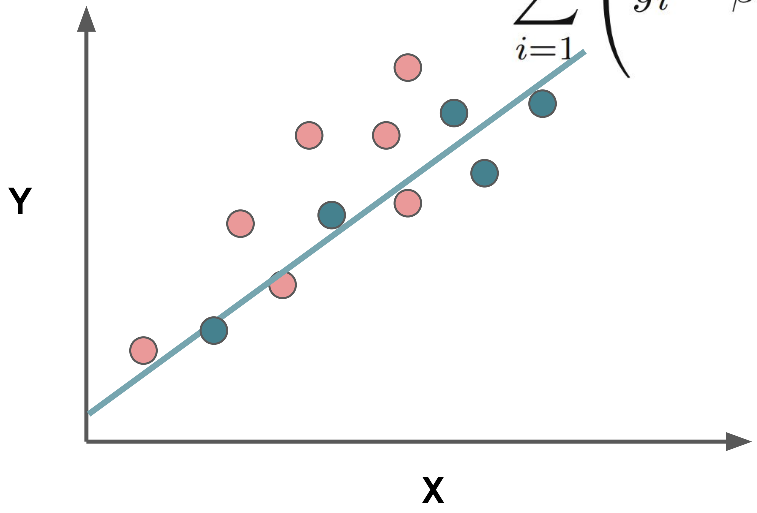
$$\sum_{i=1}^{n}\left(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij}\right)^2 + \boxed{\lambda\sum_{j=1}^{p}\beta_j^2}$$



Ridge Regression

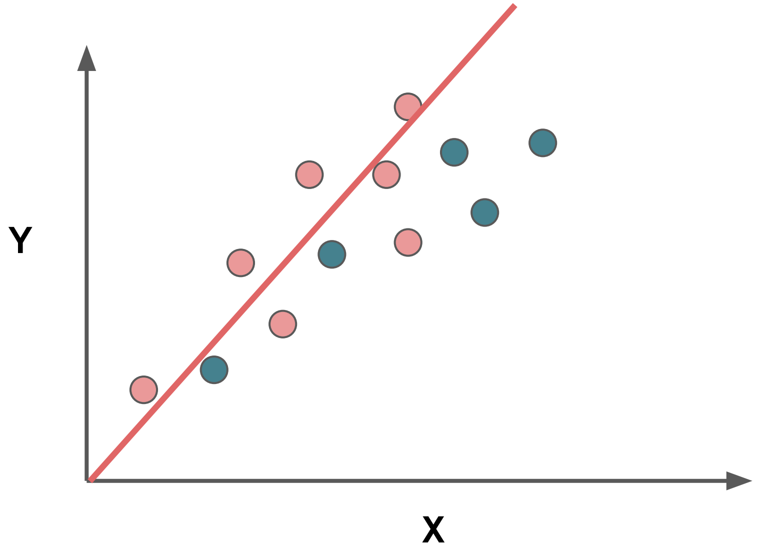
- We generalize better to unseen data

$$\sum_{i=1}^{n}\left(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij}\right)^2 + \boxed{\lambda \sum_{j=1}^{p}\beta_j^2}$$



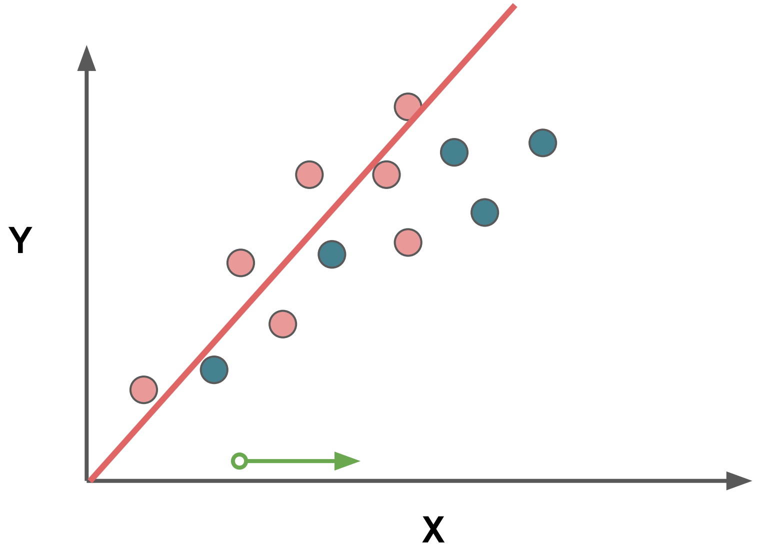
Ridge Regression

- Consider overfitting to training set:



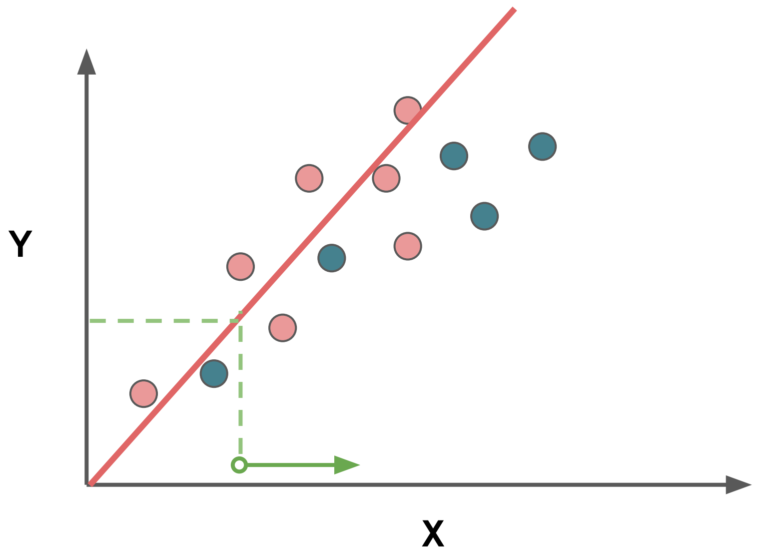
Ridge Regression

- An increase in X results in a greater y response:



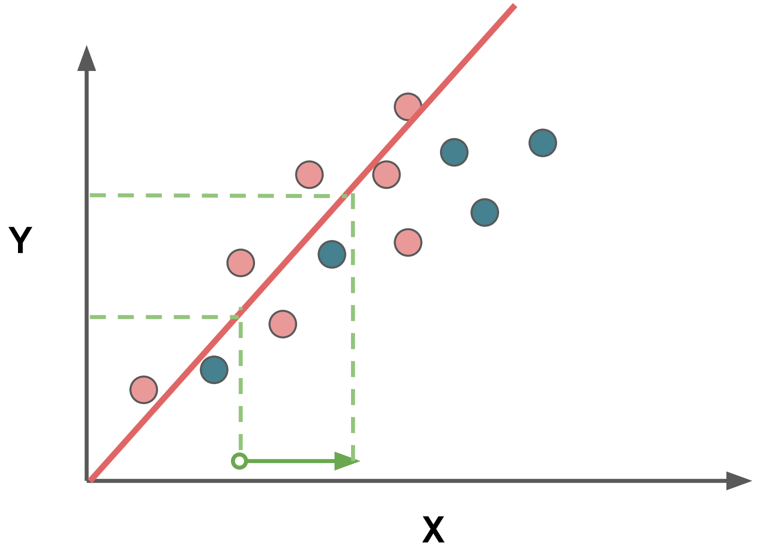
Ridge Regression

- An increase in X results in a greater y response:



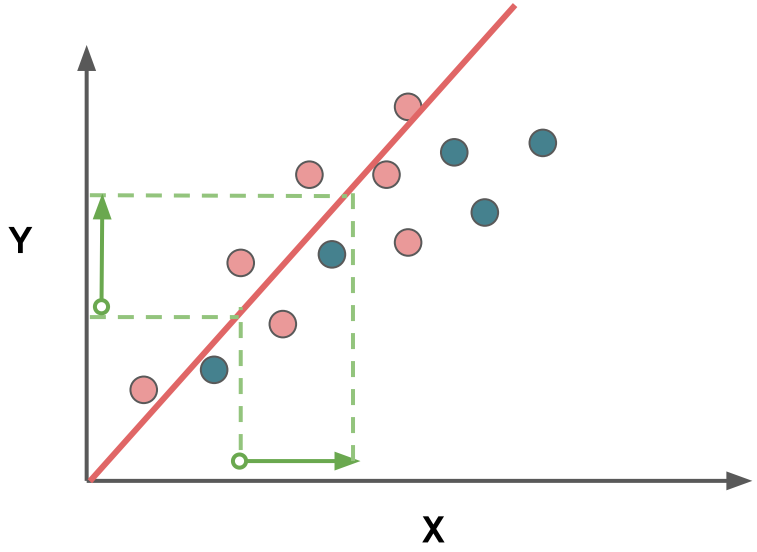
Ridge Regression

- An increase in X results in a greater y response:



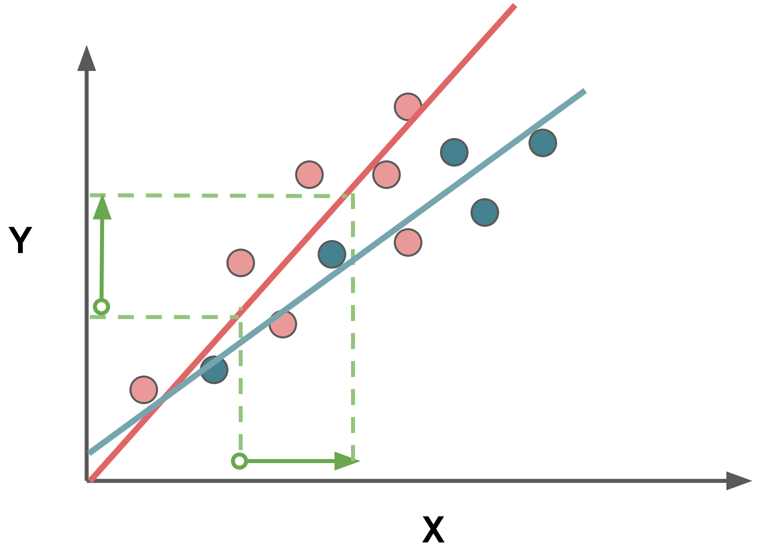
Ridge Regression

- An increase in X results in a greater y response:



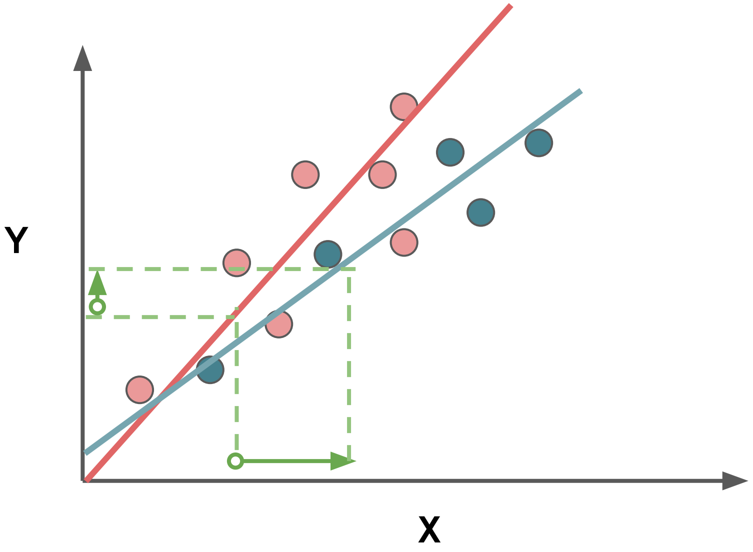
Ridge Regression

- Compare to a more generalized model that used Ridge Regression:



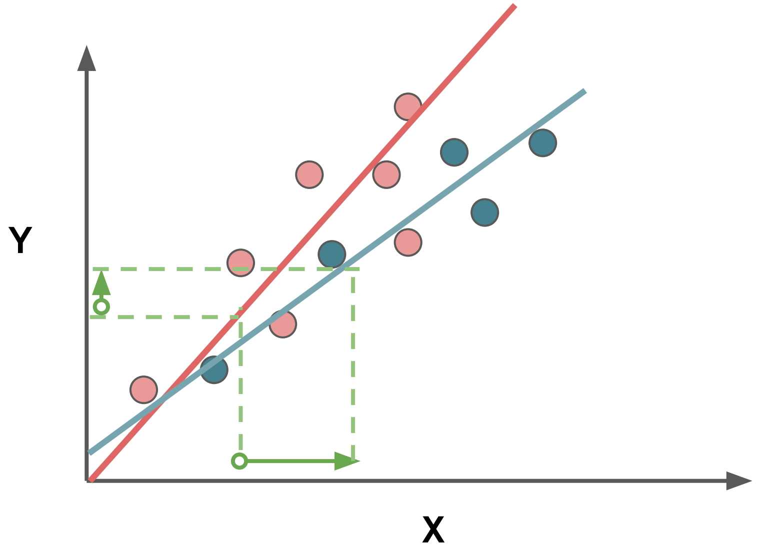
Ridge Regression

- Same feature change does not produce as much y response:



Ridge Regression

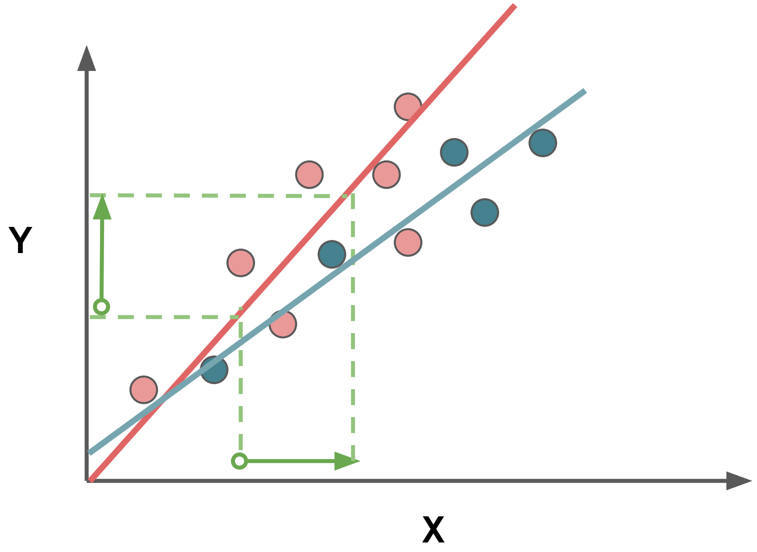
- Trying to minimize a squared Beta term leads us to punish larger coefficients.



$$\lambda \sum_{j=1}^{p} \beta_j^2$$

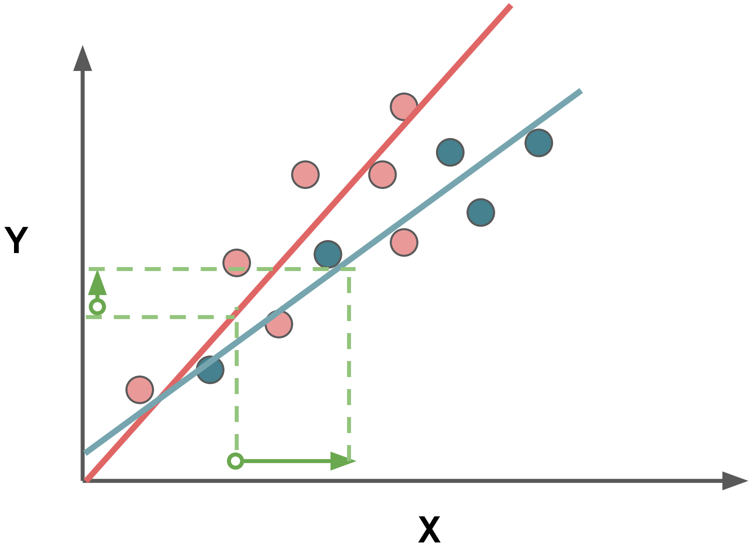
Ridge Regression

- In the case of a single feature, a larger Beta means a steeper sloped line.



Ridge Regression

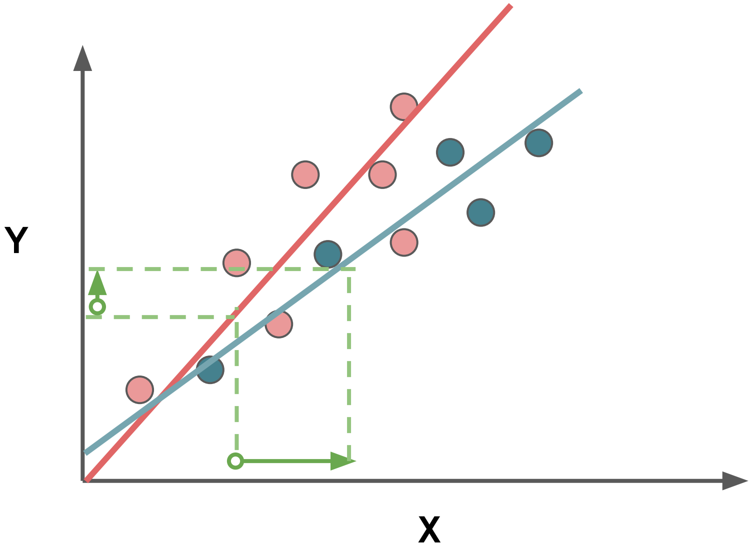
- A steeper sloped line would mean more response per increase in X value.



$$\lambda \sum_{j=1}^{p} \beta_j^2$$

Ridge Regression

- What about the lambda term? How much should we punish these larger coefficients?



$$\lambda \sum_{j=1}^{p} \beta_j^2$$

Ridge Regression

- We simply use cross-validation to explore multiple lambda options and then choose the best one!

$$\text{Error} = \sum_{i=1}^{n}\left(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij}\right)^2 + \lambda\sum_{j=1}^{p}\beta_j^2$$

Ridge Regression

L2 Regularization

Ridge Regression

- Important Note!
    - Sklearn refers to lambda as alpha within the class call!

$$\text{Error} = \sum_{i=1}^{n}\left(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij}\right)^2 + \boxed{\lambda}\sum_{j=1}^{p}\beta_j^2$$

Ridge Regression

- Important Note!
  - For cross validation metrics, sklearn uses a “scorer object”.
  - All scorer objects follow the convention that **higher** return values are **better** than lower return values.

Ridge Regression

- Important Note!
    - For example, obviously higher accuracy is better.
    - But higher RMSE is actually worse!
    - So Scikit-Learn fixes this by using a **negative** RMSE as its scorer metric.

Ridge Regression

- Important Note!
  - This allows for uniformity across **all** scorer metrics, even across different tasks types.
  - The same idea of uniformity across model classes applies to referring to the penalty strength parameter as **alpha**.

Lasso Regression

L1 Regularization

Regularization

- L1 regularization adds a penalty equal to the **absolute value** of the magnitude of coefficients.

$$\sum_{i=1}^{n}\left(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij}\right)^2 + \lambda\sum_{j=1}^{p}|\beta_j| = \mathrm{RSS} + \boxed{\lambda\sum_{j=1}^{p}|\beta_j|}$$

Regularization

- L1 regularization adds a penalty equal to the **absolute value** of the magnitude of coefficients.
    - Limits the size of the coefficients.
    - Can yield sparse models where some coefficients can become zero.

Regularization

- LASSO can force some of the coefficient estimates to be exactly equal to zero when the tuning parameter λ is sufficiently large.
- Similar to subset selection, the LASSO performs variable selection.
- Models generated from the LASSO are generally much easier to interpret.

Regularization

- LassoCV with sklearn operates on checking a number of alphas within a range, instead of providing the alphas directly.
- Let’s explore the results of LASSO in Python and Scikit-Learn!

Elastic Net

L1 and L2 Regularization

Regularization

- We’ve been able to perform Ridge and Lasso regression.
- We know Lasso is able to shrink coefficients to zero, but we haven’t taken a deeper dive into how or why that is.
- This ability becomes more clear when learning about **elastic net** which combines Lasso and Ridge together!

Regularization

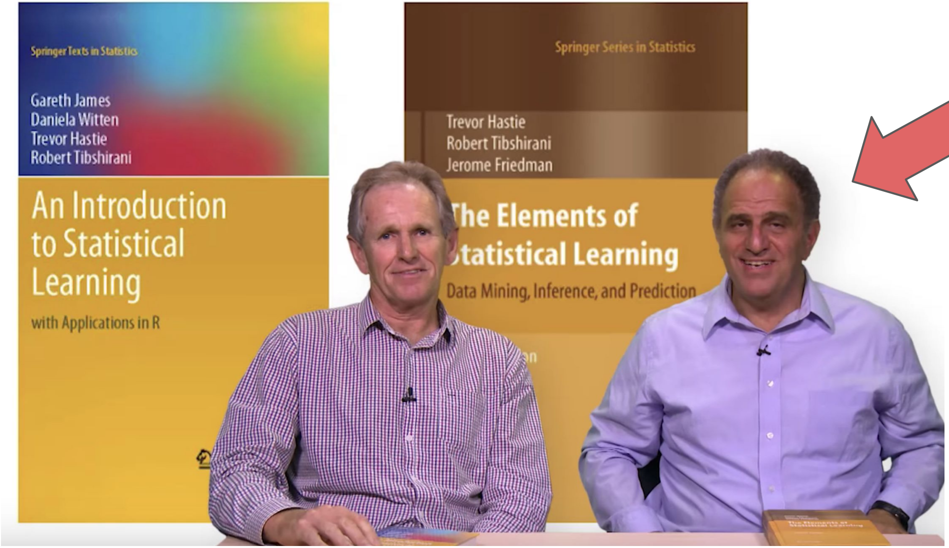
- Let's dive a little deeper into Lasso.
- Lasso was originally introduced in geophysics literature in 1986 by Symes and Santosa.
- It was later independently rediscovered and popularized in 1996 by Robert Tibshirani who coined the term "Lasso".

Regularization

* Does the name Robert Tibshirani sound familiar?

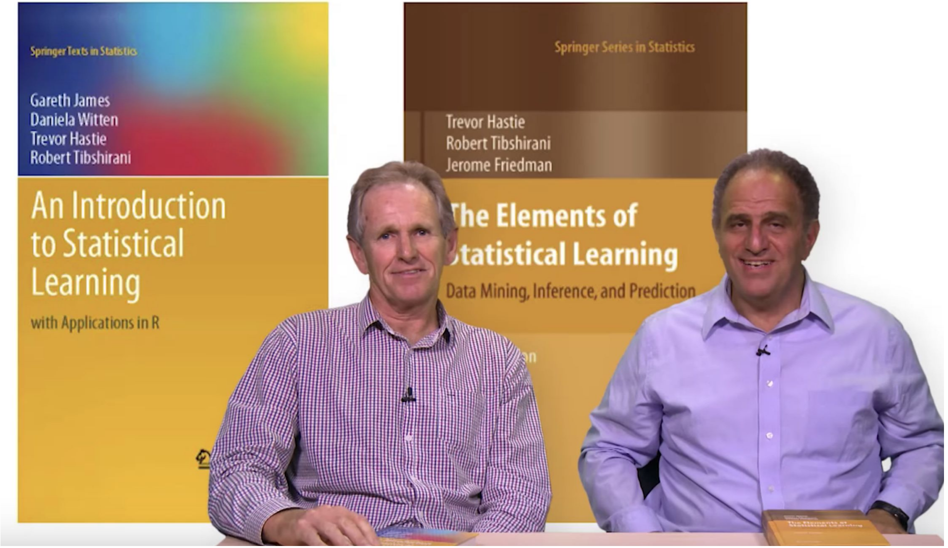
Regularization

* Robert Tibshirani is one of the authors of ISLR!



Regularization

- All the authors have a history of very impressive accomplishments!



Regularization

- We can rewrite Lasso and Ridge:

$$\underset{\beta}{\text{minimize}} \left\{ \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij} \right)^2 \right\} \quad \text{subject to} \quad \sum_{j=1}^{p} |\beta_j| \leq s$$

and

$$\underset{\beta}{\text{minimize}} \left\{ \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij} \right)^2 \right\} \quad \text{subject to} \quad \sum_{j=1}^{p} \beta_j^2 \leq s,$$

Regularization

- There is some sum **s** which allows to rewrite the penalty as a requirement:

$$\underset{\beta}{\text{minimize}} \left\{ \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij} \right)^2 \right\} \quad \text{subject to} \quad \sum_{j=1}^{p} |\beta_j| \leq s$$

and

$$\underset{\beta}{\text{minimize}} \left\{ \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij} \right)^2 \right\} \quad \text{subject to} \quad \sum_{j=1}^{p} \beta_j^2 \leq s,$$

Regularization

- There is some sum **s** which allows to rewrite the penalty as a requirement:

$$\underset{\beta}{\text{minimize}} \left\{ \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij} \right)^2 \right\} \quad \text{subject to} \quad \sum_{j=1}^{p} |\beta_j| \leq s$$

and

$$\underset{\beta}{\text{minimize}} \left\{ \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij} \right)^2 \right\} \quad \text{subject to} \quad \sum_{j=1}^{p} \beta_j^2 \leq s,$$

Elastic Net

- Start with a simple thought experiment:
  - A simple equation:
    - $\hat{y} = \beta_1 x_1 + \beta_2 x_2$
  - We know that regularization can be expressed as an additional requirement that RSS is subject to.

Elastic Net

- Start with a simple thought experiment:
  - A simple equation:
    - $\hat{y} = \beta_1 x_1 + \beta_2 x_2$
  - L1 constrains the sum of absolute values.
    - $\sum|\beta|$
  - L2 constrains the sum of squared values.
    - $\sum\beta^2$

Elastic Net

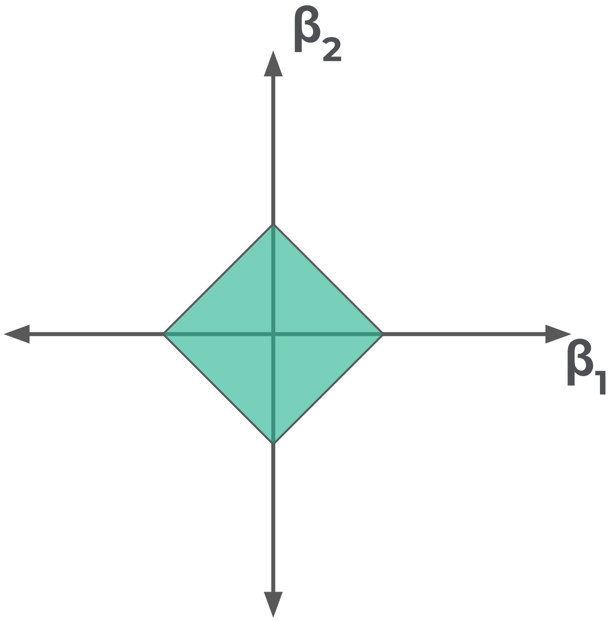
- Start with a simple thought experiment:
  - A simple equation:
    - $\hat{y} = \beta_1 x_1 + \beta_2 x_2$
  - There is some sum **s** that the penalty is less than.

Elastic Net

- For the case of only two features:
  - $\hat{y} = \beta_1 x_1 + \beta_2 x_2$
- Lasso Regression Penalty:
  - $|\beta_1| + |\beta_2| \leq s$
- Ridge Regression Penalty:
  - $\beta_1^2 + \beta_2^2 \leq s$

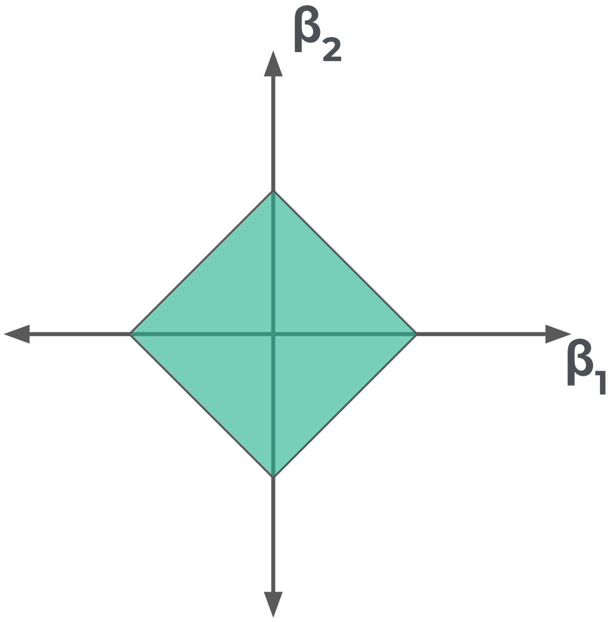
Elastic Net

- Let's plot Lasso: $|\boldsymbol{\beta_1}| + |\boldsymbol{\beta_2}| \leq \mathbf{s}$



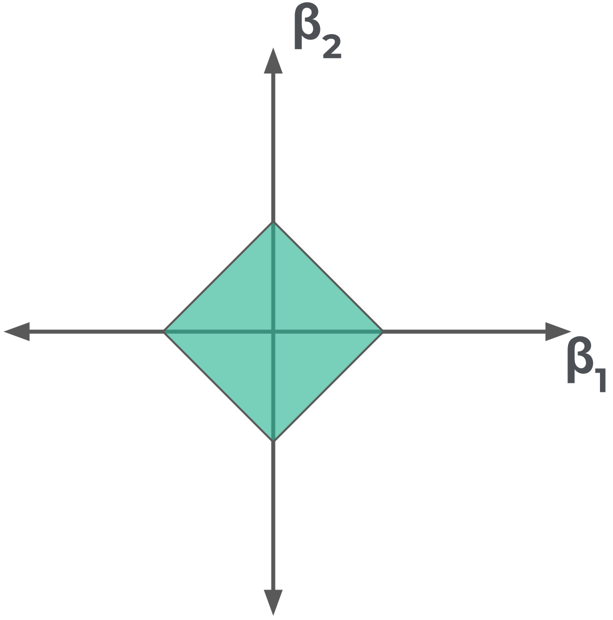
Elastic Net

- Let's plot Lasso: $|\beta_1| + |\beta_2| \leq s$



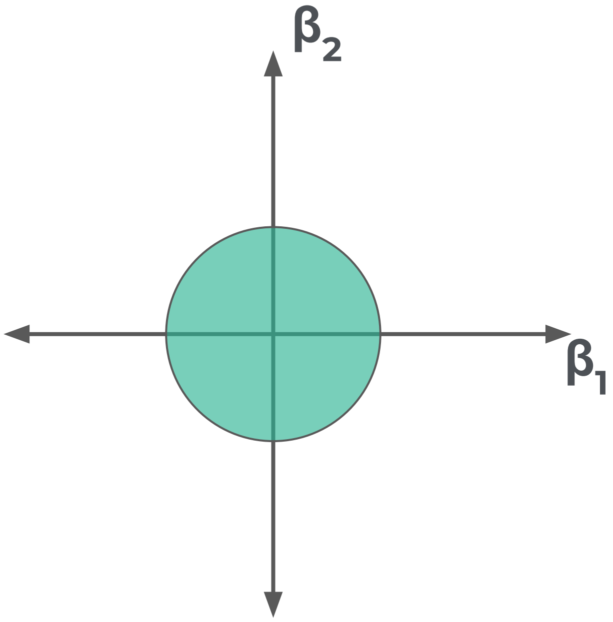
Elastic Net

- Let's plot Lasso: $|\boldsymbol{\beta_1}| + |\boldsymbol{\beta_2}| \leq \mathbf{s}$



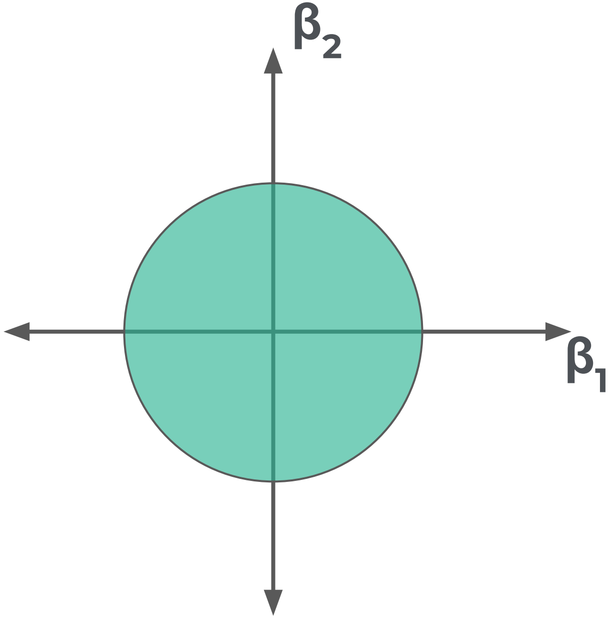
Elastic Net

- Let's plot Ridge: $\boldsymbol{\beta_1^2 + \beta_2^2 \leq s}$



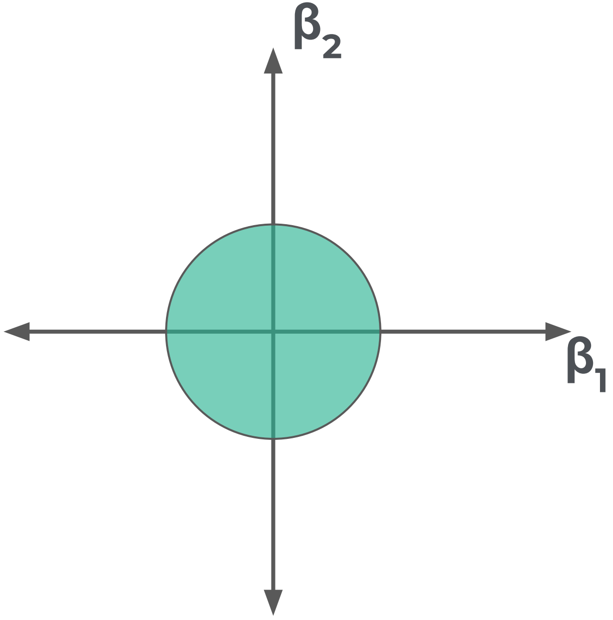
Elastic Net

- Let's plot Ridge: $\beta_1^2 + \beta_2^2 \leq s$



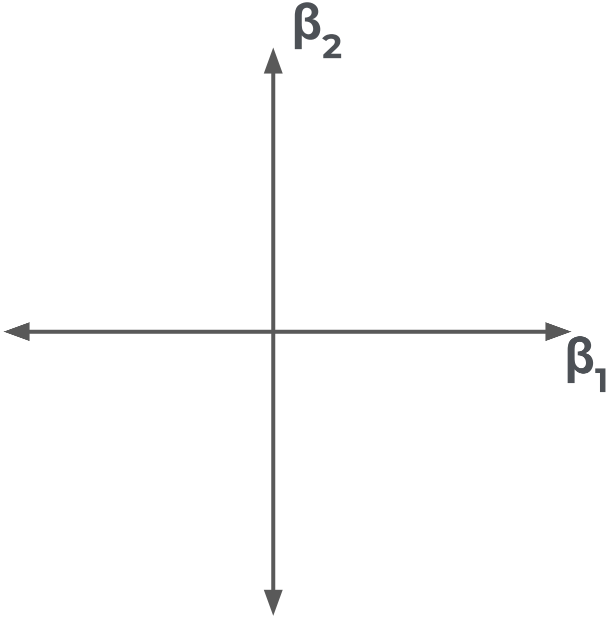
Elastic Net

- Let's plot Ridge: $\beta_1^2 + \beta_2^2 \leq s$



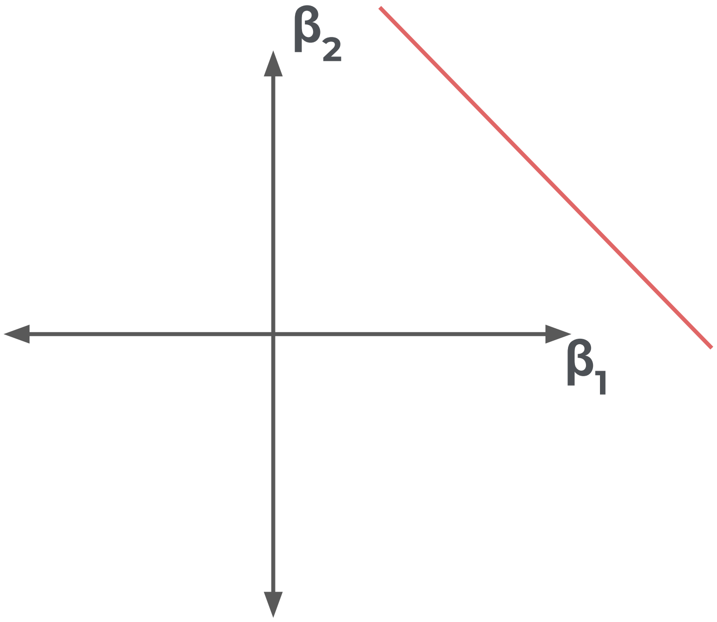
Elastic Net

- What would RSS look like?



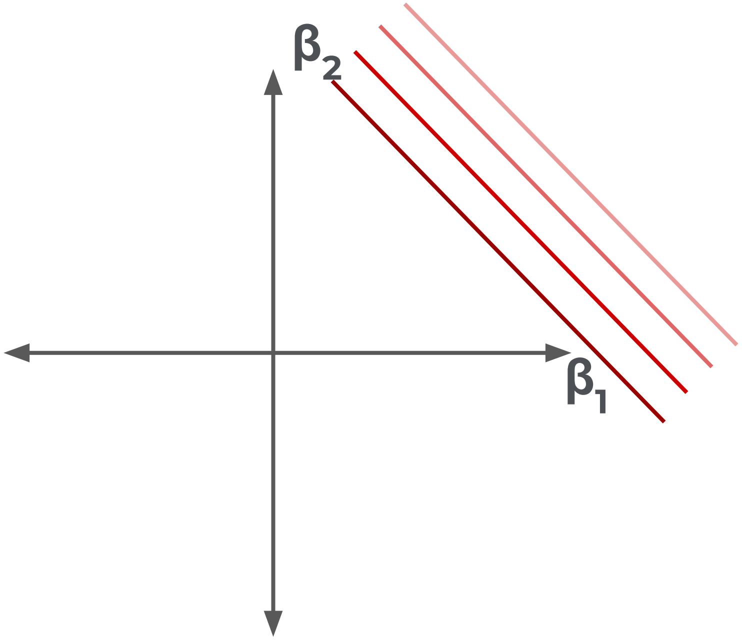
Elastic Net

* What would RSS look like?



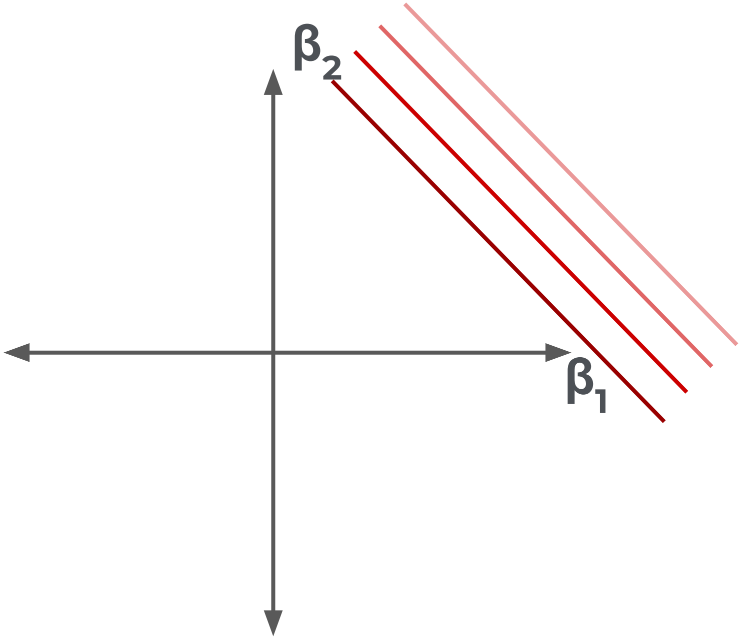
Elastic Net

- What would RSS look like?



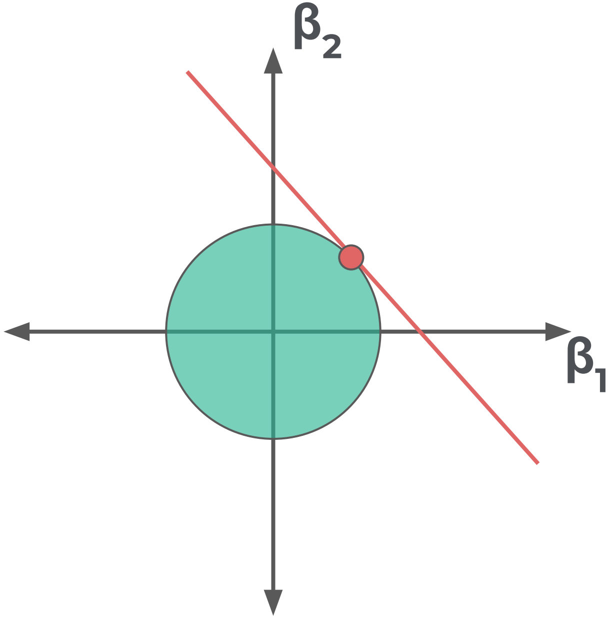
Elastic Net

- But were subject to the penalty!



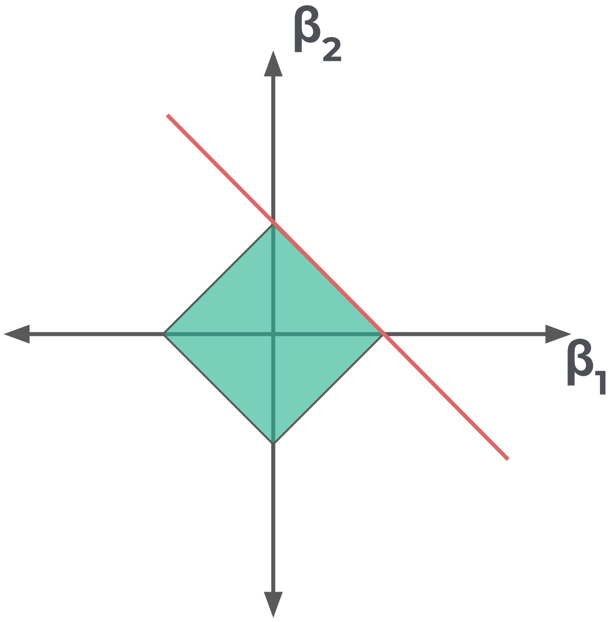
Elastic Net

- Penalty for Ridge: $\boldsymbol{\beta_1^2 + \beta_2^2 \leq s}$



Elastic Net

- Penalty for Lasso: $|\boldsymbol{\beta_1}| + |\boldsymbol{\beta_2}| \leq \mathbf{s}$



Elastic Net

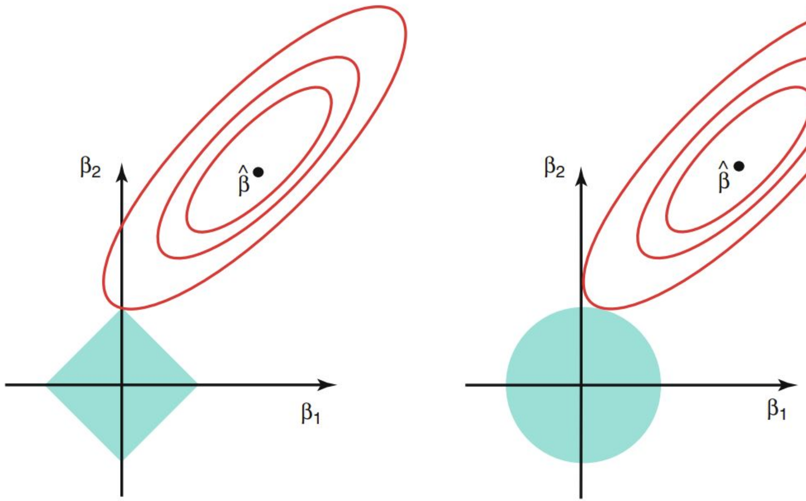
- Lasso:
  - A convex object that lies tangent to the boundary, is likely to encounter a corner of a hypercube, for which some components of **β** are identically zero.

Elastic Net

- Ridge: In the case of an n-sphere, the points on the boundary for which some of the components of **β** are zero are not distinguished from the others and the convex object is no more likely to contact a point at which some components of **β** are zero than one for which none of them are.

Elastic Net

- This is why Lasso is more likely to lead to coefficients as zero.
- This diagram is also commonly shown with contour RSS:



Elastic Net

- Elastic Net seeks to improve on both L1 and L2 Regularization by combining them:

$$\text{Error} = \sum_{i=1}^{n}\left(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij}\right)^2 + \lambda_1 \sum_{j=1}^{p}\beta_j^2 + \lambda_2 \sum_{j=1}^{p}|\beta_j|$$

Elastic Net

- Here we seek to minimize RSS and **both** the squared and absolute value terms:

$$\text{Error} = \sum_{i=1}^{n}\left(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij}\right)^2 + \lambda_1\sum_{j=1}^{p}\beta_j^2 + \lambda_2\sum_{j=1}^{p}|\beta_j|$$

Elastic Net

- Notice there are **two** distinct lambda values for each penalty:

$$\text{Error} = \sum_{i=1}^{n}\left(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij}\right)^2 + \lambda_1 \sum_{j=1}^{p}\beta_j^2 + \lambda_2 \sum_{j=1}^{p}|\beta_j|$$

Elastic Net

- We can alternatively express this as a ratio between L1 and L2:

$$\frac{\sum_{i=1}^{n}(y_i - x_i^J\hat{\beta})^2}{2n} + \lambda\left(\frac{1-\alpha}{2}\sum_{j=1}^{m}\hat{\beta}_j^2 + \alpha\sum_{j=1}^{m}|\hat{\beta}_j|\right)$$

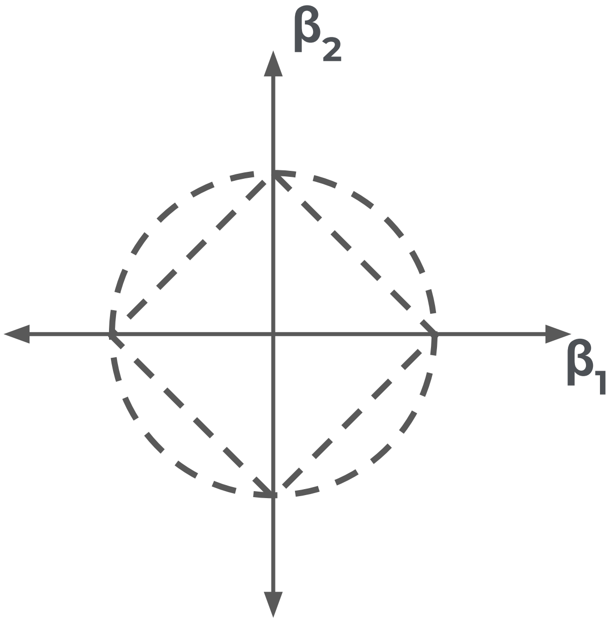
Elastic Net

- We can also use simplified notation:

$$\hat{\beta} \equiv \underset{\beta}{\operatorname{argmin}}(||y - X\beta||^2 + \lambda_2 ||\beta||^2 + \lambda_1 ||\beta||_1)$$

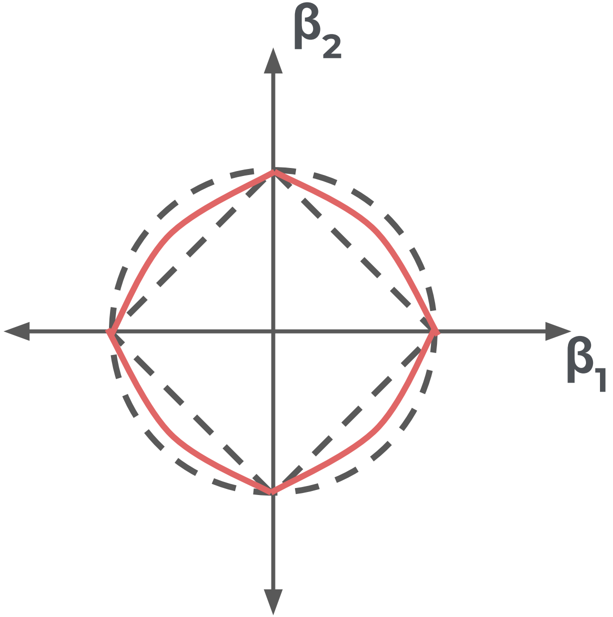
Elastic Net

- Elastic Net Penalty Region:



Elastic Net

- Elastic Net Penalty Region:



Elastic Net

- Let's explore how to perform Elastic Net with Python and Scikit-learn!

$$\frac{\sum_{i=1}^{n}(y_i - x_i^J \hat{\beta})^2}{2n} + \lambda\left(\frac{1-\alpha}{2}\sum_{j=1}^{m}\hat{\beta}_j^2 + \alpha\sum_{j=1}^{m}|\hat{\beta}_j|\right)$$

Linear Regression Project Dataset

Overview