Regression (cont.)

Session-4/5

SUMMARY of PREVIOUS CLASS

- Bias, Variance, Trade-off
- Underfitting, Overfitting:
  - Reasons
  - Outcomes
  - Solutions
- Polynomial Regression
  - Overfitting analysis
  - Order selection

SUMMARY of PREVIOUS CLASS

Polinomial Regression Model Building

1- Import Library

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import PolynomialFeatures
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
```

2- Data Preparation / Train-Test Split

```
df = pd.read_csv("Advertising.csv")
X = df.drop(columns ="sales") #df[["TV", "radio", "newspaper"]]
y = df["sales"]

polyFeature = PolynomialFeatures(degree = 3, include_bias=False)
Xnew=polyFeature.fit_transform(X) # Obtain new X for splitting
X_train, X_test, y_train, y_test = train_test_split(Xnew, y, test_size = 0.3, random_state =42)
```

3- Model Building and Fitting

```
model = LinearRegression()
model.fit(X_train, y_train)
```

4- Prediction

```
y_pred_train = model.predict(X_train)
y_pred_test = model.predict(X_test)
```

5- Evaluation

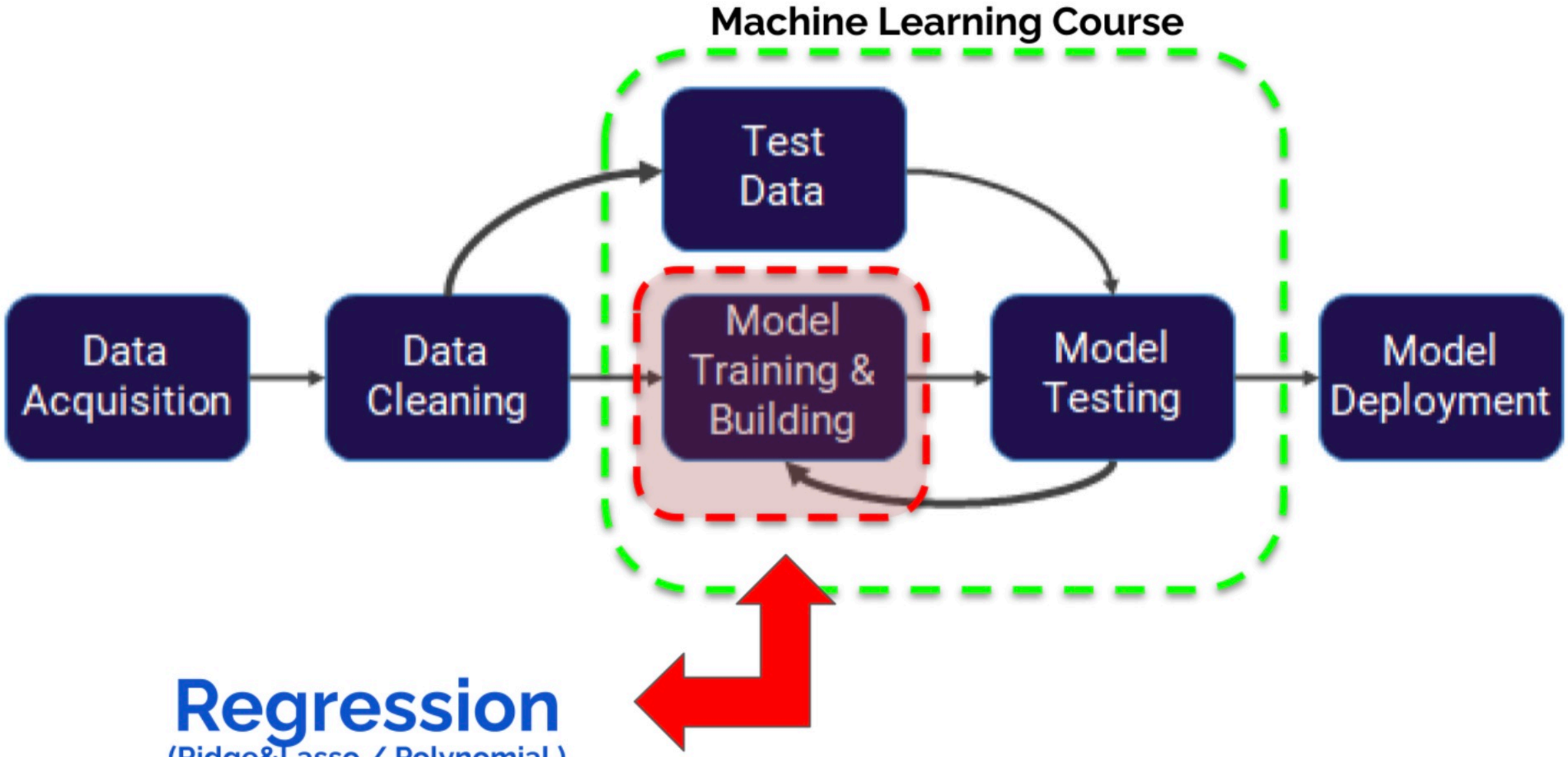
```
#Model Performance Evaluation for Train Data
mae = mean_absolute_error(y_train, y_pred_train)
mse = mean_squared_error(y_train, y_pred_train)
rmse = np.sqrt(mean_squared_error(y_train, y_pred_train))
R2_score = r2_score(y_train, y_pred_train)
print("Model Performance Evaluation for Train Data:")
print("--------------------------")
print(f"R2_score \t: {R2_score}")
print(f"MAE \t\t: {mae}")
print(f"MSE \t\t: {mse}")
print(f"RMSE \t\t: {rmse}")
```

```
#Model Performance Evaluation for Test Data
mae = mean_absolute_error(y_test, y_pred_test)
mse = mean_squared_error(y_test, y_pred_test)
rmse = np.sqrt(mean_squared_error(y_test, y_pred_test))
R2_score = r2_score(y_test, y_pred_test)
print("Model Performance Evaluation for Test Data:")
print("--------------------------")
print(f"R2_score \t: {R2_score}")
print(f"MAE \t\t: {mae}")
print(f"MSE \t\t: {mse}")
print(f"RMSE \t\t: {rmse}")
```

Regularization

- Overview
- Multicollinearity
- Feature Scaling
- Ridge Regression Theory
- Lasso Regression Theory
- Elastic-Net
- Cross-Validation and Grid Search

Where are we?


Machine Learning Course
Test Data
Data Acquisition
Data Cleaning
Model Training & Building
Model Testing
Model Deployment
Regression
(Ridge&Lasso / Polynomial)

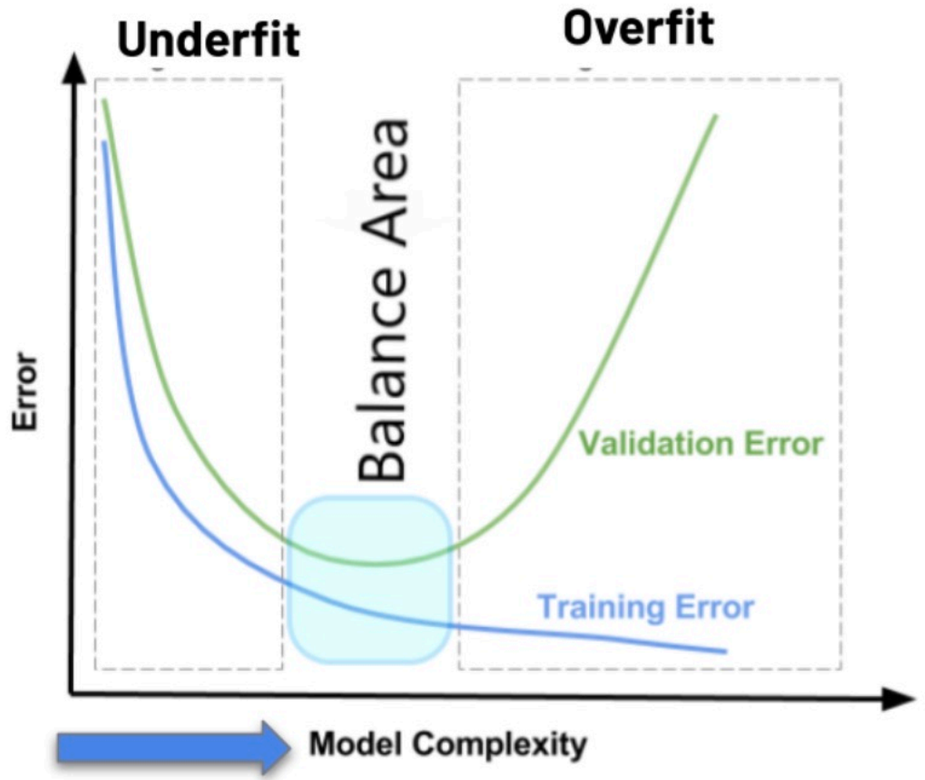
Regularization

- *Overview*

Regularization
HOW TO DEAL WITH

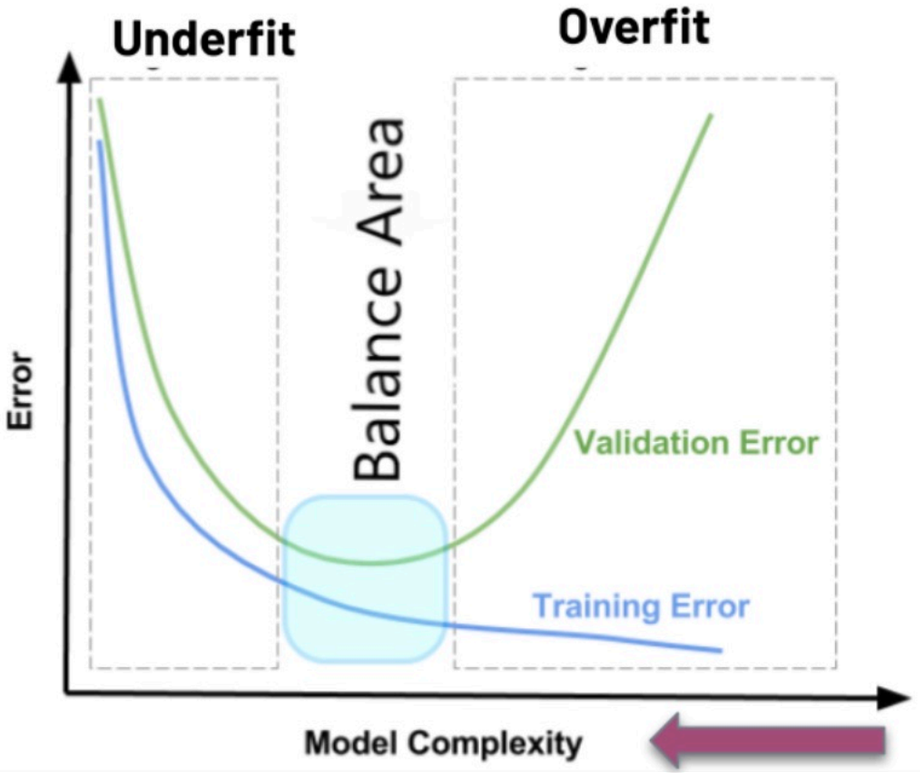
UNDERFITTING ?

- Find a **more complex** model (Polynomial Reg.)



OVERFITTING ?

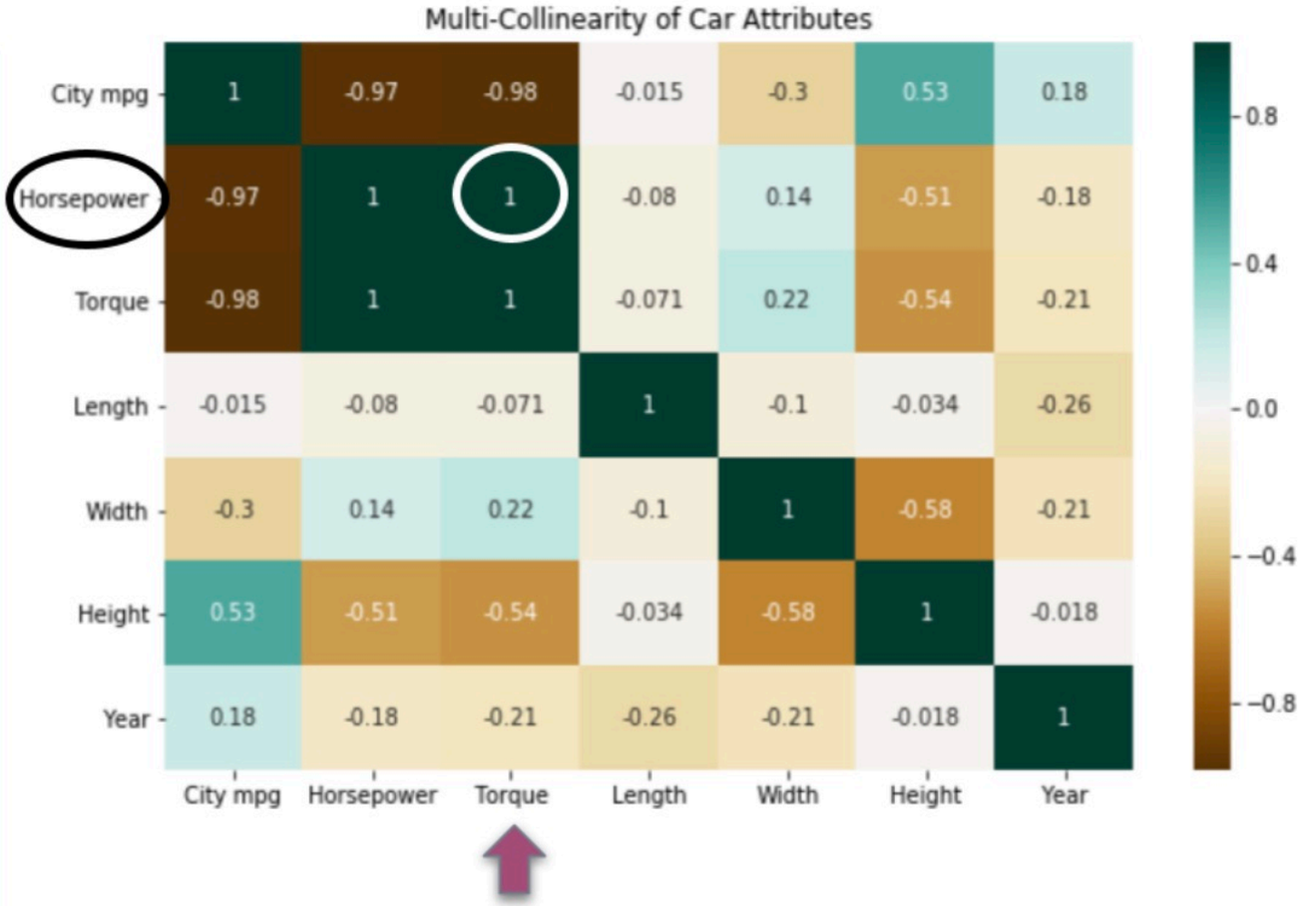
- Decrease the number of parameters
- More training data / Cross Validation
- **Regularization (Lasso&Ridge)**



Regularization

In Linear regression, if there is a correlation between independent variables, **multicollinearity** occurs.

With this problem, the feature coefficients may be **unstable**, there may be different signs and coefficient for the model features than they should be, **the feature variances may be larger** than they are, very small changes in the data may show huge differentiations in the feature coefficients.

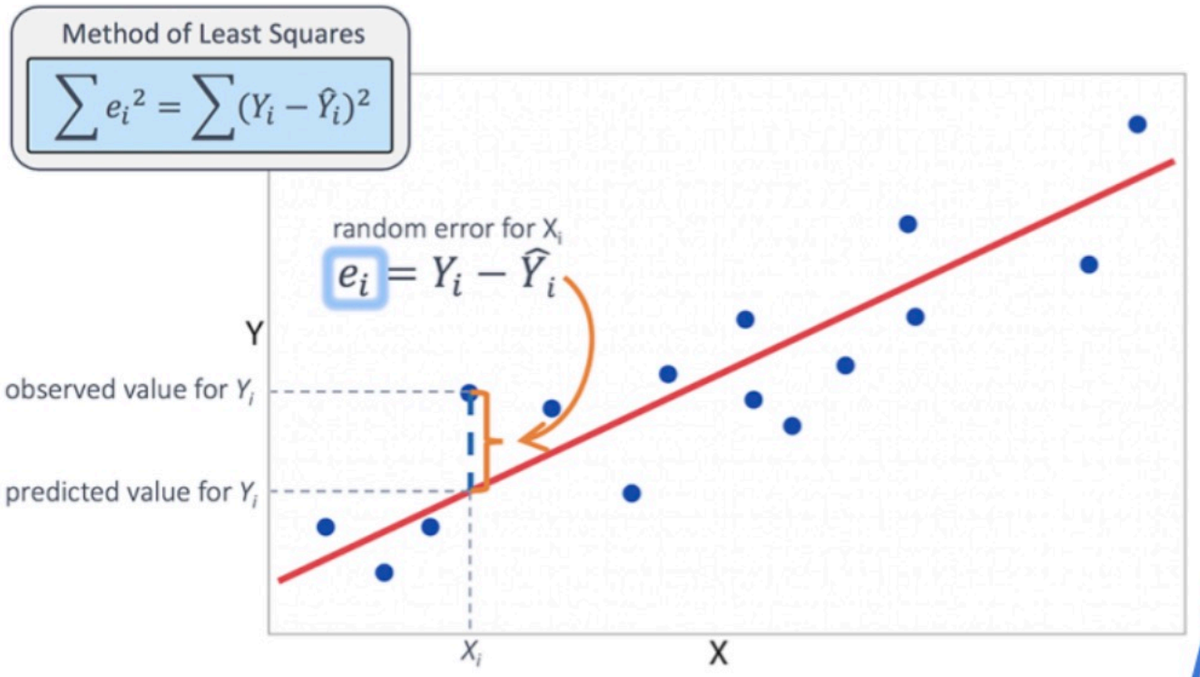


Need

Regularization

Regularization

Ordinary Least Squares (OLS)



Find the best fit line by minimizing the squares of errors.

Regularization

$$\sum e_i^2 = \sum (Y_i - \hat{Y}_i)^2$$

+

Penalty

Works by penalizing the OLS parameters.

Regularization

Approximation to y_i is given by: $\hat{y}_i = \hat{\beta}_0 + \hat{\beta}_1 x_i$

The residual sum: $\sum_{i=1}^{m} \hat{e}_i^{\,2} = \sum_{i=1}^{m} (y_i - \hat{y}_i)^2 = \sum_{i=1}^{m} (y_i - \hat{\beta}_0 - \hat{\beta}_1 x_i)^2$

Ridge Regression (L2) + **Penalty** **LASSO Regression (L1)**

Elastic-Net

$$\underbrace{\sum_{i=1}^{n} (y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij})^2}_{\text{Residual Sum (The Least Squares)}} + \underbrace{\lambda \sum_{j=1}^{p} \beta_j^{\,2}}_{\text{Ridge Reg. Penalty}}$$

$$\underbrace{\sum_{i=1}^{n} (y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij})^2}_{\text{Residual Sum (The Least Squares)}} + \underbrace{\lambda \sum_{j=1}^{p} |\beta_j|}_{\text{LASSO Reg. Penalty}}$$

Ridge Regression

Approximation to y_i is given by: $\hat{y}_i = \hat{\beta}_0 + \hat{\beta}_1 x_i$

The residual sum: $\sum_{i=1}^{m} \hat{e}_i^{\,2} = \sum_{i=1}^{m} (y_i - \hat{y}_i)^2 = \sum_{i=1}^{m} (y_i - \hat{\beta}_0 - \hat{\beta}_1 x_i)^2$

Ridge Regression (L2)

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

Residual Sum (The Least Squares)

Ridge Reg. Penalty

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

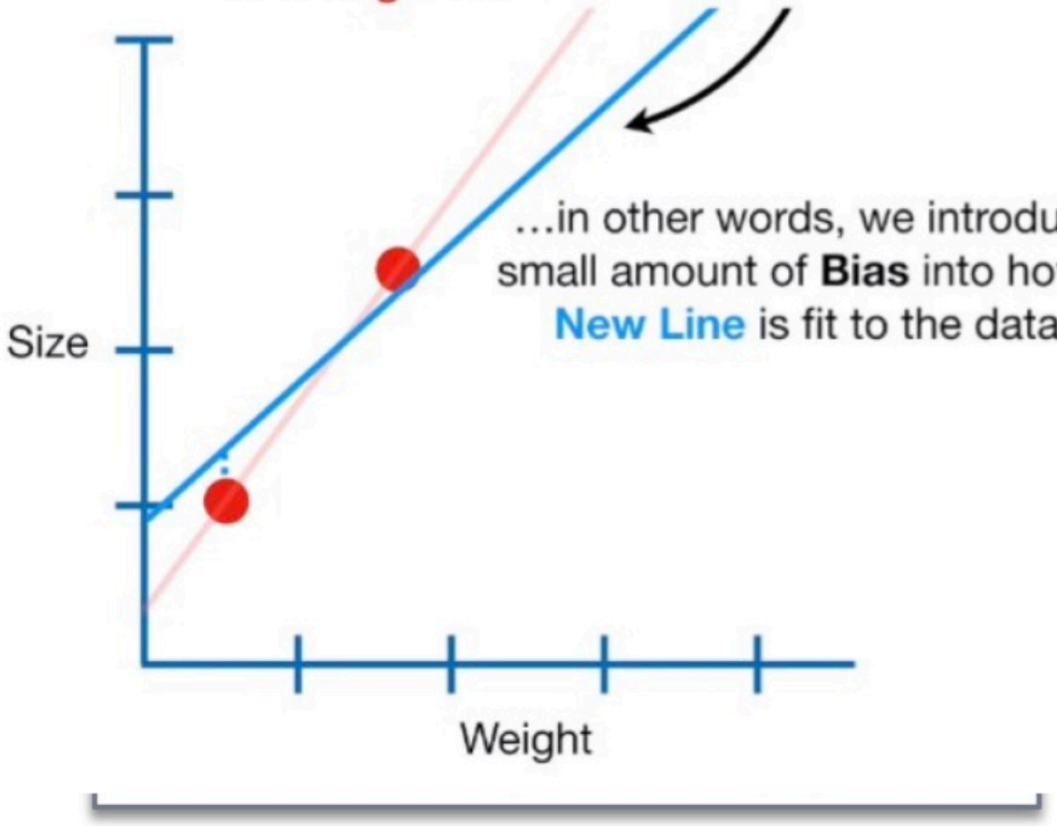
p **:** number of features

β **:** coefficient of each feature (slope)

λ **: Lambda (hyperparameter)**

Ridge Regression

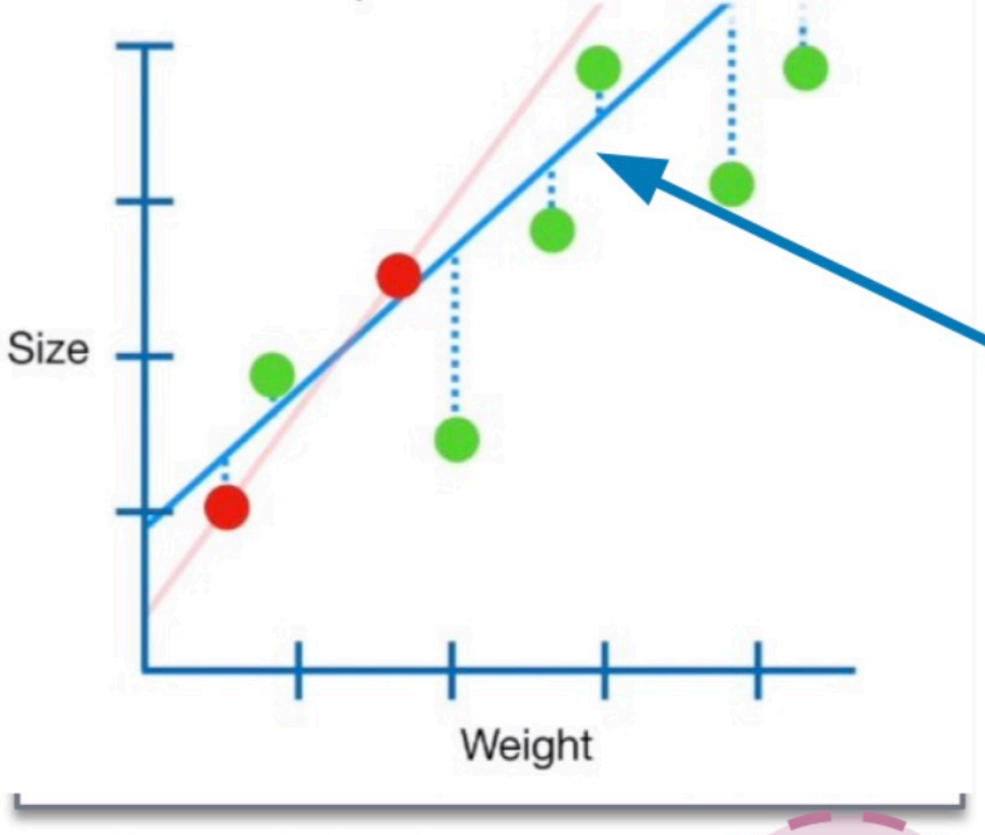
The main idea behind **Ridge Regression** is to find a **New Line** that doesn't fit the **Training Data** as well…



…in other words, we introduce a small amount of **Bias** into how the **New Line** is fit to the data…

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

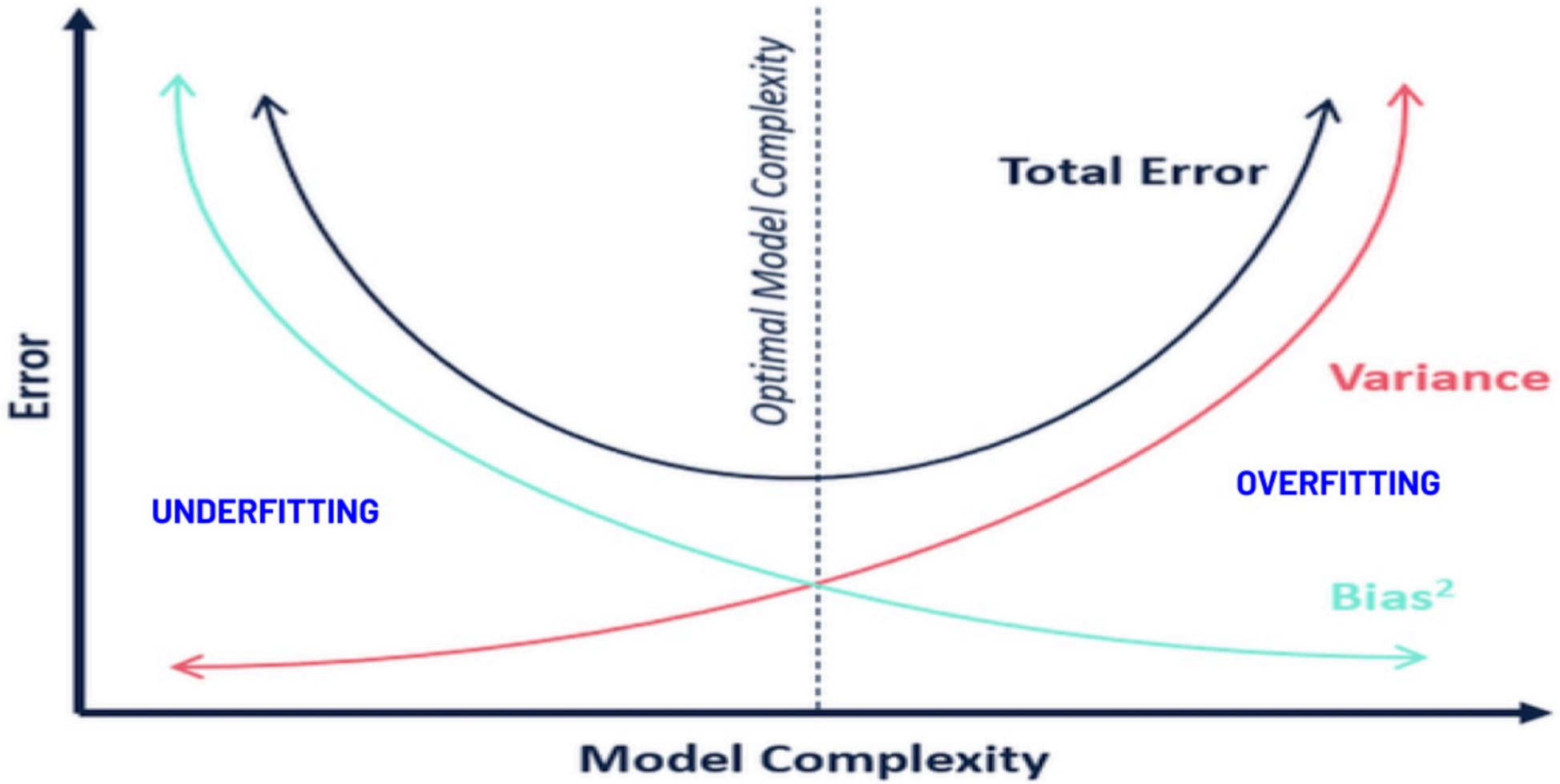
In other words, by starting with a slightly worse fit, **Ridge Regression** can provide better long term predictions.



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

Pic.credit : Statquest

Ridge Regression



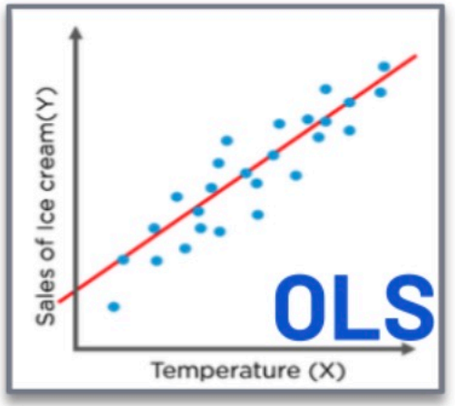
Ridge Regression

λ : Lambda (tuning effects)

$\lambda \to 0$

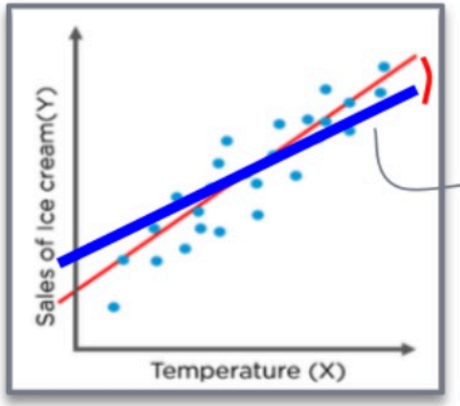
$$\underbrace{\sum_{i=1}^{n}(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij})^2}_{\text{OLS}} + \lambda \sum_{j=1}^{p}\beta_j^2$$

$\lambda \to 0$



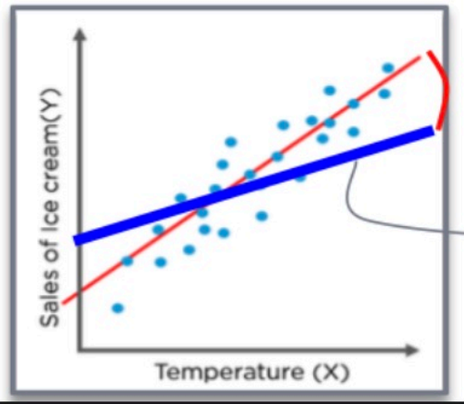
$\lambda \to 1$

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



$\lambda \to \dots$

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

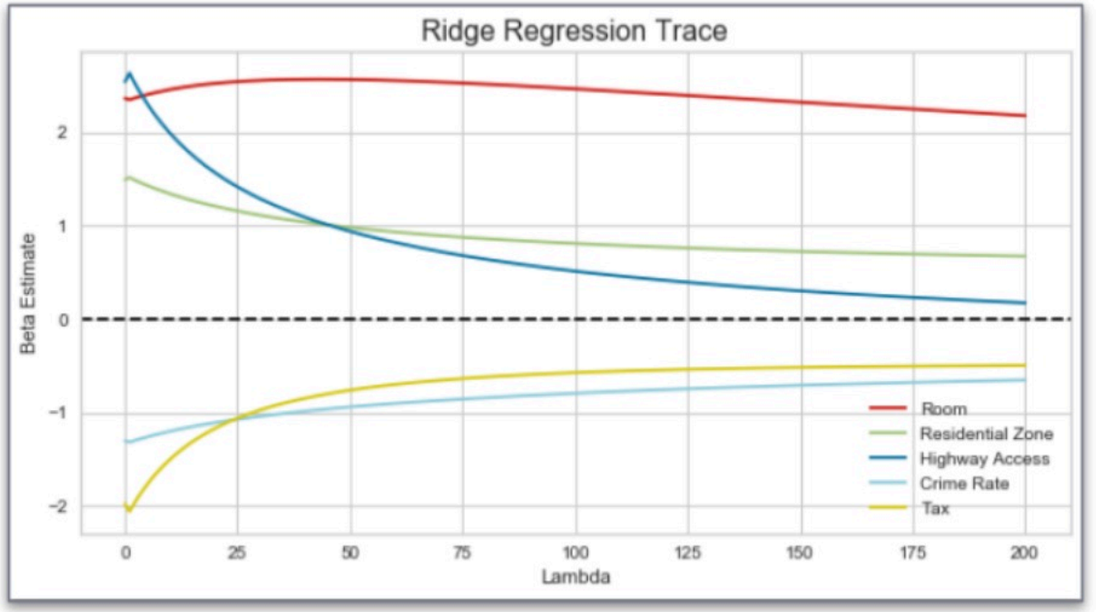


New line

Ridge Regression

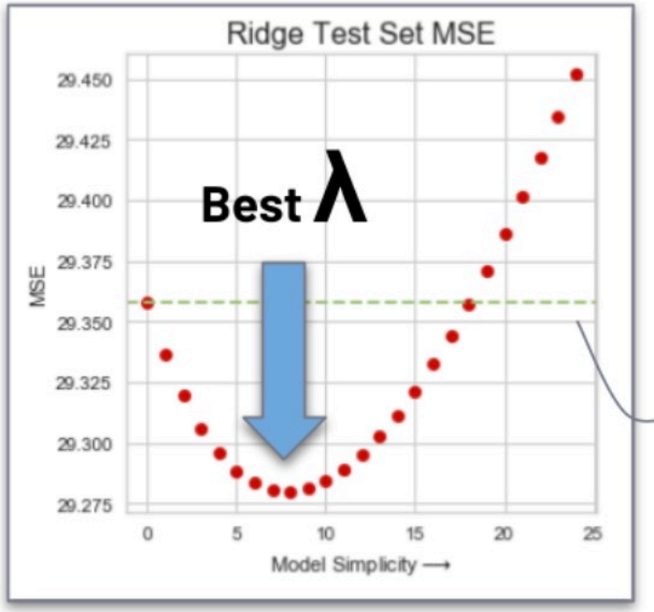
How to find the best λ (Lambda) (hyper-parameter tuning)

Need to *iteration* of **λ (Lambda)** values and evaluate prediction performances with **error metrics**.



House Price

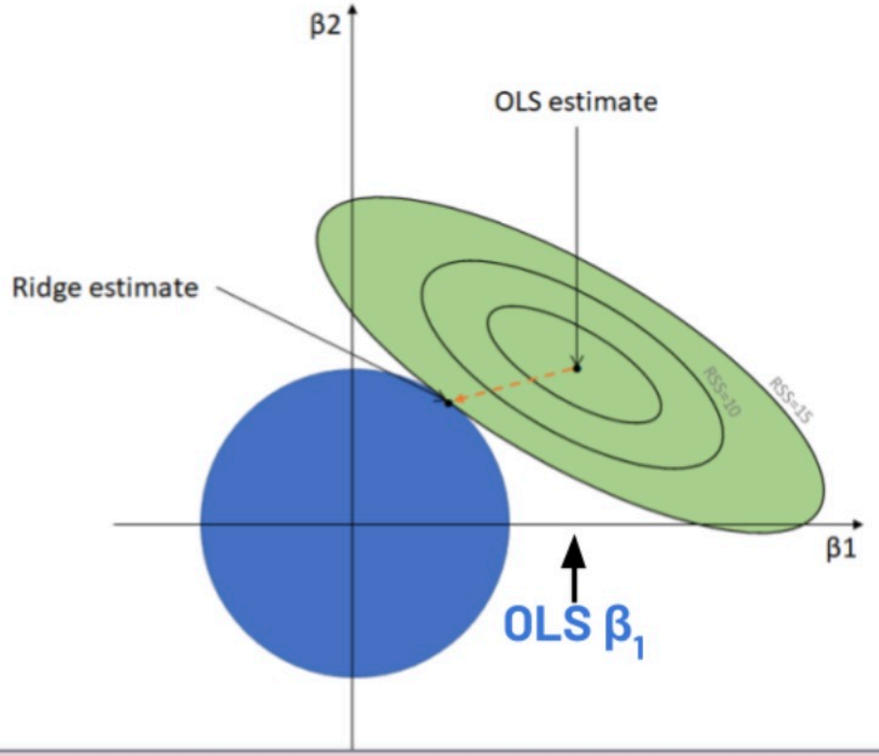
Sample

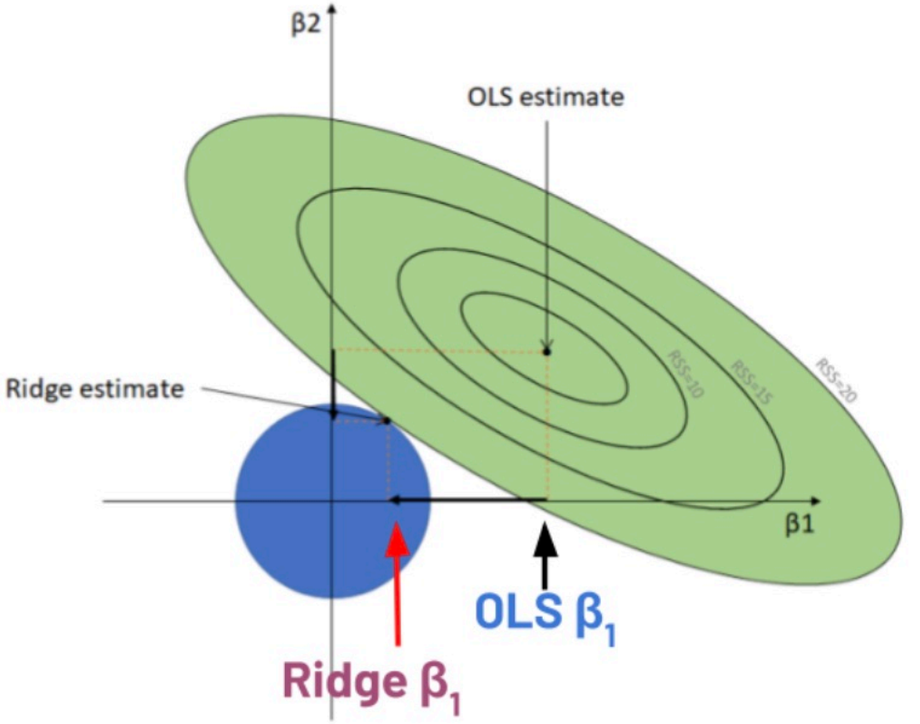


OLSs' MSE

Attention: We will use **"alpha"** as a hyperparameter name instead of lambda (λ) in python. Because lambda represents another function in python.

Ridge Regression





Each contour is a connection of spots where the RSS is the same, centered with the OLS estimate where the RSS is the lowest (where it best fits the training set)

How ridge regression works is how we tune the size of the circle with λ. The key point is that β's change at a different level.

RSS : Residual Sum of Squares

Ridge Regression

Pros & Cons

Pros:

– Good for few data sets
(if features (p) > data (n))

– Good for multicollinearity

– Biased but smaller variance and smaller regression metrics

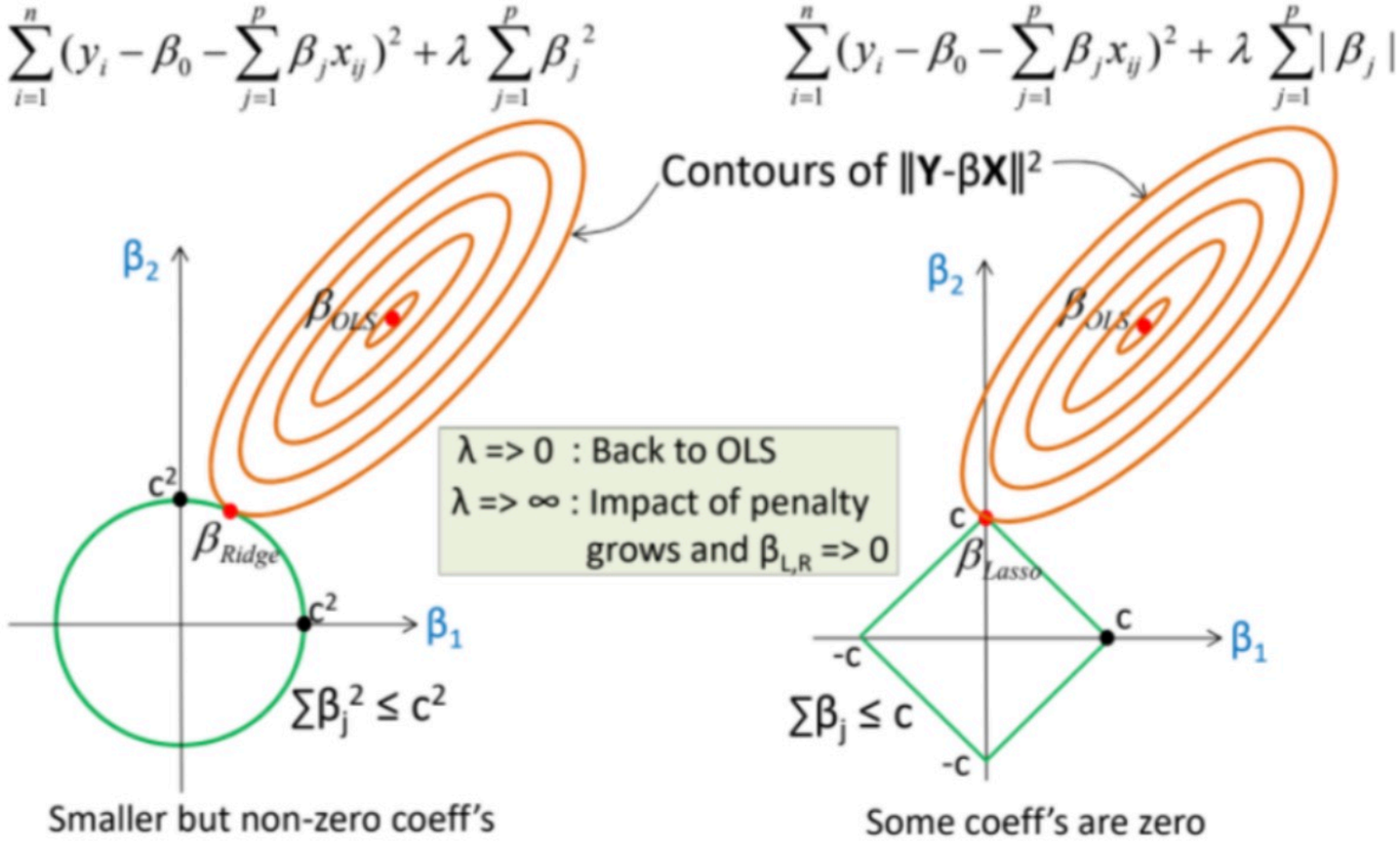
Cons:

– Not suitable for variable selection or for assessing the relative importance of each predictor.

LASSO Regression

LASSO regression has very many **similarities with Ridge Regression**.

Exceptions,



Lasso regression,

- takes the absolute value of the coefficients **(| β |)** instead of squaring (β^2) them, so some attributes are **completely ignored**.
- plays an important role not only in reducing overfit learning, but also in **feature selection**.

LASSO Regression

Pros & Cons

Pros:

– Allows (features (p) > data (n))

– Good for variable selection

– Biased but smaller variance and smaller regression metrics

Cons:

– In case of **highly correlated predictors**, Lasso tends to **pick only one** of them, shrinking the coefficients for others to 0

(cannot do grouped selection, selects just one variable)

Ridge & LASSO Regression

Summary:

- **OLS** simply finds the **best unbiased linear** fit for given data,
- Ridge & LASSO regressions give a **bias to loss function (using feature coefficients)**,
- Ridge & LASSO are **good for overfitting** and useful for **multicollinearity**
- MAE, MSE, RMSE or R^2 can be used to find the **best lambda**

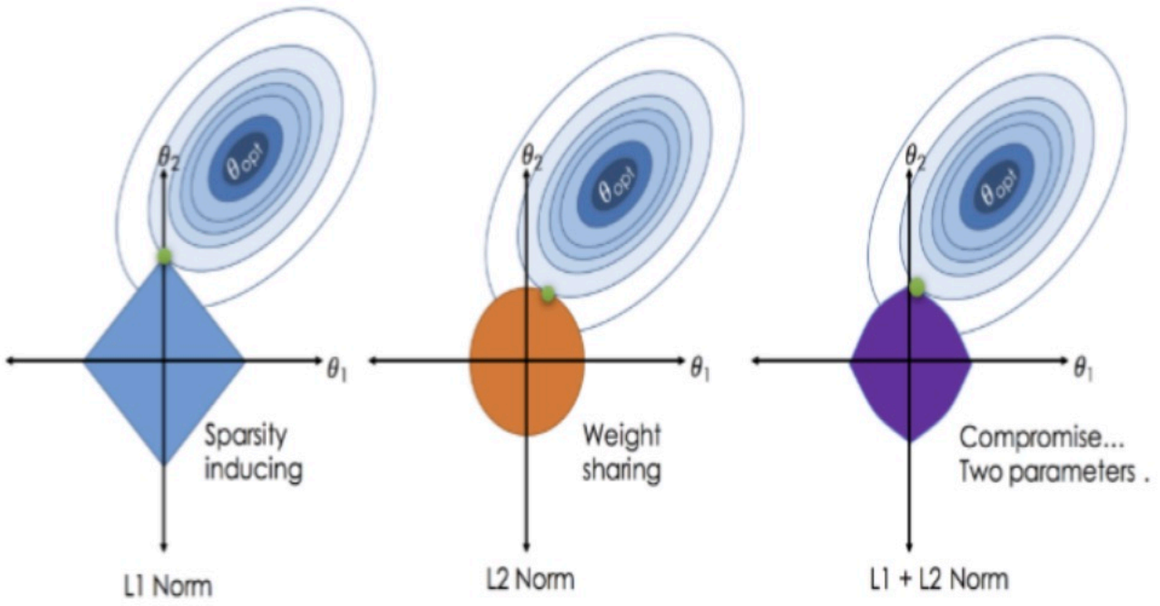
Ridge:

Good for ***group selection***, but not good for eliminating predictors

LASSO:

Good for ***eliminating predictors,*** but not good for grouped selection

Elastic-NET



L1 Penalty: $R(w) := \frac{1}{2}\sum_{i=1}^{n}|w_i|$

L2 Penalty: $R(w) := \frac{1}{2}\sum_{i=1}^{n} w_i^2$

Elastic-Net Penalty:

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

A convex combination of L1 and L2 Penalty.

Elastic-Net regularization term is a straightforward blend of both Ridge and Lasso's regularization terms,

You can handle the blend proportion alpha. At the point when l1_ratio= 1, Elastic Net behaves the same as Lasso Regression, and when l1_ratio= 0, it is identical to Ridge Regression.

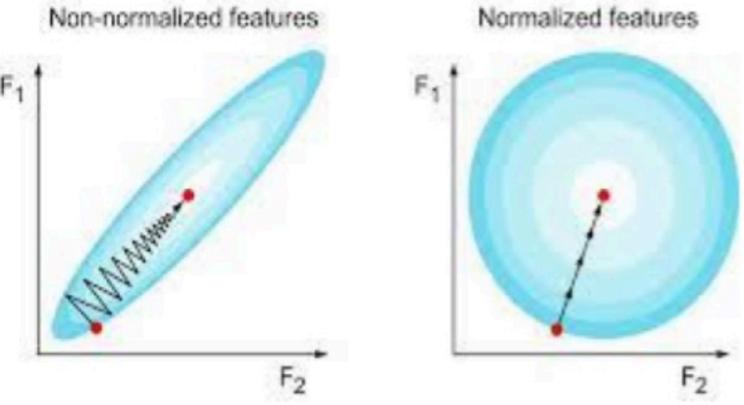
Regularization

- *Feature Scaling*
- *Cross Validation and Grid Search*

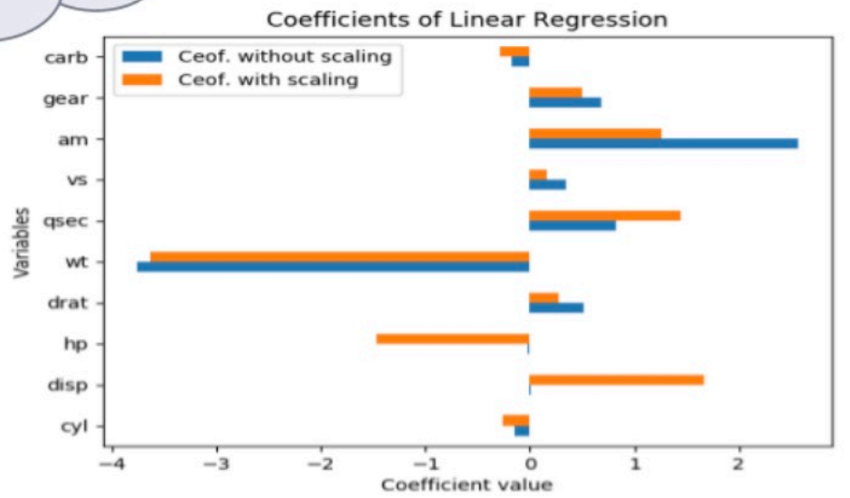
Feature Scaling

Gradient Descent

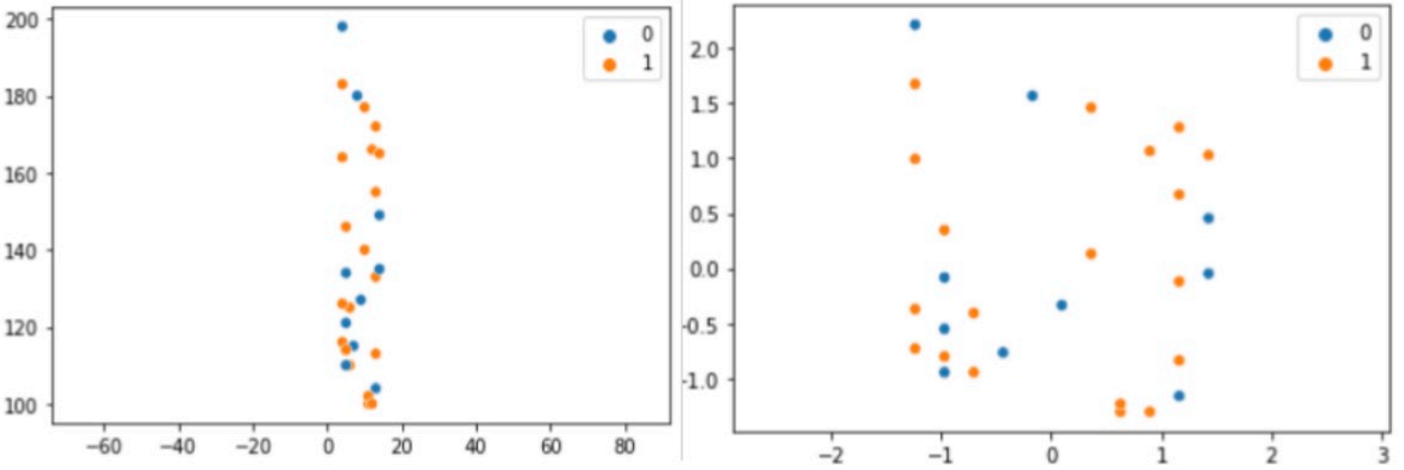
Gradient descent with and without feature scaling



Model Coefficients



Algorithms Rely on Distance Metrics



Feature Scaling

- If you train your model with scaled data, you **must scale** the unseen input before prediction.
- You can compare scaled coefficients one and other easily. However, you can't compare one of these with an unscaled one.
- If you are not sure about scale will perform well or not, **chose scaling.**

Feature Scaling

Standardization

- To achieve a data which mean = 0 and deviation = 1
- Z-score normalization

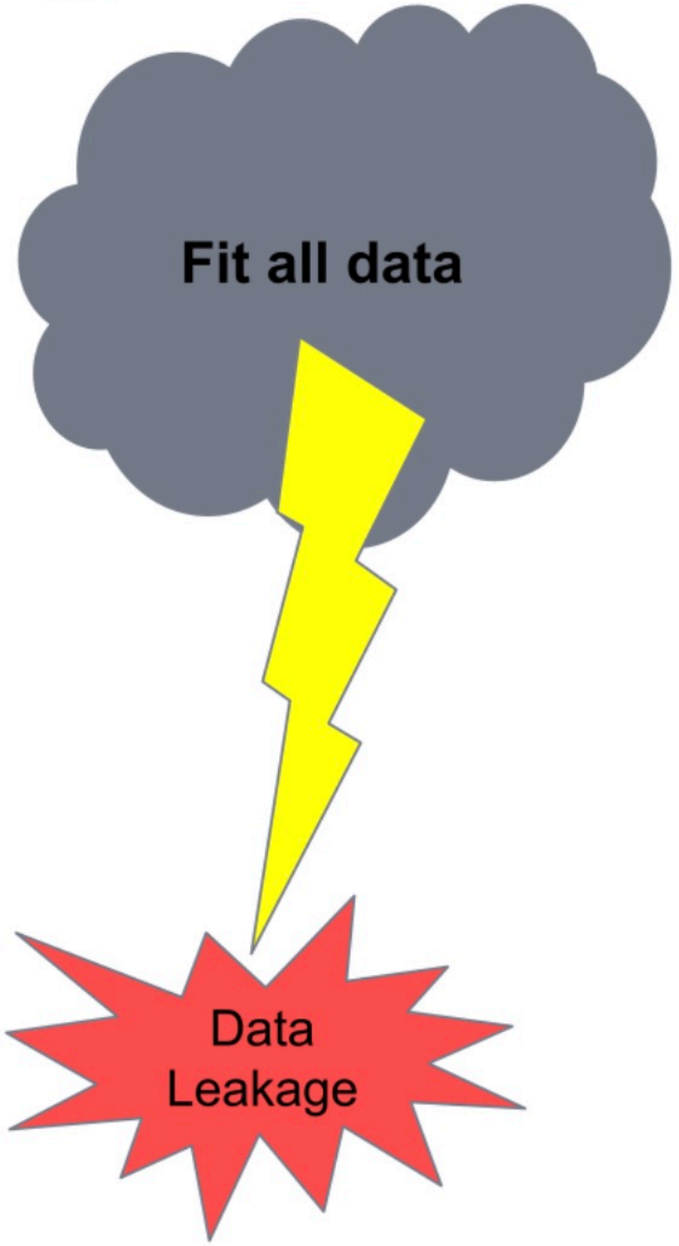
$$X_{changed} = \frac{X - \mu}{\sigma}$$

Normalization

- Shrink all data values between 0-1

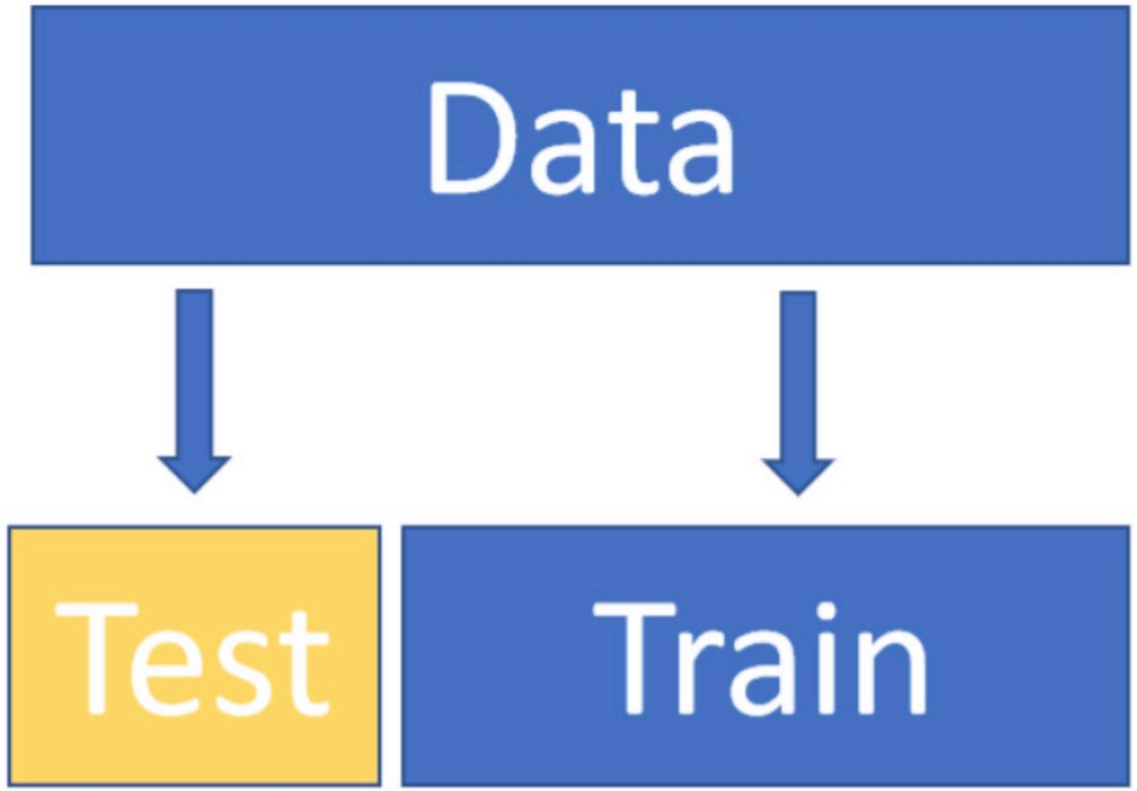
$$X_{changed} = \frac{X - X_{min}}{X_{max} - X_{min}}$$

Feature Scaling



Cross Validation and Grid Search

Why we are performing train test split ?



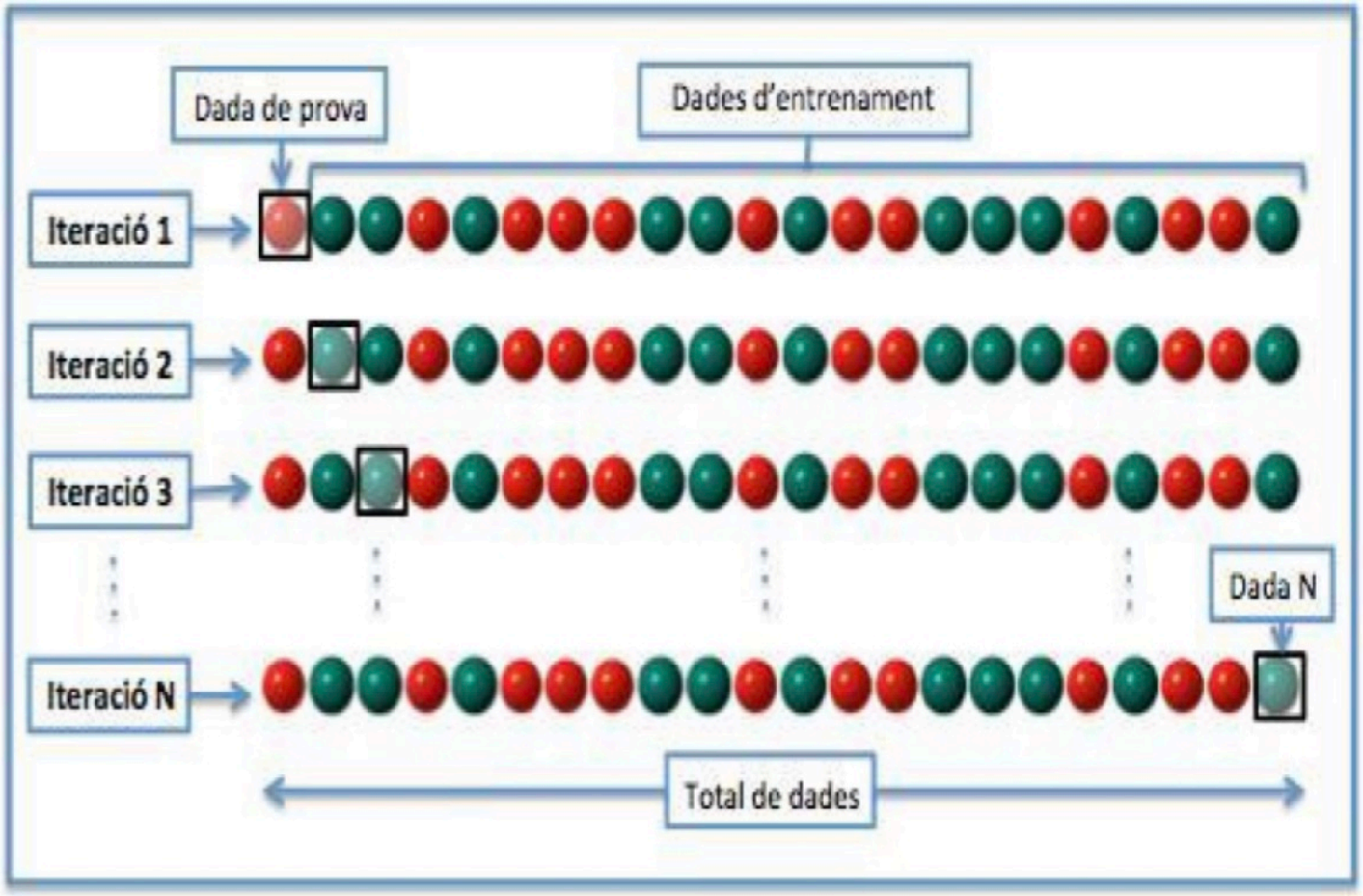
Cross Validation and Grid Search

K-fold cross validation

iteration 1 accuracy = %92
iteration 2 accuracy = %95
iteration 3 accuracy = %89
.
.
.
iteration k accuracy = %90

Final Accuracy = Mean of all iterations

Cross Validation and Grid Search



LOO cross validation

iteration 1 accuracy = %92
iteration 2 accuracy = %95
iteration 3 accuracy = %89

.
.
.

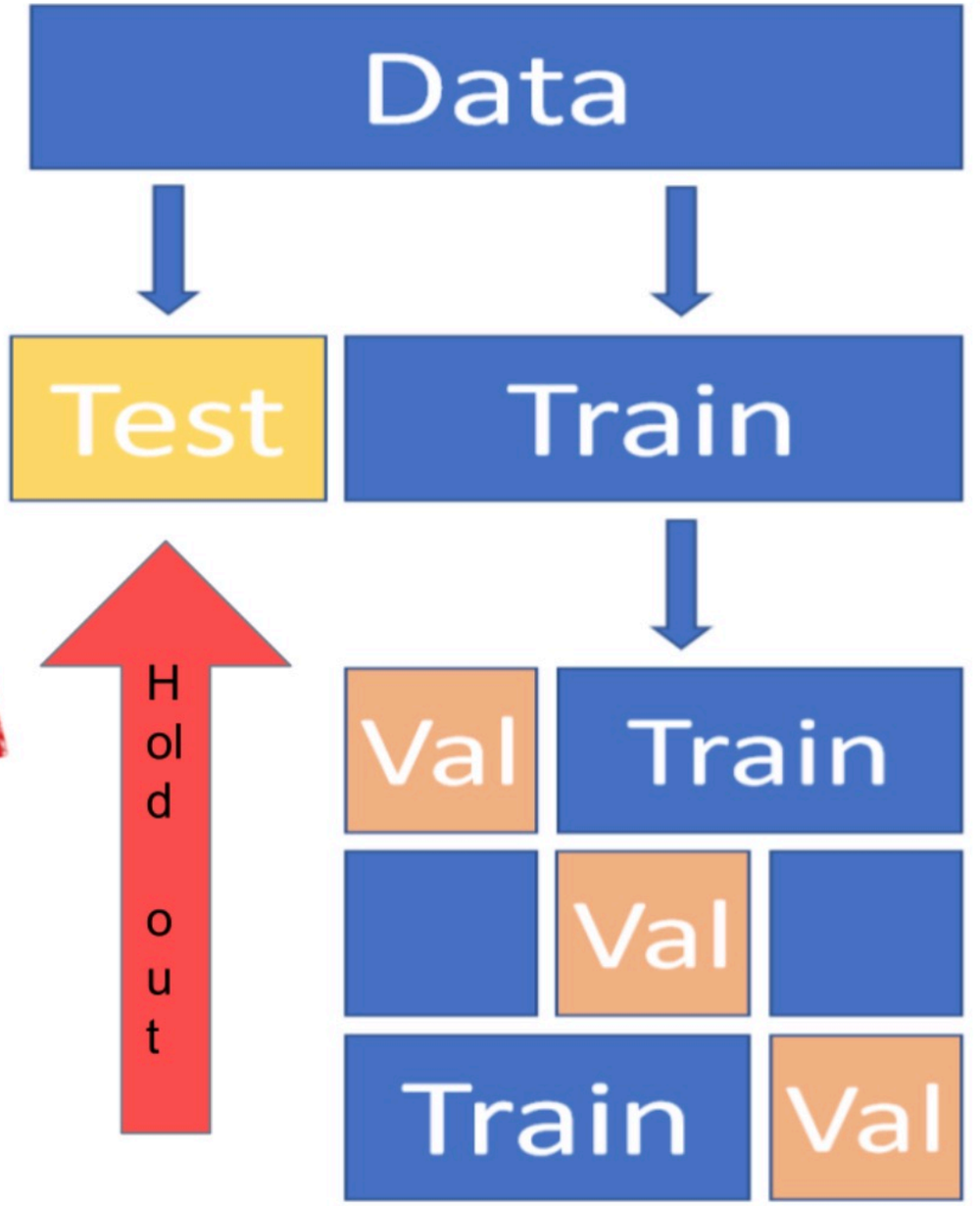
Iteration NOO accuracy = %90

Final Accuracy = Mean of all iterations

LOO : Leave One Out
NOO : Number of observations

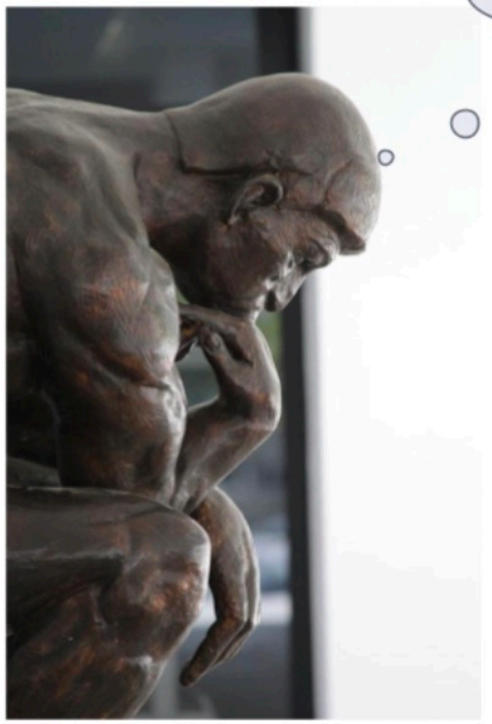
Cross Validation and Grid Search

How can we actually sure performance of our model ?



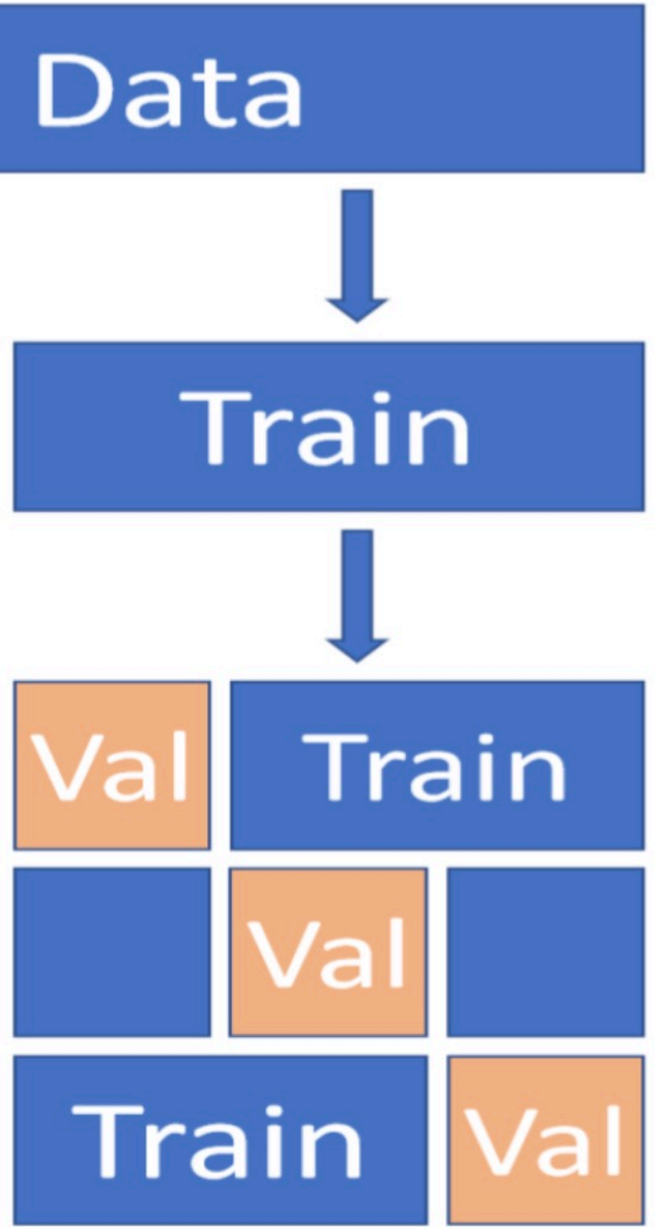
Cross Validation and Grid Search

What about hyperparameters?



Grid Search

GridSearchCV



What is regularization?
Why is it useful?