An Introduction to Machine Learning with Python Programming
11 Sep 2023 - 20 Oct 2023

Conducted by:
IHUB Divya Sampark, IIT Roorkee
and
Ritvij Bharat Private Limited (RBPL)

Presented by:
Shreyas Shukla

Cross Validation

- Is there a way we can achieve the following:
  - Train on **ALL** the data
  - Evaluate on **ALL** the data?

Consider this dataset:

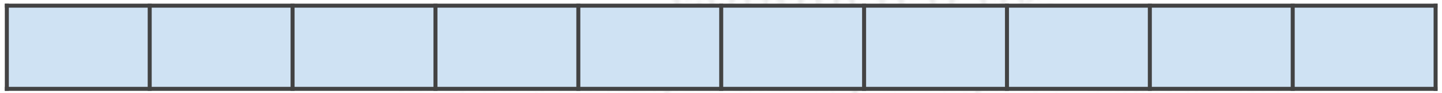
	X		**y**
Area m^2	**Bedrooms**	**Bathrooms**	**Price**
200	3	2	$500,000
190	2	1	$450,000
230	3	3	$650,000
180	1	1	$400,000
210	2	2	$550,000

Consider training vs testing:

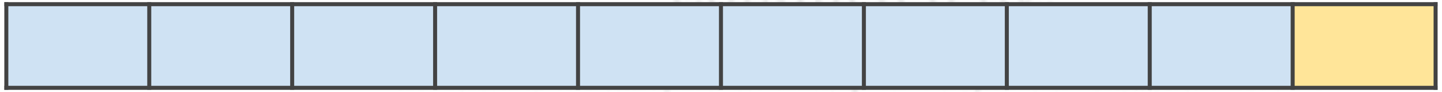
	X			**y**
	$\mathbf{x_1}$	$\mathbf{x_2}$	$\mathbf{x_3}$	$\mathbf{y}$
TRAIN	$x^1{}_1$	$x^1{}_1$	$x^1{}_1$	y_1
	$x^2{}_1$	$x^2{}_1$	$x^2{}_1$	y_2
	$x^3{}_1$	$x^3{}_1$	$x^3{}_1$	y_3
TEST	$x^4{}_1$	$x^4{}_1$	$x^4{}_1$	y_4
	$x^5{}_1$	$x^5{}_1$	$x^5{}_1$	y_5

Now we can represent full data and splits:

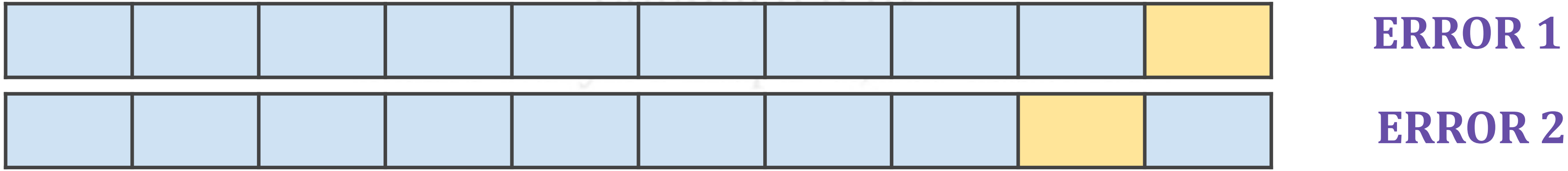
Split data into K equal parts:



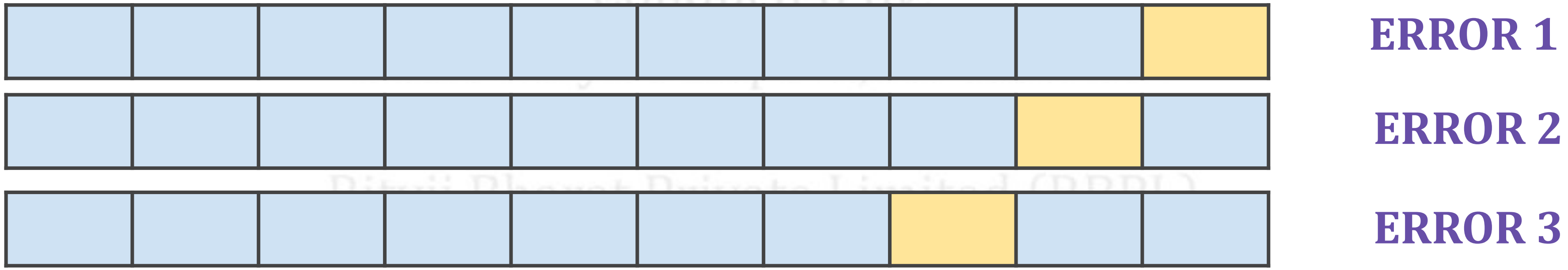
- 1/K left as test set
- Train model and get error metric for split:



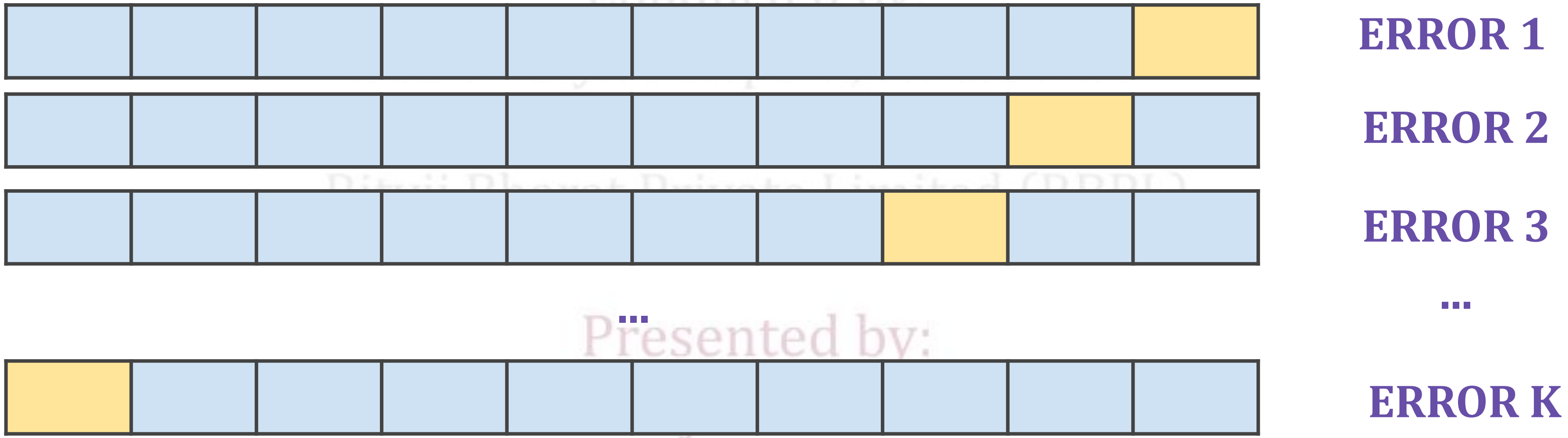
Repeat for another 1/K split



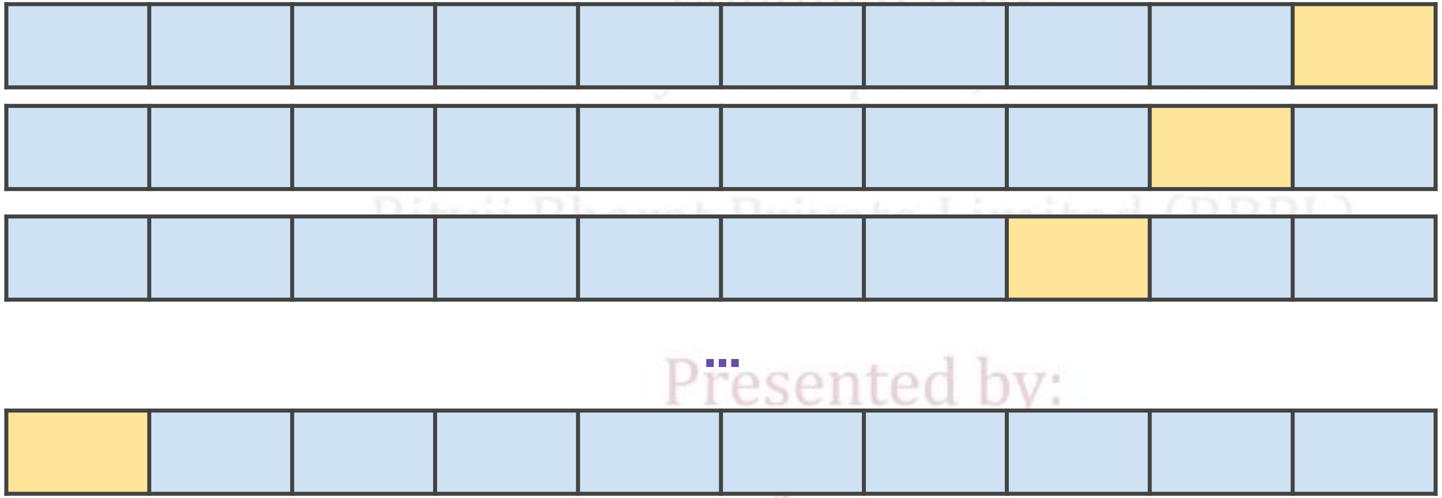
And again



Do it for all possible splits



Get average error



ERROR 1

ERROR 2

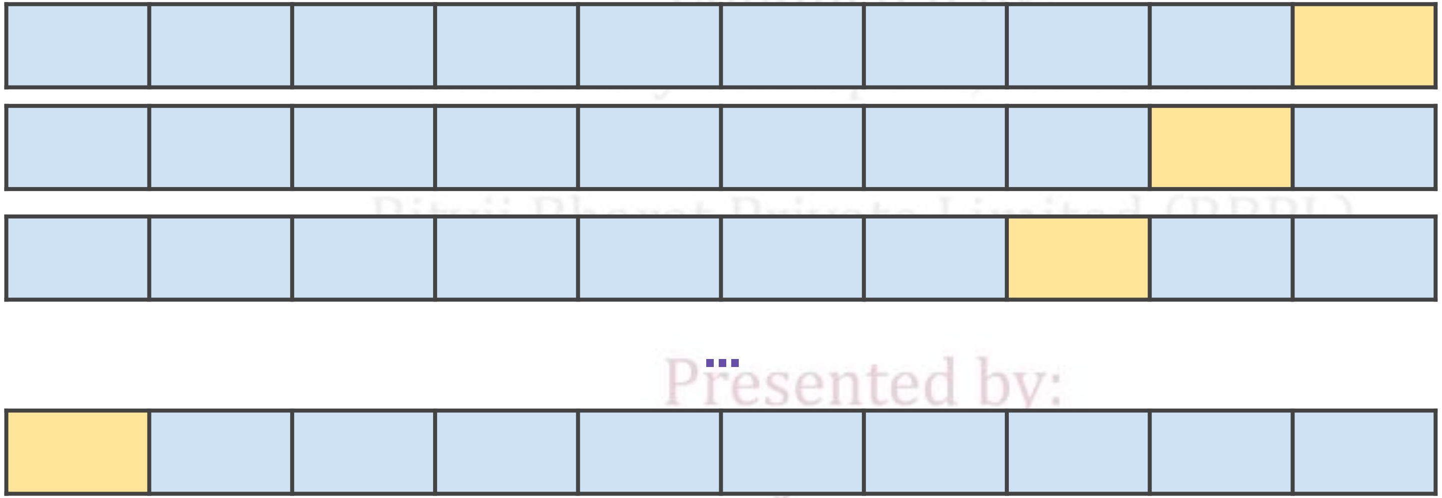
ERROR 3

...

ERROR K

MEAN ERROR

Average error is the expected performance



ERROR 1

ERROR 2

ERROR 3

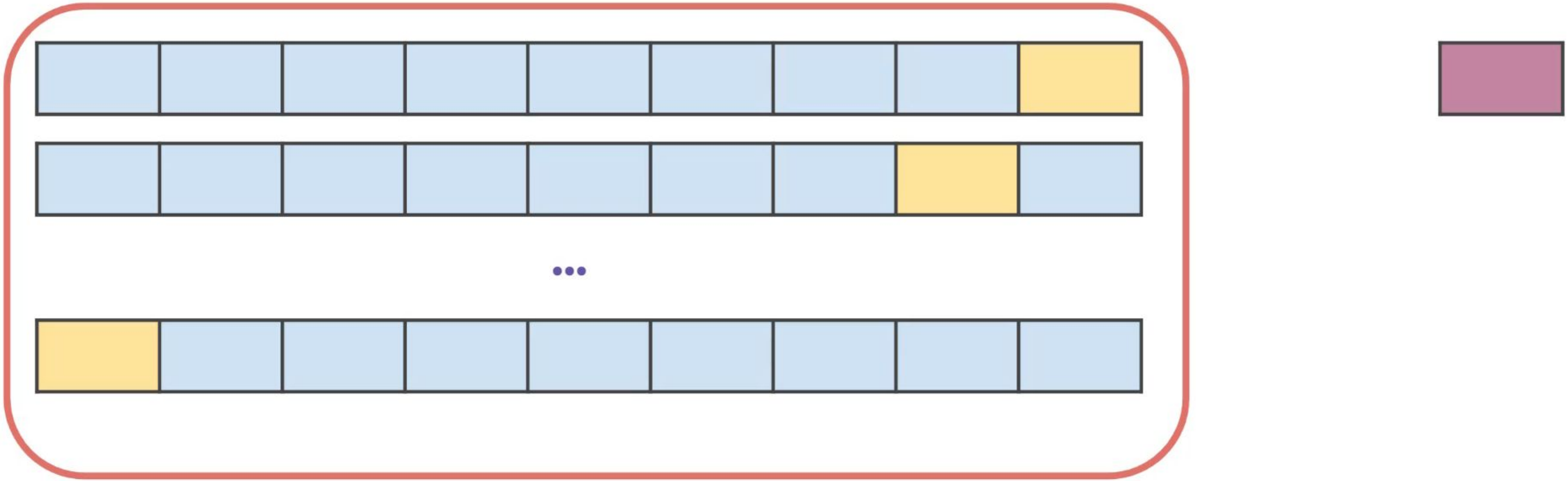
...

ERROR K

MEAN ERROR

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

Hold-Out Test Set: Train | Validation Split | Test



Regularization for Linear Regression

Jupiter Exercise

Ridge Regression

- Help reduce the potential for overfitting to the training data.
- Adds a penalty term to the error based on the squared value of the coefficients.
- Ridge Regression is a regularization method for Linear Regression.

General formula for the regression line:

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

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

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

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

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

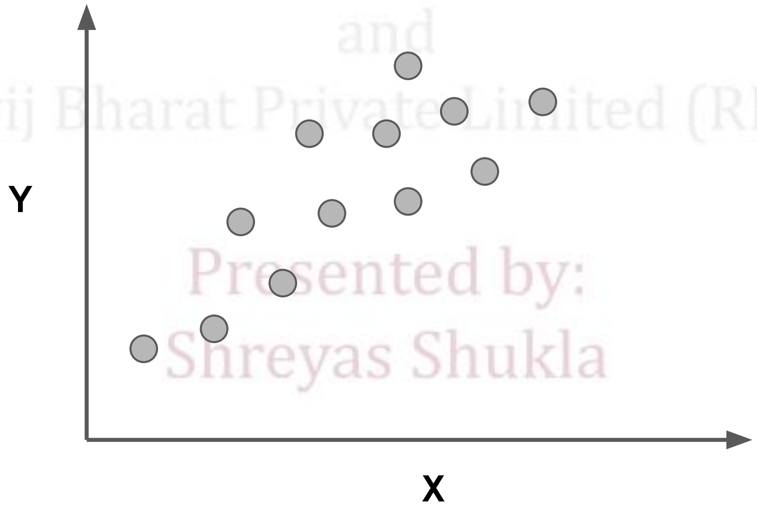
Summarize RSS:

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

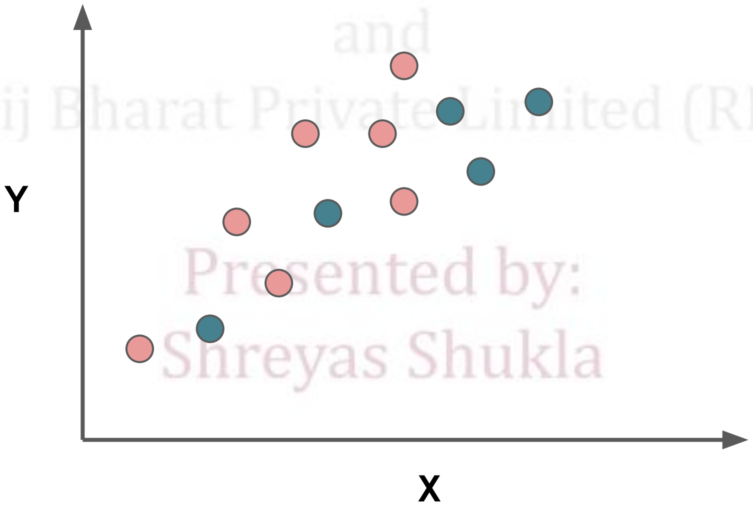
- Ridge Regression adds a **shrinkage penalty**
- Ridge Regression seeks to minimize this entire error term **RSS + Penalty**.
- **Shrinkage penalty** based off the squared coefficient:
- **Shrinkage penalty** has a **tunable lambda parameter which determines how severe the penalty is. Theoretically,** it can be any value from 0 to positive infinity.

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

Thought experiment

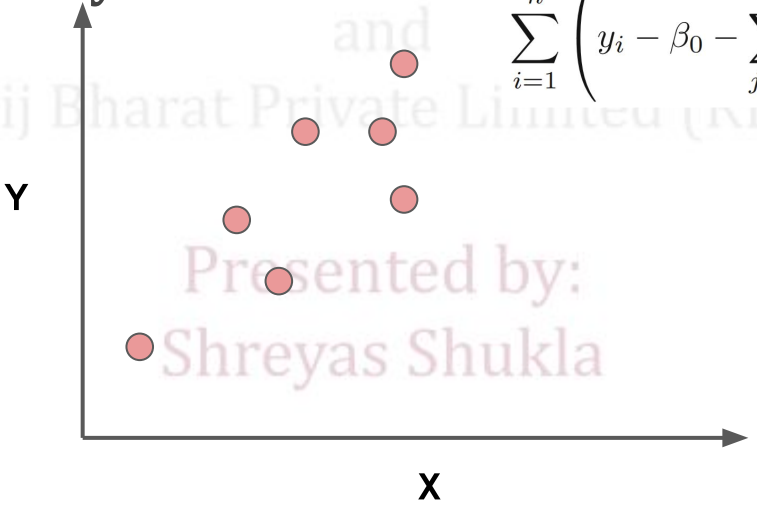


Split the dataset into a training set and test set:



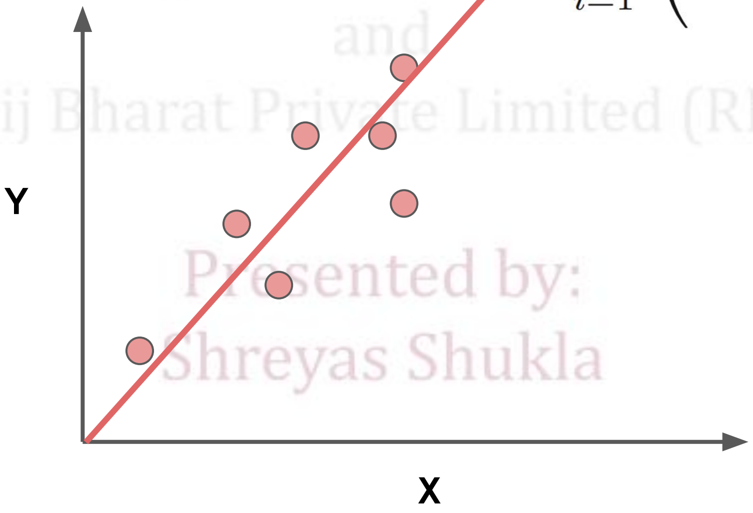
- Now we can fit on the training data to produce the line: $\hat{y} = \boldsymbol{\beta}_1 \mathbf{x} + \boldsymbol{\beta}_0$
- Regardless of RSS or Ridge error, we're still trying to create a line: $\hat{y} = \boldsymbol{\beta}_1 \mathbf{x} + \boldsymbol{\beta}_0$
- The only difference would be the coefficients found.
- **First let's fit using only RSS...**

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



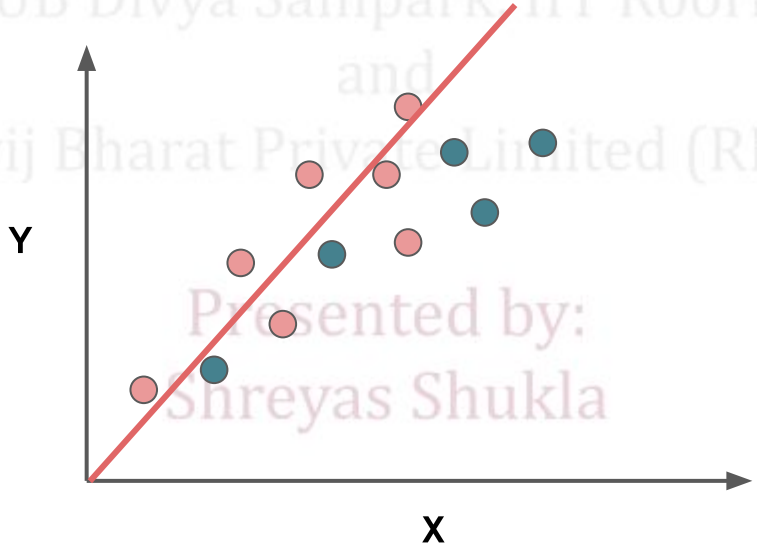
- Our fitted $\hat{y} = \beta_1 x + \beta_0$
- Appears to have over fit to training data.

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



This means we have high **variance.**

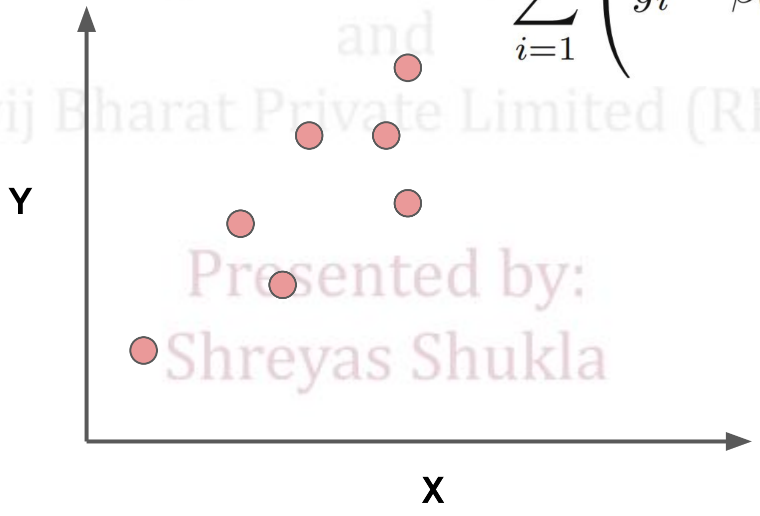
Could we introduce a little more **bias** to significantly **reduce** variance?



- Would adding the penalty term help generalize with more **bias**?
- Adding bias can help generalize $\hat{y} = \beta_1 x + \beta_0$

- Let’s imagine trying to reduce the Ridge Regression error term:
- There is λ and the squared slope coefficient.

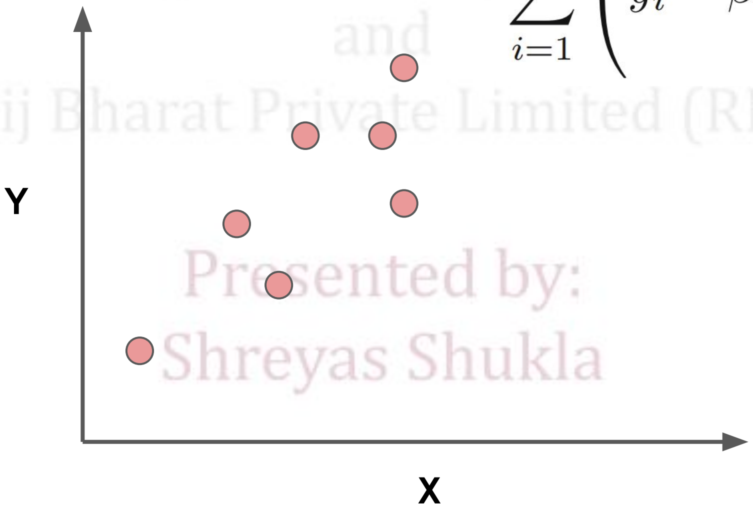
$$\sum_{i=1}^{n}\left(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij}\right)^2 + \boxed{\lambda\sum_{j=1}^{p}\beta_j^2}$$



Assume λ = 1

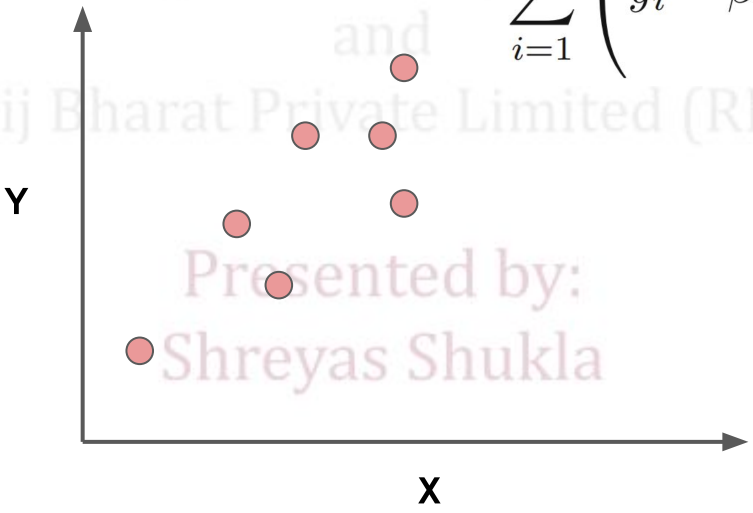
Then essentially, we're trying to minimize is the beta coefficient and the beta coefficient squared.

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



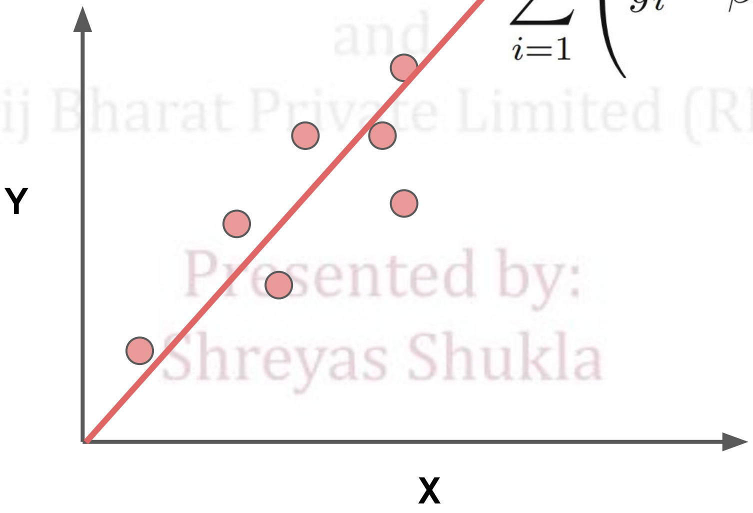
This punishes a large slope for $\hat{y} = \beta_1 x + \boxed{\beta_0}$

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



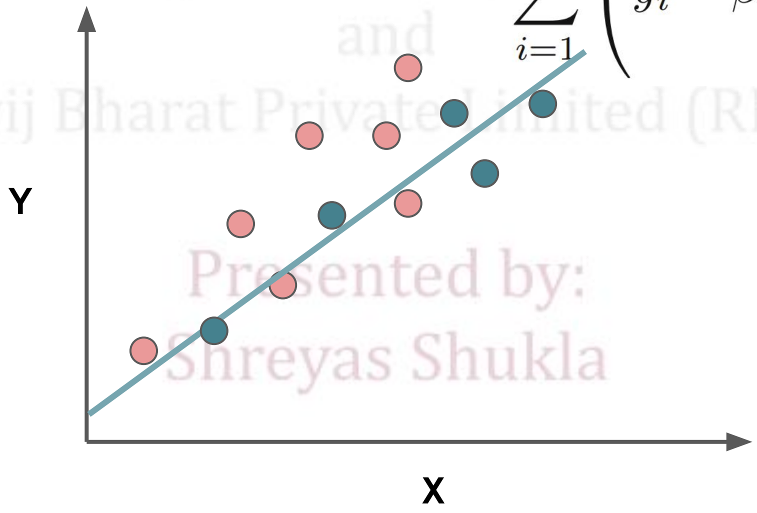
For single feature this lowers slope at the cost of some additional bias.

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

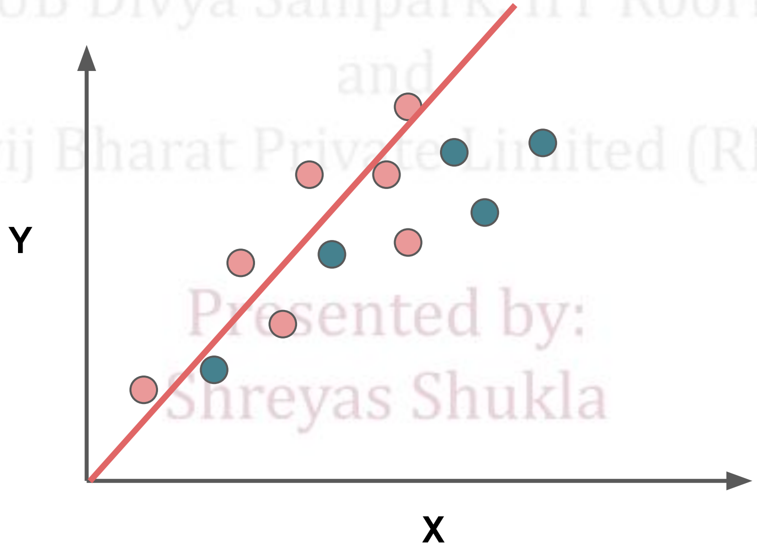


Generalize better to unseen data

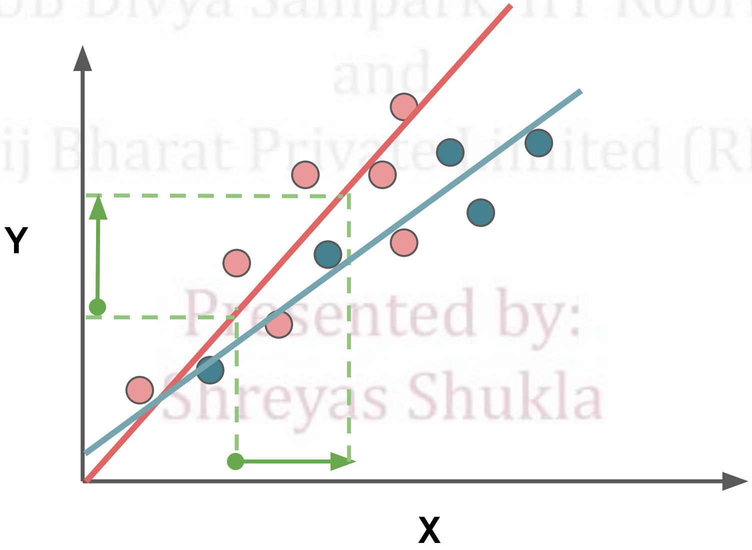
$$\sum_{i=1}^{n}\left(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij}\right)^2 + \boxed{\lambda \sum_{j=1}^{p}\beta_j^2}$$



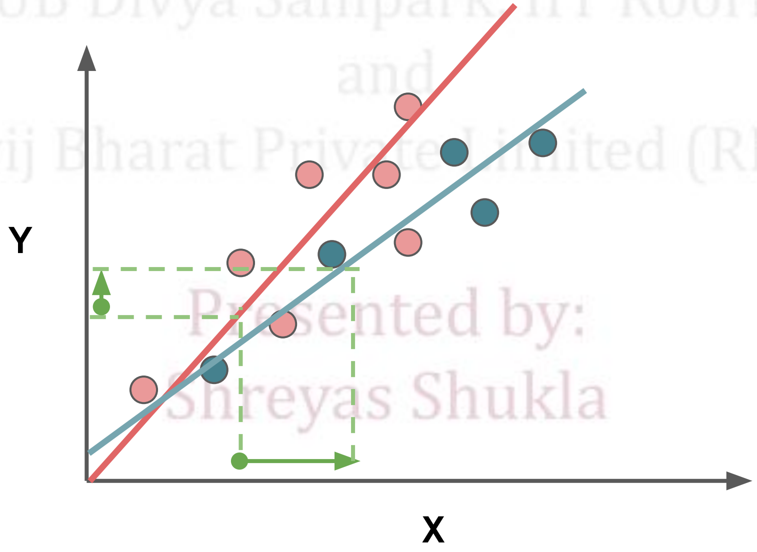
- Consider overfitting to training set
- An increase in X results in a greater y response:



- Compare to a more generalized model that used Ridge Regression
- Same feature change does not produce as much y response:

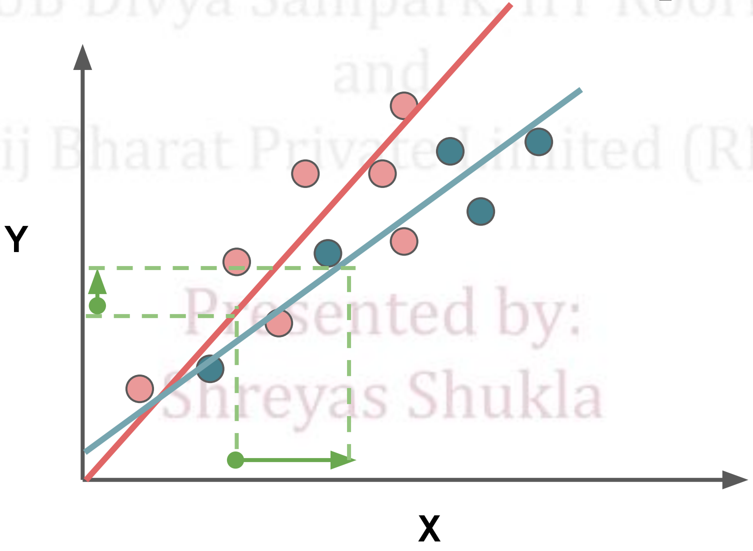


Same feature change does not produce as much y response



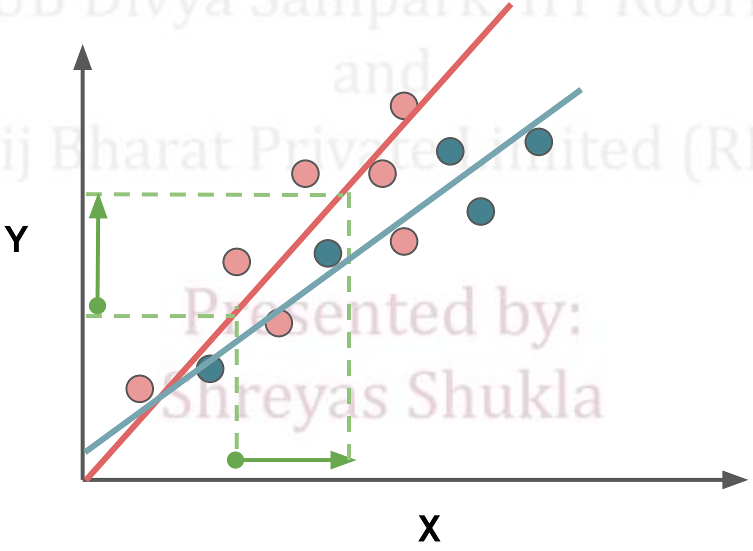
Ridge Regression

- Trying to minimize a squared Beta term leads us to punish larger coefficients.
- In the case of a single feature, a larger Beta means a steeper sloped line.
- A steeper sloped line would mean more response per increase in X value.



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

Again, in the case of a single feature that larger beta means a steeper sloped line and that would would mean more response per increase in X value.



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

Important Notes

- Sklearn refers to lambda as alpha
- 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.
- 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.

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

- 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

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

L1 adds a penalty which is equal to the **absolute value** of the magnitude of coefficients.

How is it different from L2 ?

- Limits the size of the coefficients.
- Can yield sparse models where some coefficients can become zero.

- LASSO can make some of the coefficients to be zero when the tuning parameter λ is sufficiently large.
- As a result, Models generated from the LASSO are generally much easier to interpret.

- LassoCV 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!