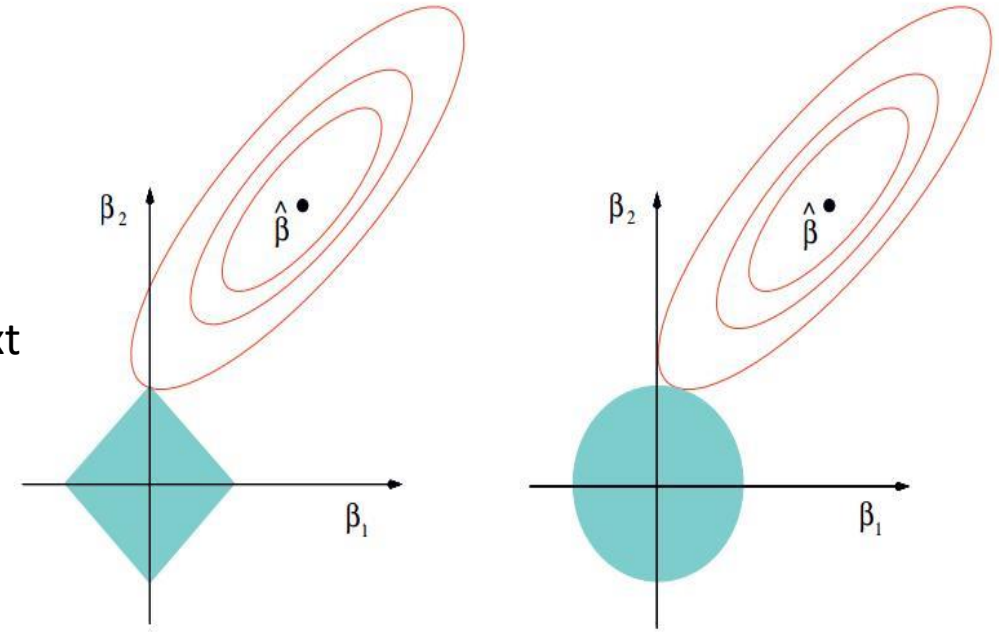
Lasso – Ridge Regression

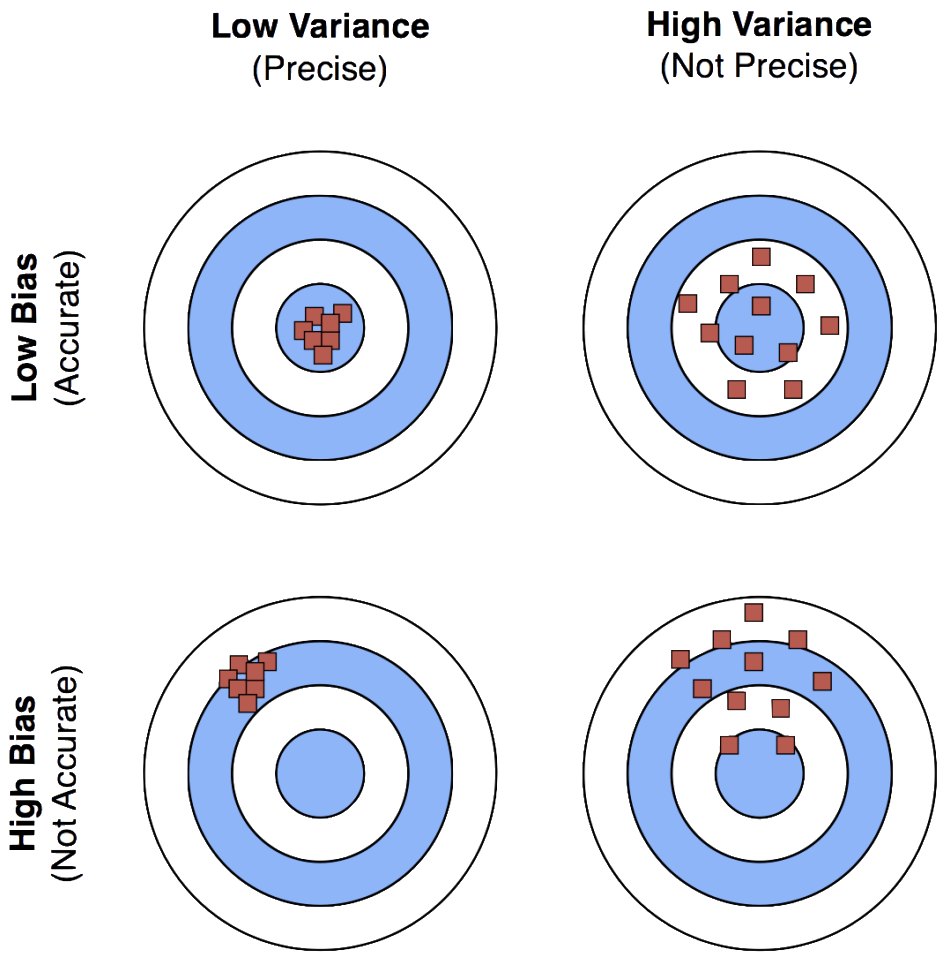
Click to add text
Click to add text



Objective

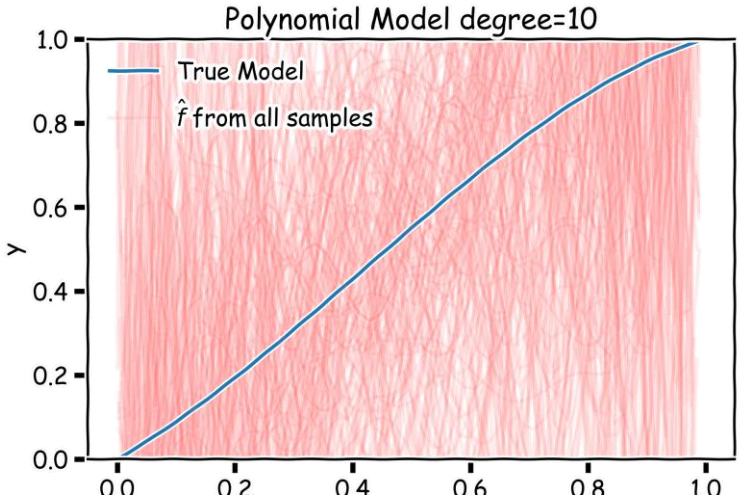
- Bias Variance Trade-off
- Regularized Regression
- Lasso Regression
- Ridge regression
- Choosing λ
- Evaluation metrics
- Implementation

Bias Variance Trade-off

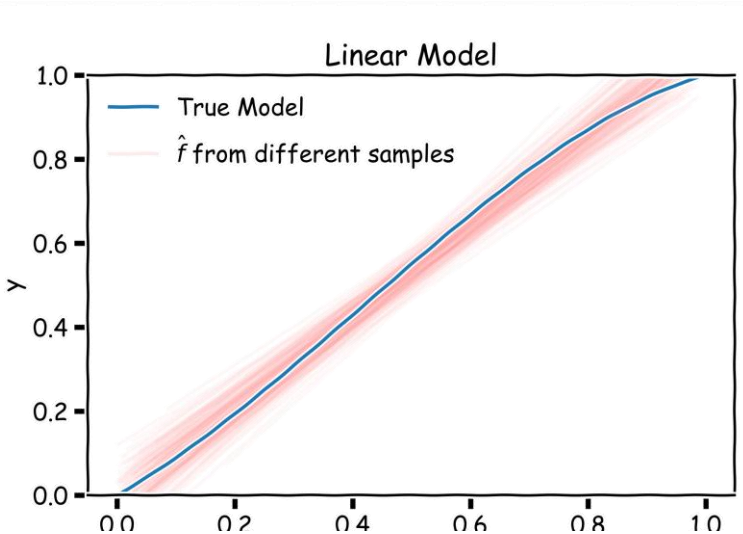


Bias vs Variance

- Best-fit models using degree 10 polynomial

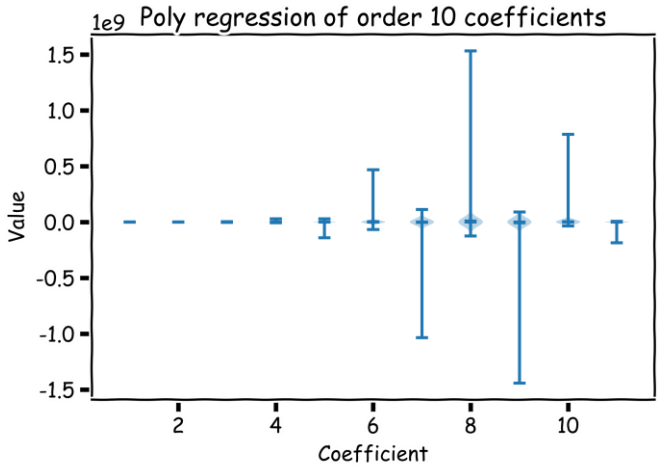


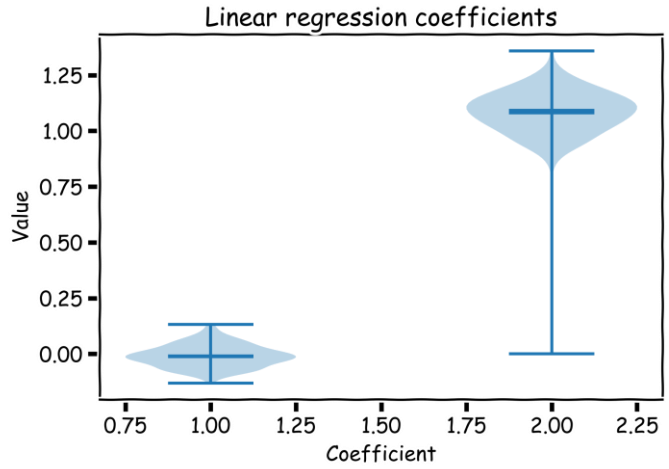
- 2000 best fit straight lines, each fitted on a different 20-point training set.



Bias vs Variance

Poly regression of order 10 coefficients

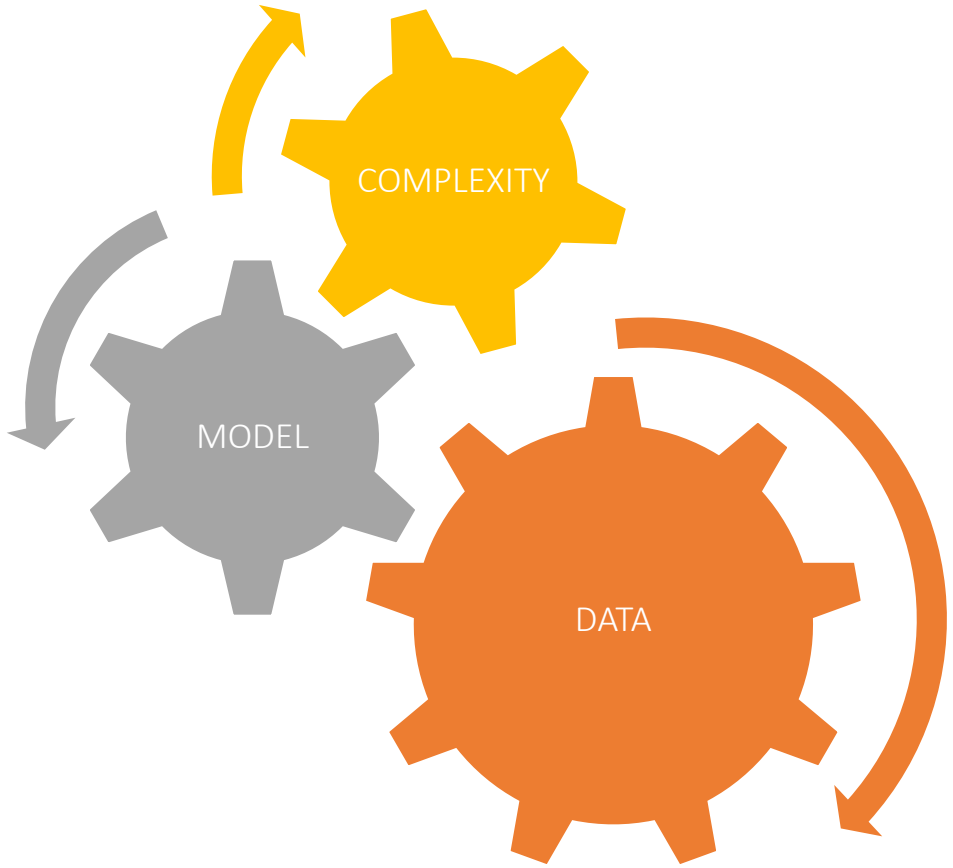




Linear regression coefficients

Regularized Regression

- Regularization: LASSO and Ridge
- Geometric Interpretation



Regularization: An Overview

- The idea of regularization revolves around modifying the loss function L; in particular, we add a regularization term that penalizes some specified properties of the model parameters.

$$L_{reg}(\beta) = L(\beta) + \lambda R(\beta),$$

  - where λ is a scalar that gives the weight (or importance) of the regularization term.

  - Fitting the model using the modified loss function L_{reg} would result in model parameters with desirable properties (specified by R).

LASSO Regression

- Since we wish to discourage extreme values in model parameter, we need to choose a regularization term that penalizes parameter magnitudes. For our loss function, we will again use MSE.
- Together our regularized loss function is:
- Note that $\sum_{j=1}^{J} |\beta_j|$ is the l_1 norm of the vector b

$$L_{LASSO}(\beta) = \frac{1}{n}\sum_{i=1}^{n} |y_i - \boldsymbol{\beta}^\top \boldsymbol{x}_i|^2 + \lambda \sum_{j=1}^{J} |\beta_j|. \qquad \sum_{j=1}^{J} |\beta_j| = \|\boldsymbol{\beta}\|_1$$

Ridge Regression

- Alternatively, we can choose a regularization term that penalizes the squares of the parameter magnitudes. Then, our regularized loss function is:

$$L_{Ridge}(\beta) = \frac{1}{n}\sum_{i=1}^{n} |y_i - \boldsymbol{\beta}^\top \boldsymbol{x}_i|^2 + \lambda \sum_{j=1}^{J} \beta_j^2.$$

- Note that $\sum_{j=1}^{J} |\beta_j|^2$ is the square of the $\boldsymbol{l_2}$ norm of the vector $\boldsymbol{\beta}$

$$\sum_{j=1}^{J} \beta_j^2 = \|\boldsymbol{\beta}\|_2^2$$

Choosing λ ?

- In both ridge and LASSO regression, larger choice of the **regularization parameter** λ, the more heavily penalize large values in ***b,***
- If l is close to zero, we **recover the MSE**, i.e. ridge and LASSO regression is just ordinary regression.
- If l is sufficiently large, the **MSE term will be insignificant,** and the regularization term will force $\boldsymbol{b}_{\text{ridge}}$ and $\boldsymbol{b}_{\text{LASSO}}$ to be close to zero.
- To avoid ad-hoc choices, we should select λ using cross-validation.

Ridge, LASSO - Computational complexity

- Solution to ridge regression:

$$\beta = (X^T X + \lambda I)^{-1} X^T Y$$

- Solution to LASSO regression:

  LASSO has **no conventional analytical solution**,

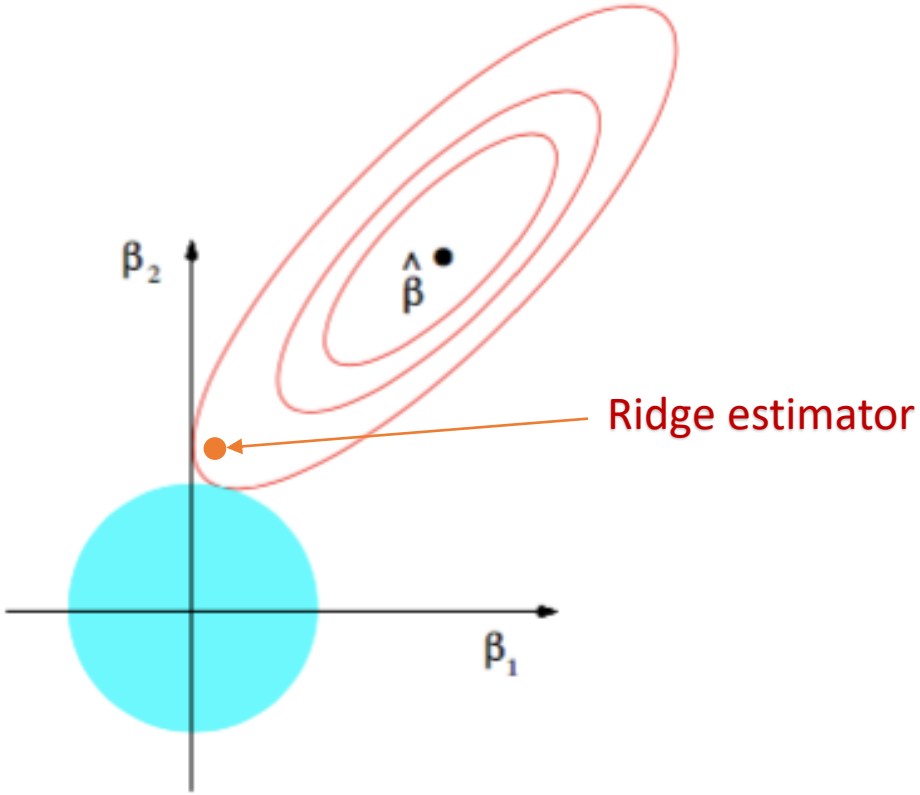
  L1 norm has no derivative at 0.

  Concept of subdifferential or subgradient to find a manageable expression.

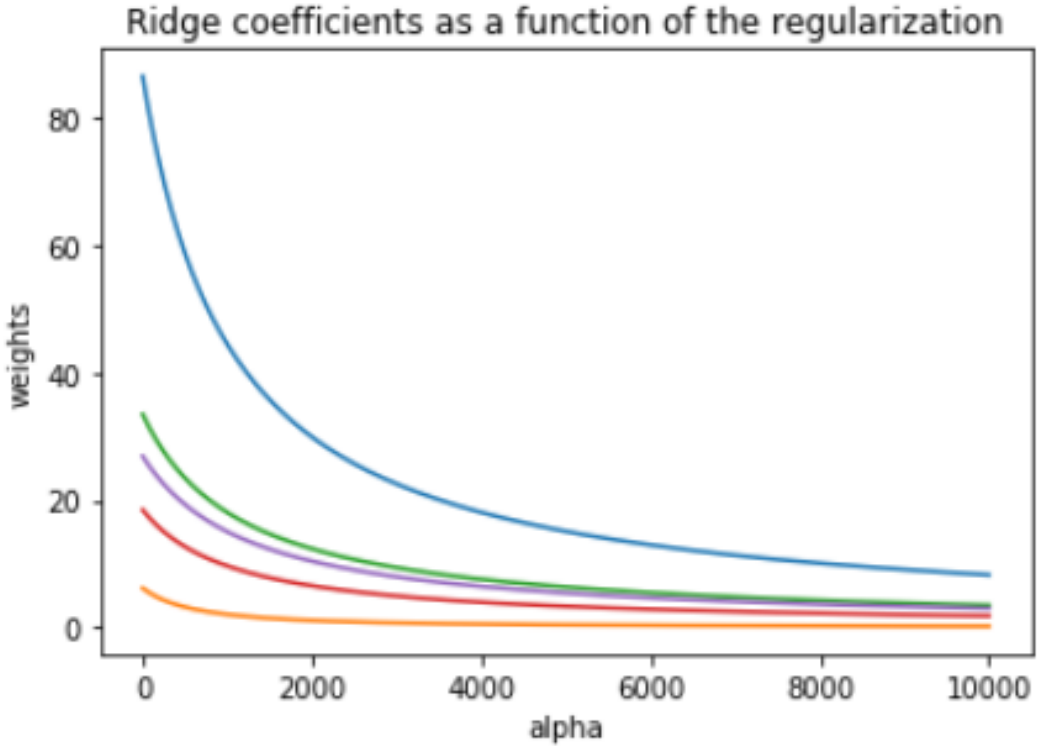
Regularization Parameter Selection

- The solution of the Ridge/Lasso regression involves three steps:
- Iterate over a range of values for l
- Find the minimum of the ridge/Lasso regression loss function (using the formula for ridge) and record the *MSE* **on the validation set**.
- Find the l that gives the smallest *MSE*

Ridge visualized..

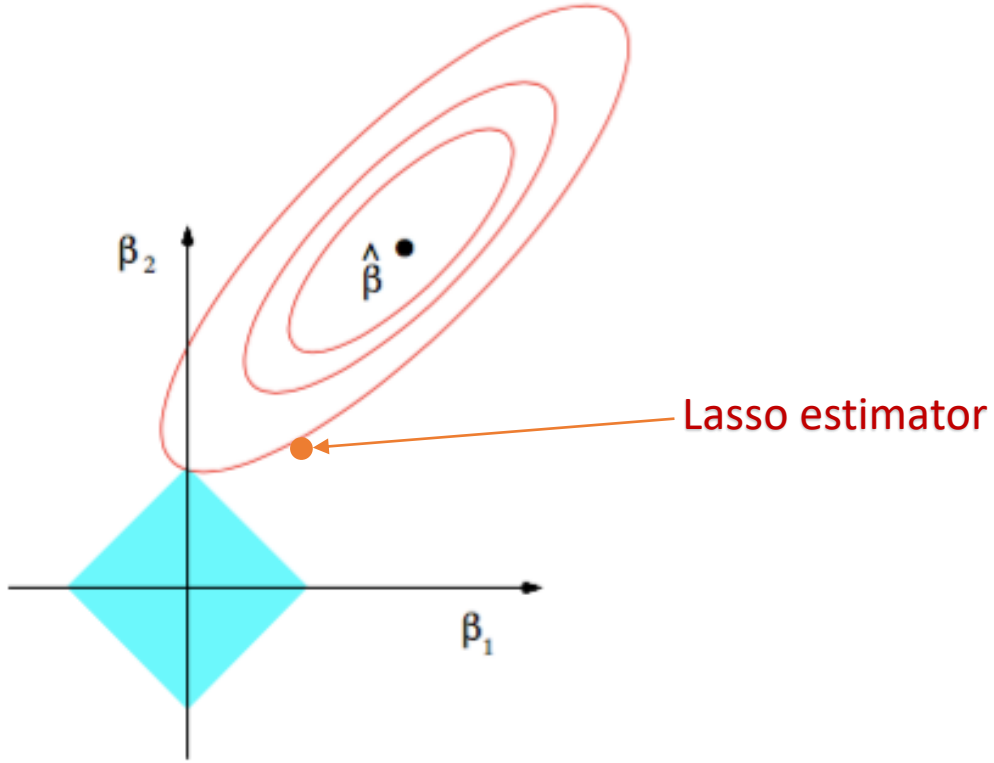


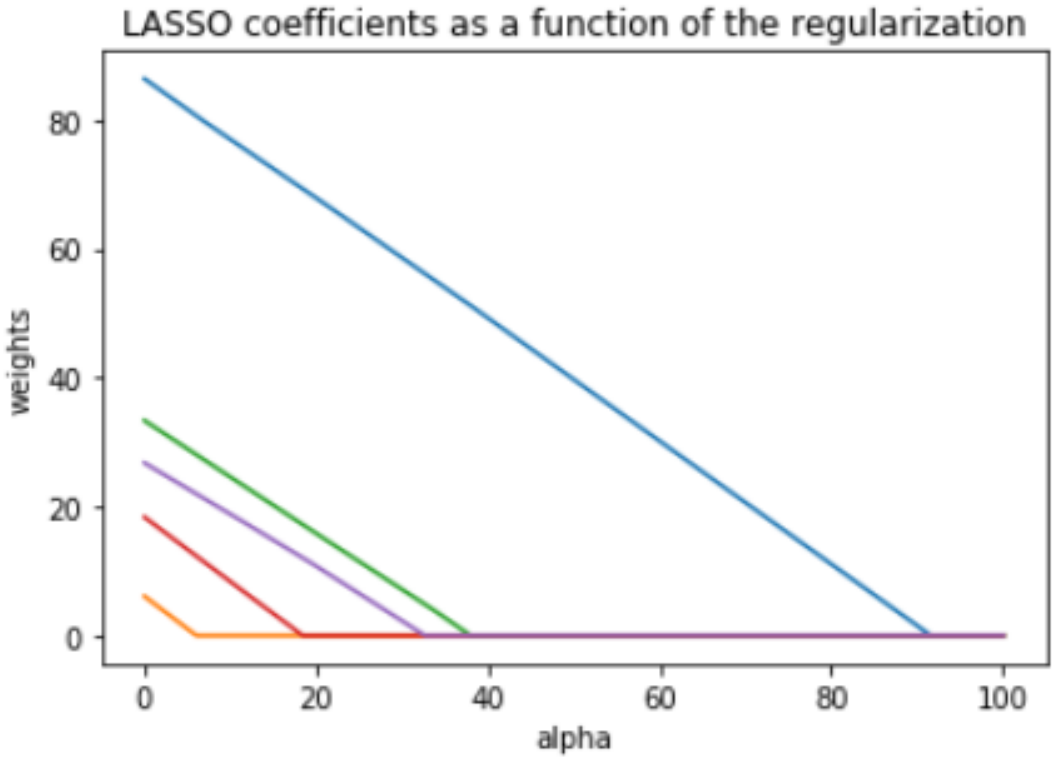
The ridge estimator is where the constraint and the loss intersect.



The values of the coefficients decrease as lambda increases, but they are not nullified.

LASSO visualized





The Lasso estimator tends to zero out parameters as the OLS loss can easily intersect with the constraint on one of the axis.

The values of the coefficients decrease as lambda increases and are nullified fast.

Evaluation metrics

- Provide a key role in the development of a model, as it provides insight to areas that require improvement.
- We'll be reviewing a number of model evaluation metrics, including:

1. Mean Absolute Error (MAE),
2. Mean Squared Error (MSE), and
3. Root Mean Squared Error (RMSE).

Evaluation metrics

- **Mean absolute error (MAE)** is the mean of the absolute value of the errors.
- It's just the average error.
- **Mean Squared Error (MSE**) is the mean of the squared error.
- It's more popular than Mean absolute error because the focus is geared more towards large errors.
- **Root Mean Squared Error (RMSE)** is the square root of the mean squared error. Root Mean Squared Error is most popular because it is interpretable in the same units as the response vector (or 'y' units) making it easy to relate its information.

$$MAE = \frac{1}{n}\sum_{j=1}^{n}|y_J - \hat{y}_j|$$

$$MAE = \frac{1}{n}\sum_{j=1}^{n}(y_j - \hat{y}_j)^2$$

$$RMSE = \sqrt{\frac{1}{n}\sum_{j=1}^{n}(y_j - \hat{y}_j)^2}$$

Evaluation Matrics

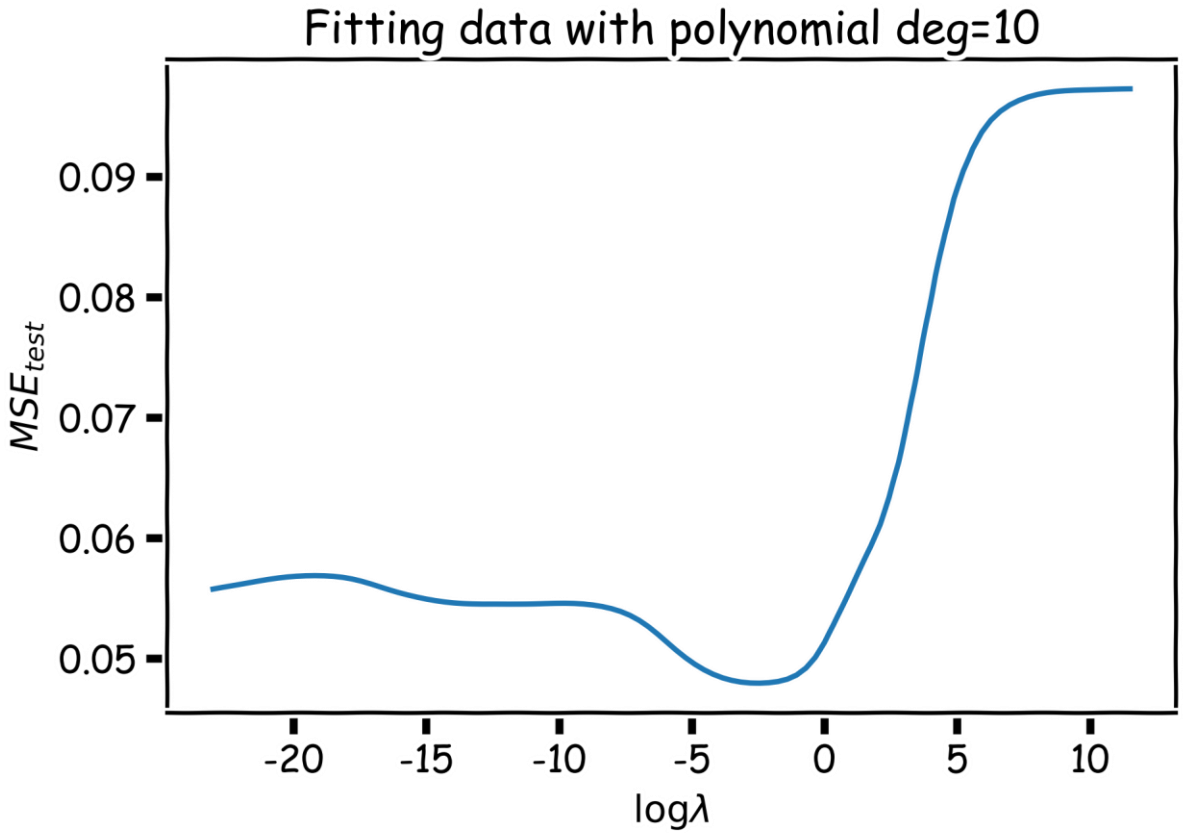
- **Relative Absolute Error (RAE),** also known as Residual sum of absolute errors, where y_bar is a mean value of y, takes the total absolute error and normalizes it by dividing by the total absolute error of the simple predictor.
- **Relative Squared Error (RSE)** is very similar to "Relative absolute error ", but is widely adopted by the data science community, as it is used for calculating R-squared.
- **R-squared** is not error, but is a popular metric for the accuracy of your model.
- The higher the R-squared, the better the model fits your data.

$$RAE = \frac{\sum_{j=1}^{n} |y_j - \hat{y}_j|}{\sum_{j=1}^{n} |y_j - \bar{y}_j|}$$

$$RSE = \frac{\sum_{j=1}^{n} (y_j - \hat{y}_j)^2}{\sum_{j=1}^{n} (y_j - \bar{y}_j)^2}$$

$$R^2 = 1 - RSE$$

Ridge regularization with **validation** only: step by step



Variable Selection as Regularization

- Since LASSO regression tend to produce zero estimates for a number of model parameters - we say that LASSO solutions are **sparse** - we consider LASSO to be a method for variable selection.

- Many prefer using LASSO for variable selection (as well as for suppressing extreme parameter values) rather than stepwise selection.

- **Question:** What are the pros and cons of the two approaches?

Performance Evaluation

The goal of regression is to build a model to accurately predict an unknown case.

After building the model, we have to perform regression evaluation.

Here, we'll discuss two types of evaluation approaches that can be used to achieve this goal.

These approaches are: train and test on the same dataset, and train/test split.

Performance Evaluation

- When considering evaluation models, we clearly want to choose the one that will give us the most accurate results.

- So, the question is**, how we can calculate the accuracy of our model?**

- In other words, how much can we trust this model (such as linear regression) for prediction of an unknown sample, using a given a dataset ?

- There are 2 approaches:
  - Train and Test on same
  - Train and Test split

Train and Test on the Same dataset.

- In this approach, you train the model on the entire dataset, then you test it using a portion of the same dataset.
- For instance, assume that we have 10 records in our dataset.
- We use the entire dataset for training, and we build a model using this training set.
- Now, we select a small portion of the dataset, such as row numbers 6 to 9, but without the labels.
- This set, is called a test set, which has the labels, but the labels are not used for prediction, and is used only as ground truth.
- The labels are called “Actual values” of the test set.

Train and Test on the Same dataset.

- Now, we pass the feature set of the testing portion to our built model, and predict the target values.
- Finally, we compare the predicted values by our model with the actual values in the test set.
- This indicates how accurate our model actually is.
- Let’s look at one of the simplest metrics to calculate the accuracy of our regression model.
- The error of the model is calculated as the average difference between the predicted and actual values for all the rows.

Train and Test on the Same dataset.

	ENGINESIZE	CYLINDERS	FUELCONSUMPTION_COMB	CO2EMISSION
0	2.0	4	8.5	196
1	2.4	4	9.6	221
2	1.5	4	5.9	136
3	3.5	6	11.1	255
4	3.5	6	10.6	244
5	3.5	6	10.0	230
6	3.5	6	10.1	232
7	3.7	6	11.1	255
8	3.7	6	11.6	267
9	2.4	4	9.2	**212**

Test

y

Train

Actual values

$$\text{Error} = \frac{1}{n}\sum_{J=1}^{n}\left|y_j - \hat{y}_j\right|$$

$$= \frac{(234-232)+(256-255)+(267-267)+\cdots}{4}$$

ŷ

	Prediction
6	234
7	256
8	267
9	210

Predicted values

Train and Test on the Same dataset.

- This evaluation approach would most likely have a **high "training accuracy" and a low "out-of-sample accuracy",** since the model knows all of the training data points from the training.

- **Training accuracy:** It is the percentage of correct predictions that the model makes when using the train dataset to verify the predictions.

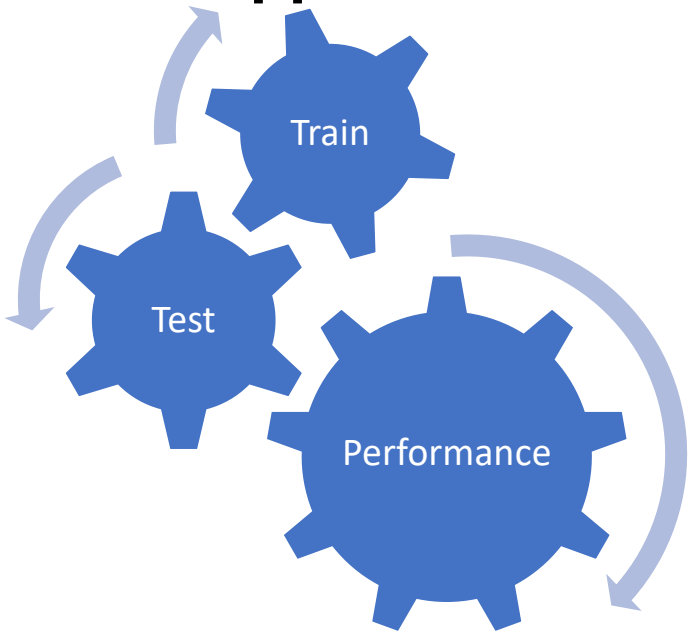
Train and Test on the Same dataset.

- A high training accuracy isn't necessarily a good thing.
- For instance, having a high training accuracy may result in an 'over-fit' of the data.
- This means that the model is overly trained to the dataset, which may capture noise and produce a non-generalized model.
- Doing a "train and test" on the same dataset will most likely have low out-of-sample accuracy due to the likelihood of being over-fit.

Any other alternative?

- So, how to improve out-of-sample accuracy?

- **By using another evaluation approach called "Train/Test Split.**



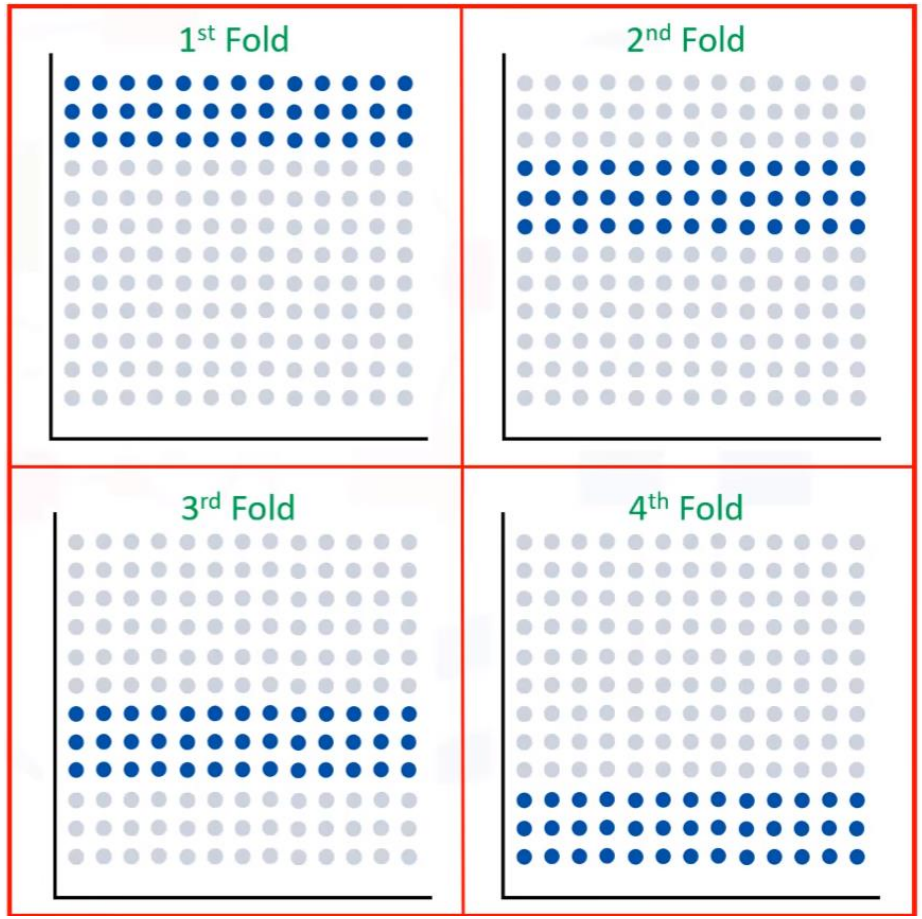
Train/Test Split

- In this approach, we select a portion of our dataset for training, for example, rows 0 to 5 and the rest is used for testing, for example, rows 6 to 9.
- The model is built on the training set.
- Then, the test feature set is passed to the model for prediction.
- And finally, the predicted values for the test set are compared with the actual values of the testing set.
- This will provide a more accurate evaluation on out-of-sample accuracy because the testing dataset is NOT part of the dataset that has been used to train the data.
- It is more realistic for real world problems.

Train/Test Split

- Since we know the outcome of each data point in this dataset, it is great to test with such database!
- Since this data has not been used to train the model, the model has no knowledge of the outcome, thus making it truly out-of-sample testing.
- However, please ensure that you train your model with the testing set afterwards, as you don’t want to lose potentially valuable data.
- The issue with train/test split is that it’s highly dependent on the datasets on which the data was trained and tested.
- Another evaluation model, called "K-Fold Cross-validation," resolves most of these issues.

K-Fold method



- The entire dataset is represented by the points in the image at the top left.
- If we have k=4 folds, then we split up this dataset as shown here.
- In the first fold, for example, we use the first 25 percent of the dataset for testing, and the rest for training.

K-Fold method

- Then, in the next round (or in the second fold), the second 25 percent of the dataset is used for testing and the rest for training the model.

- Again the accuracy of the model is calculated.

Hands On