Applied Machine Learning

Amin Nazari

Spring 2025

Chapter 1

Supervised Learning

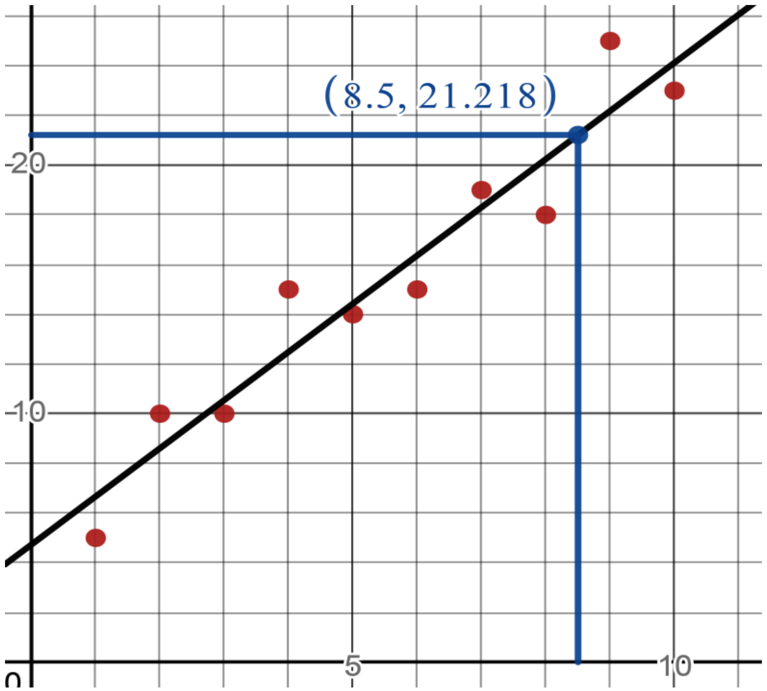
Linear Regression

Outline

- Linear Regression
- Correlation matrix
- Loss function and Gradient descent
- Learning Rate
- Non-linear regression
- Overfitting
  - Train-test split
  - Regularization

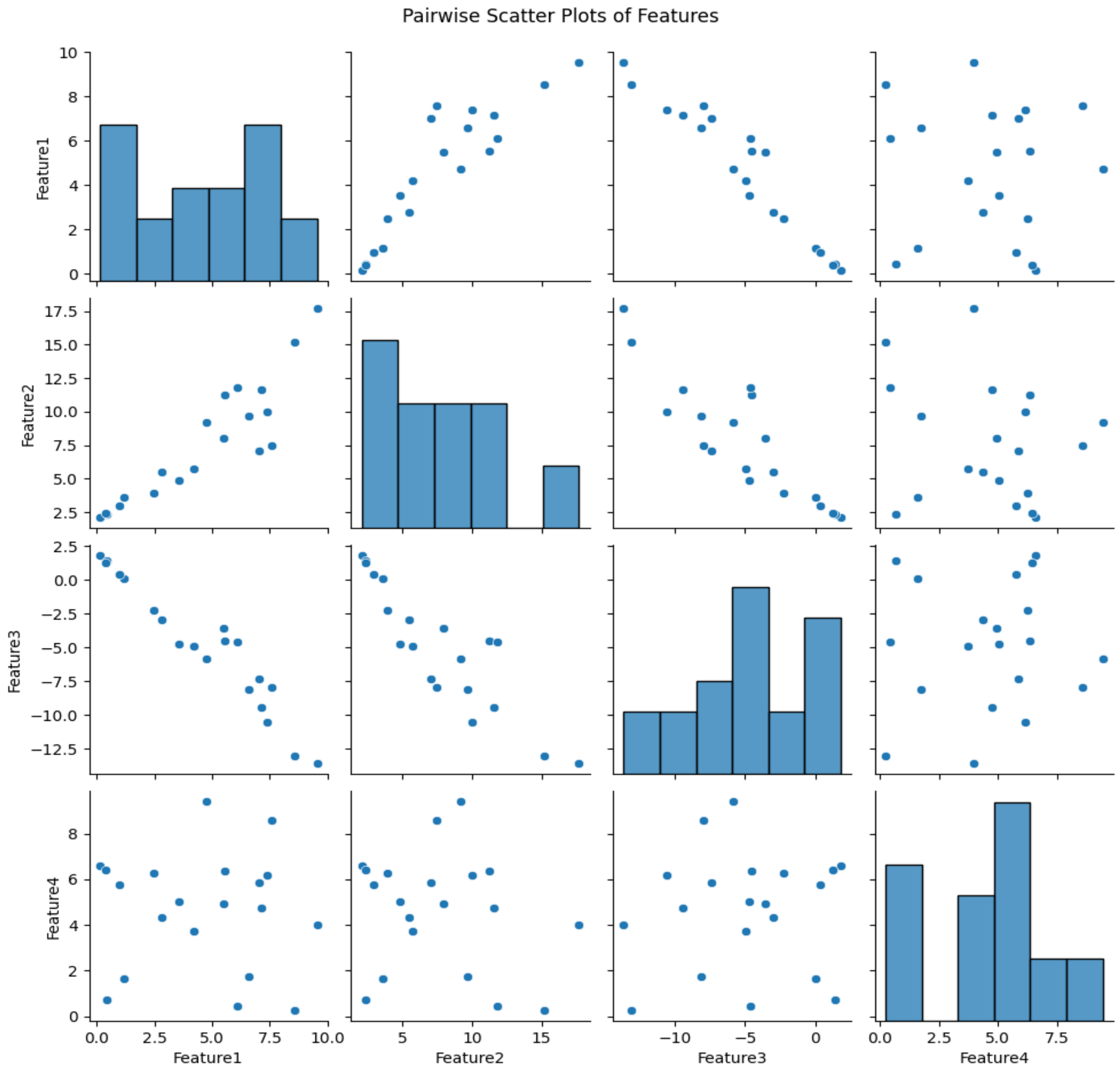
Linear Regression

- The goal of linear regression is to find a linear relationship between one or more independent variables and a dependent variable.
- A linear regression uses the fitting of a line or hyperplane to model the relationship between the variables.
- This linear model can then be used to predict the value of the dependent variable for new data.



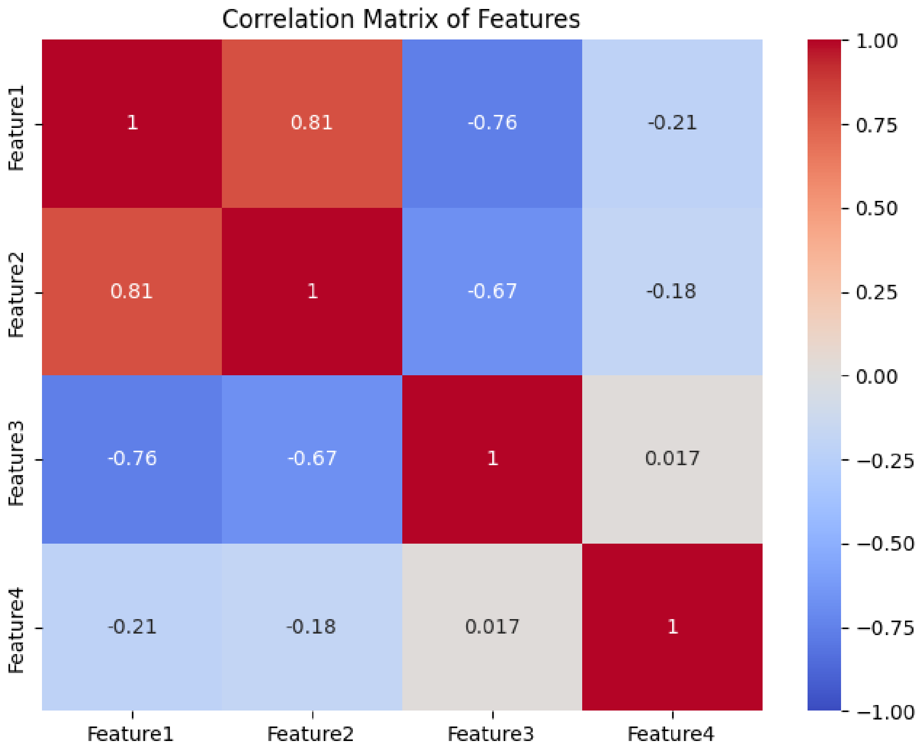
Linear Regression

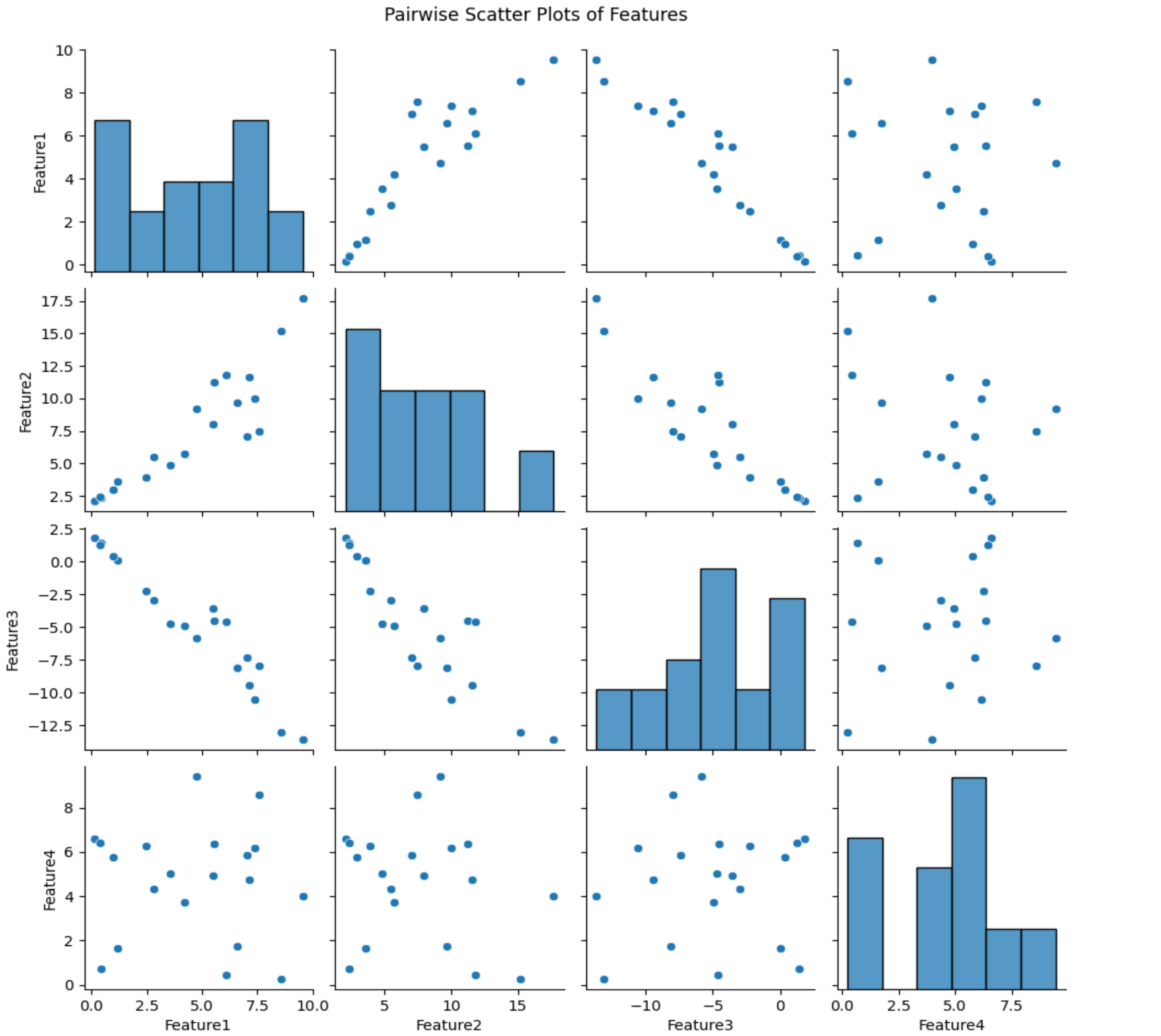
- Is linear regression feasible?



Correlation

$$r = \frac{n\Sigma xy - (\Sigma x)(\Sigma y)}{\sqrt{n\Sigma x^2 - (\Sigma x^2)}\sqrt{n\Sigma y^2 - (\Sigma y^2)}}$$



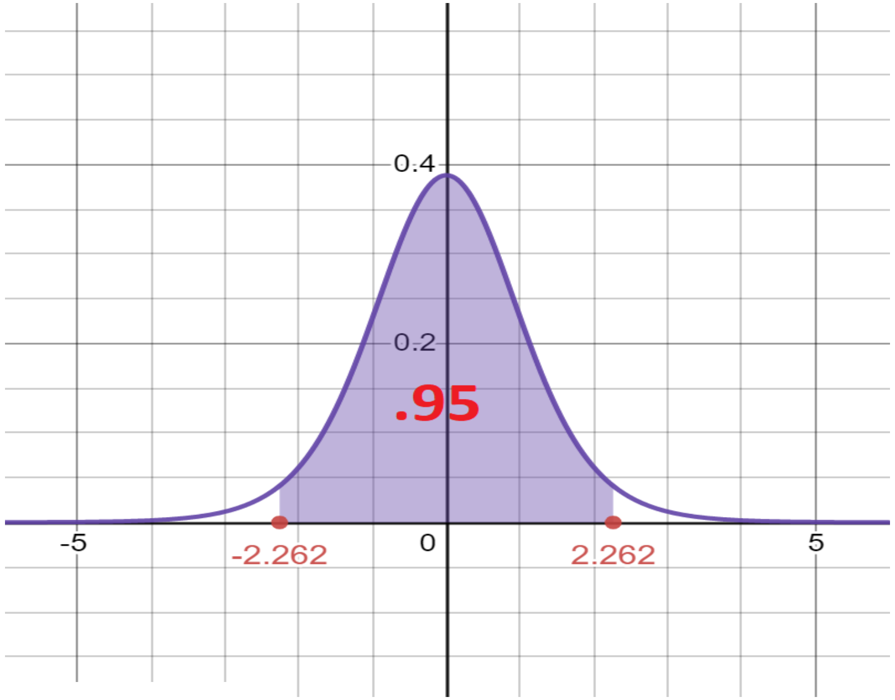


Correlation (Statistical significance)

$$r = \frac{n\Sigma xy - (\Sigma x)(\Sigma y)}{\sqrt{n\Sigma x^2 - (\Sigma x^2)}\sqrt{n\Sigma y^2 - (\Sigma y^2)}}$$

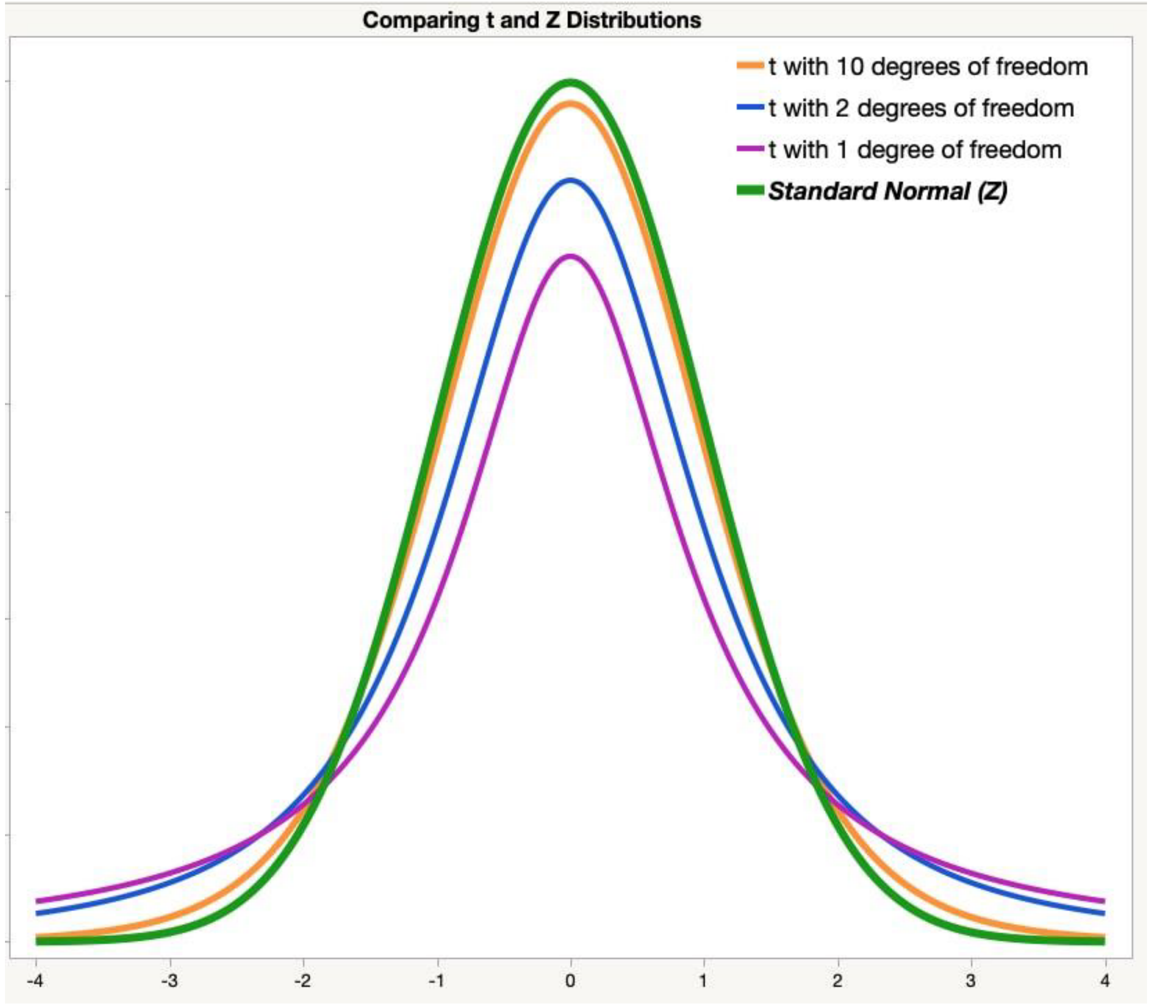
H0:ρ = 0 (meaning no relationship)
H1:ρ ≠ 0 (relationship exists)

If our test value (t) falls outside this range (-2.262, 2.262), we can reject our null hypothesis.(With 95% confidence)



$$t = \frac{r}{\sqrt{\frac{1-r^2}{n-2}}}$$

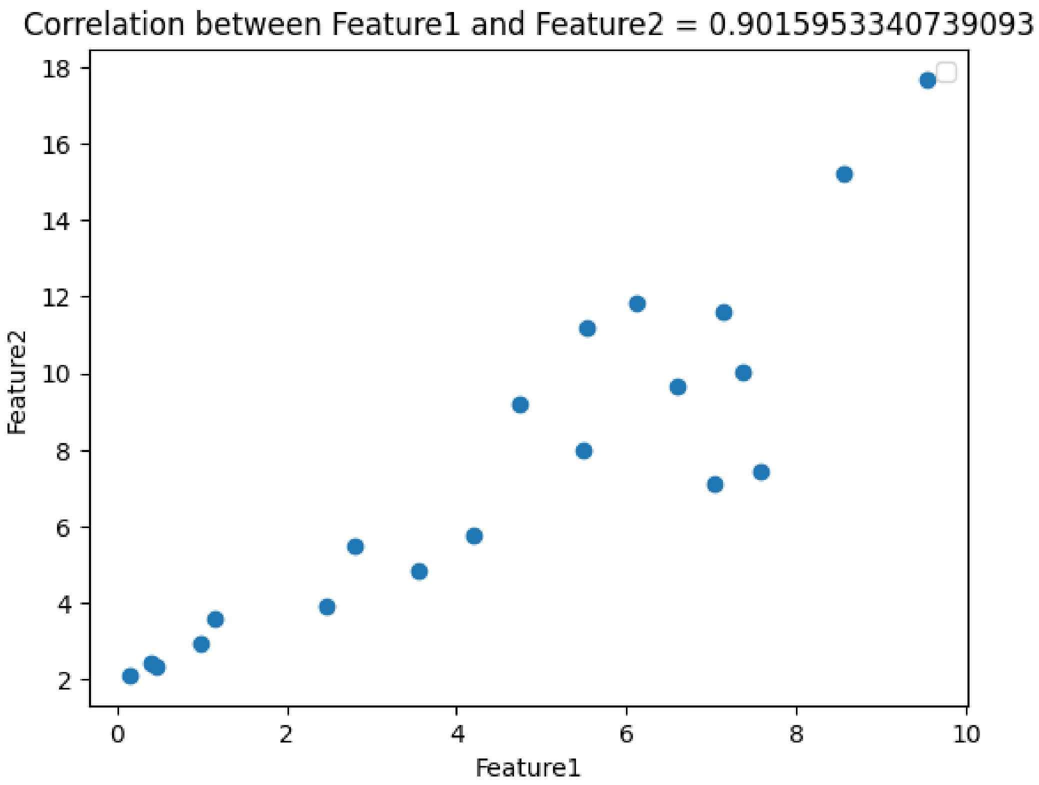
t-distribution and normal-distribution

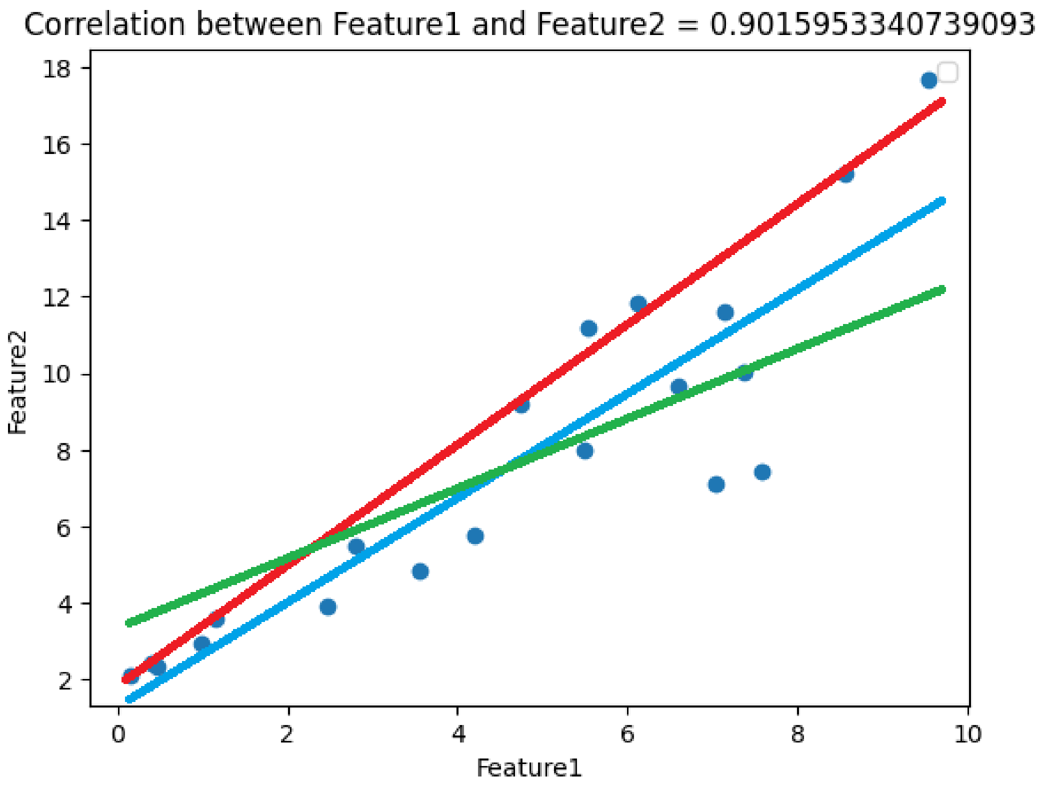


Python Code

- Correlation Matrix
- Statistical significance
- p-value describes the likelihood of data occurring if the null hypothesis were true

Which line is best?



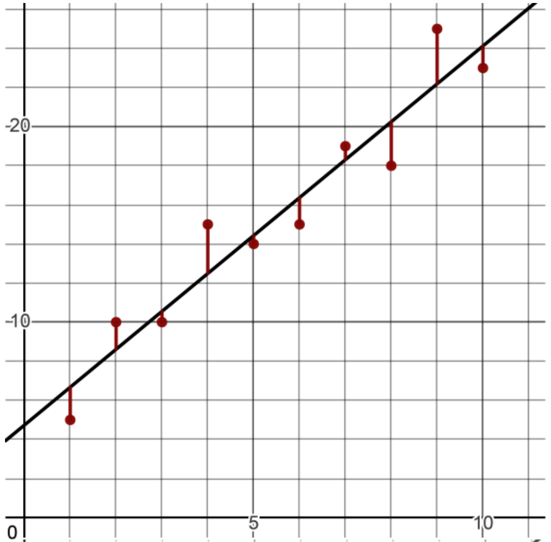


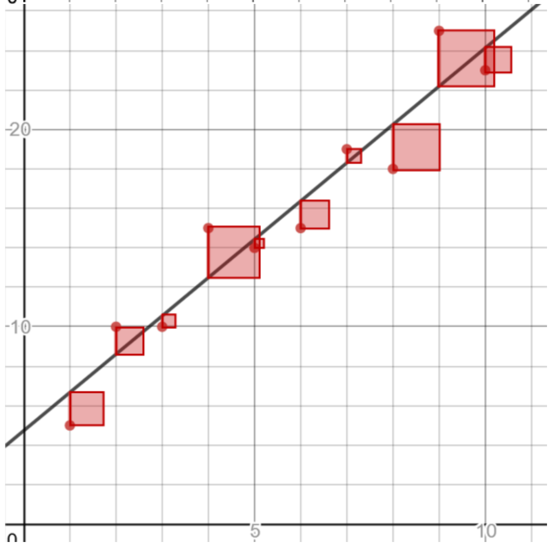
Model and loss function

$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n$$

$$J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2$$

- m is the number of training examples.
- $y^{(i)}$ is the actual value for the i-th training example.
- $h_\theta(x^{(i)})$ is the predicted value for the i-th training example.



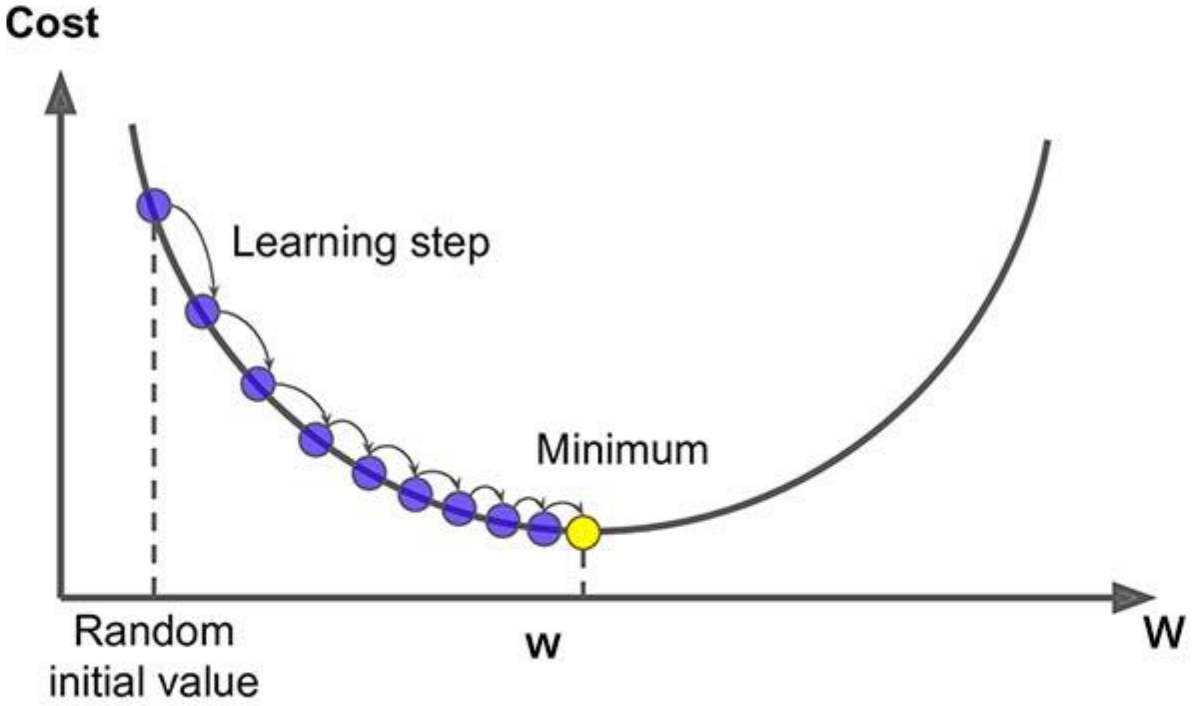


Gradient Descent

$\forall\ \boldsymbol{\theta_j}$ = ***Random initialize***

$$\theta_j := \theta_j - \alpha \frac{\partial J(\theta)}{\partial \theta_j}$$

$$\frac{\partial J(\theta)}{\partial \theta_j} = \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

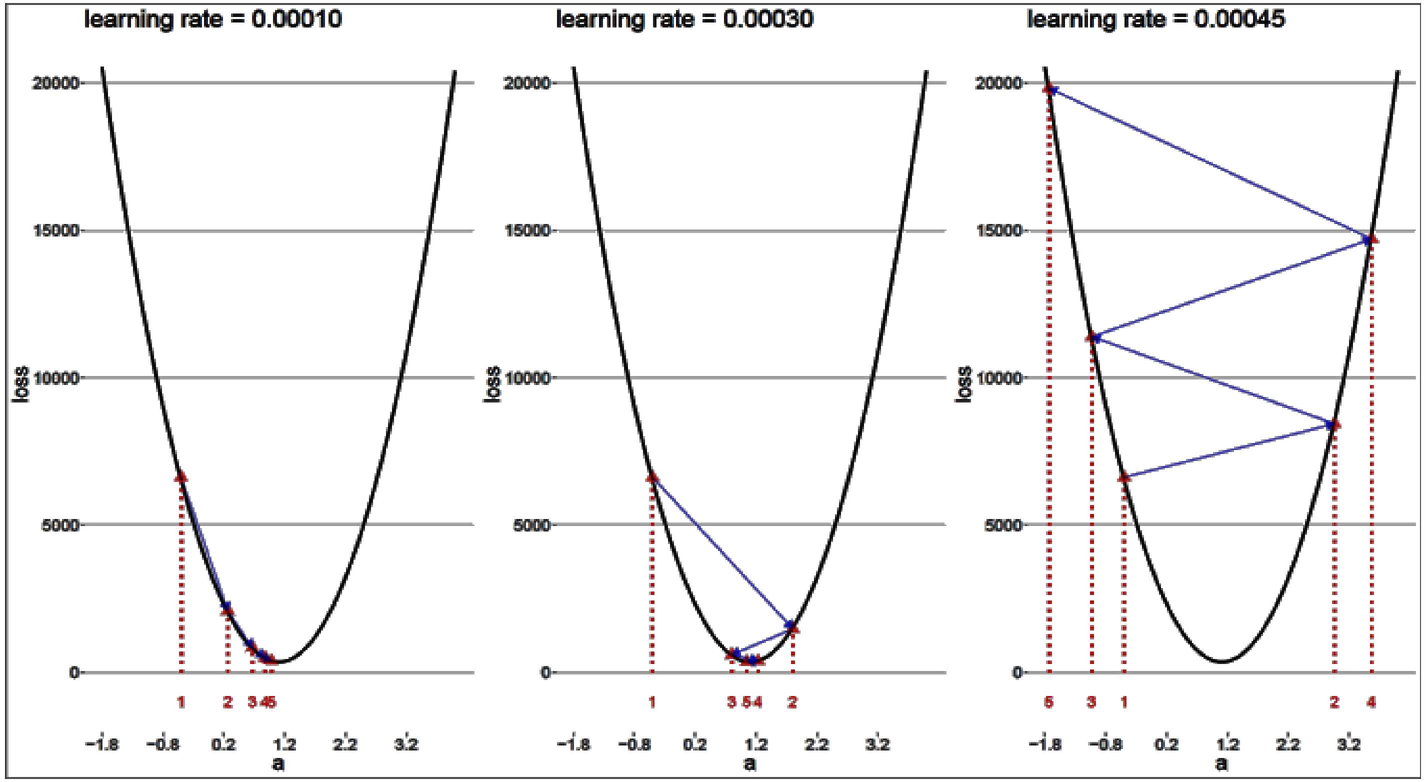


Gradient Descent

```
Initialize model parameters (weights and biases) randomly
Set learning rate (α)
For each iteration:
    Initialize total_loss = 0
    Initialize gradients for weights and biases as zero
    For each data point (x_i, y_i) in the training set:
        predicted_y_i = model(x_i, weights, biases)
        gradient_weights, gradient_biases = compute_gradients(x_i, y_i, predicted_y_i)
        gradients_weights += gradient_weights
        gradients_biases += gradient_biases
    gradients_weights /= N
    gradients_biases /= N

    weights = weights - α * gradients_weights
    biases = biases - α * gradients_biases
```

Learning Rate



Python Code

- Solving by hand
  - X = [2, 4, 5, 6, 6, 8, 8, 9, 9]
  - y = [1, 4, 4, 5, 6, 7, 6 , 8, 10]
- Gradient Descent
- Stochastic Gradient Descent
- Convergence and Learning Rate

Inverse matrix technique

y = βX

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \quad \mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} \quad \beta = \begin{bmatrix} \beta_0 \\ \beta_1 \end{bmatrix}$$

Inverse matrix technique

y = βX

$$\mathbf{x} = \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \vdots \\ 1 & x_n \end{bmatrix}$$

$$\mathbf{e}(\beta) = \mathbf{y} - \mathbf{x}\beta$$

Inverse matrix technique

$$MSE(\beta) = \frac{1}{n}\sum_{i=1}^{n} e_i^2(\beta)$$

$$MSE(\beta) = \frac{1}{n}\mathbf{e}^T\mathbf{e}$$

$$\begin{aligned} MSE(\beta) &= \frac{1}{n}\mathbf{e}^T\mathbf{e} \\ &= \frac{1}{n}(\mathbf{y}-\mathbf{x}\beta)^T(\mathbf{y}-\mathbf{x}\beta) \\ &= \frac{1}{n}(\mathbf{y}^T-\beta^T\mathbf{x}^T)(\mathbf{y}-\mathbf{x}\beta) \\ &= \frac{1}{n}\left(\mathbf{y}^T\mathbf{y}-\mathbf{y}^T\mathbf{x}\beta-\beta^T\mathbf{x}^T\mathbf{y}+\beta^T\mathbf{x}^T\mathbf{x}\beta\right) \end{aligned}$$

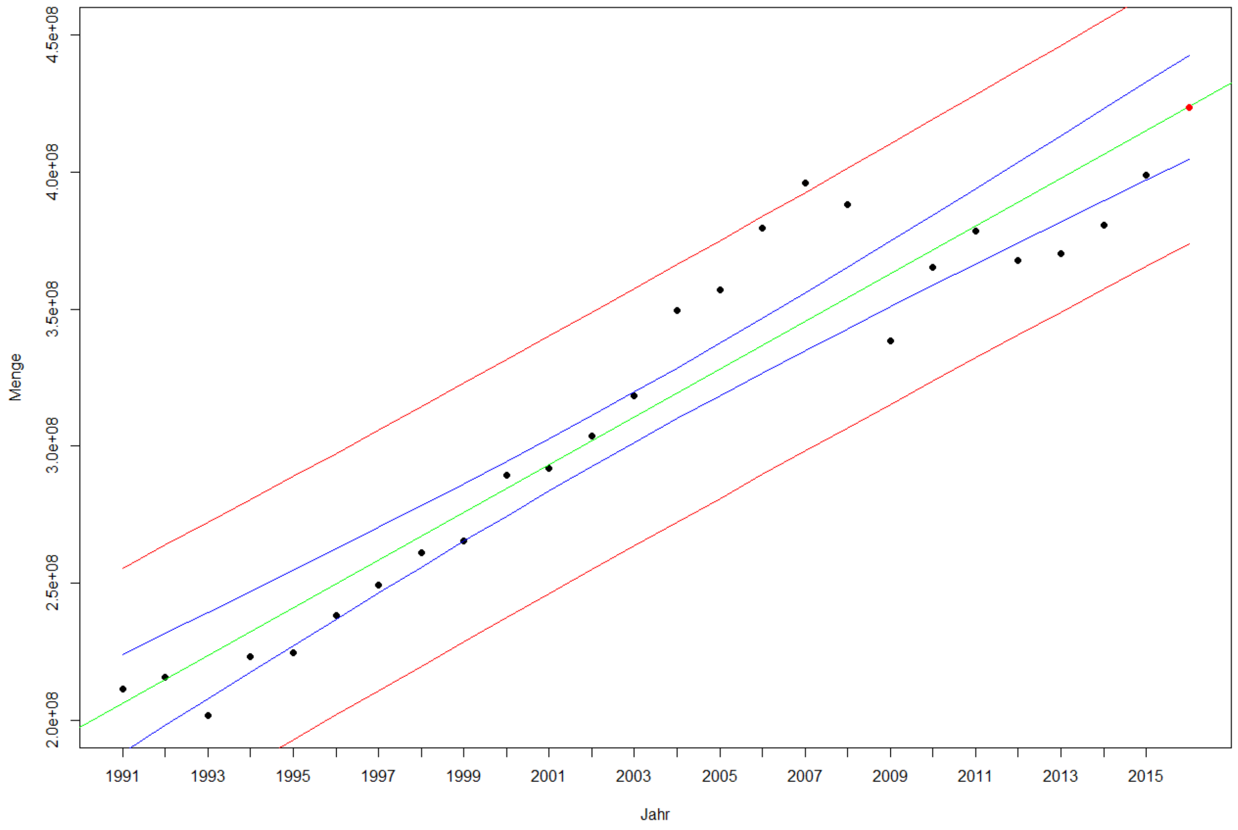
$$\begin{aligned} \nabla MSE(\beta &= \frac{1}{n}\left(\nabla\mathbf{y}^T\mathbf{y}-2\nabla\beta^T\mathbf{x}^T\mathbf{y}+\nabla\beta^T\mathbf{x}^T\mathbf{x}\beta\right) \\ &= \frac{1}{n}\left(0-2\mathbf{x}^T\mathbf{y}+2\mathbf{x}^T\mathbf{x}\beta\right) \\ &= \frac{2}{n}\left(\mathbf{x}^T\mathbf{x}\beta-\mathbf{x}^T\mathbf{y}\right) \end{aligned}$$

$$\boxed{\widehat{\beta} = (\mathbf{x}^T\mathbf{x})^{-1}\mathbf{x}^T\mathbf{y}}$$

Prediction intervals

$$E = t_{.025} * S_e * \sqrt{1 + \frac{1}{n} + \frac{n(x_0 + \bar{x})^2}{n\left(\sum x^2\right) - \left(\sum x\right)^2}}$$

$$S_e = \sqrt{\frac{\sum(y - \hat{y})^2}{n-2}}$$

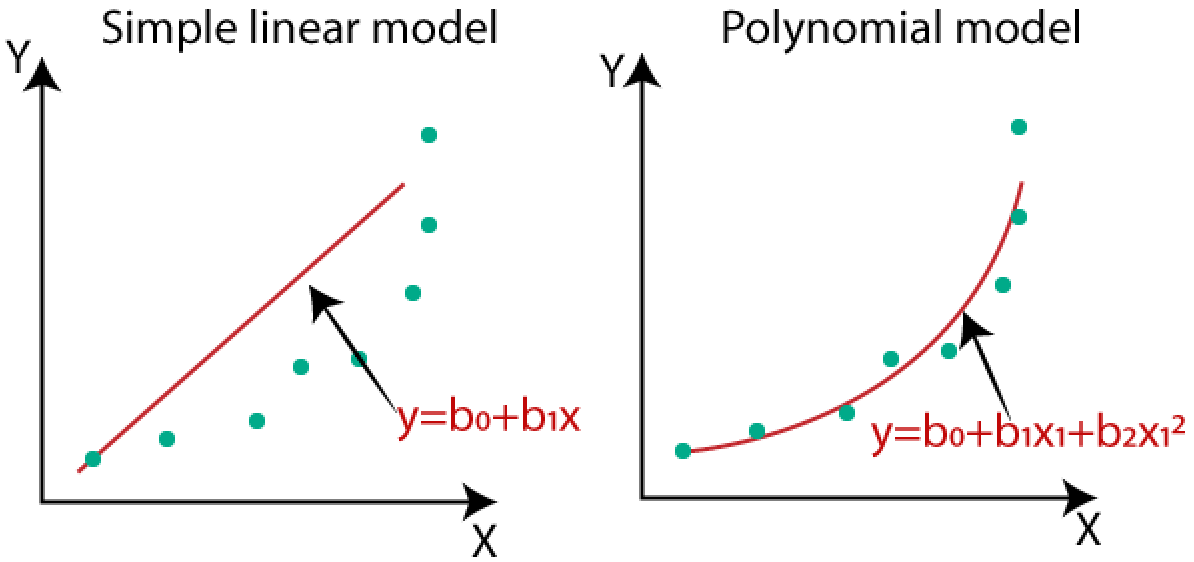


Python Code

- Analytical approach
- Prediction intervals

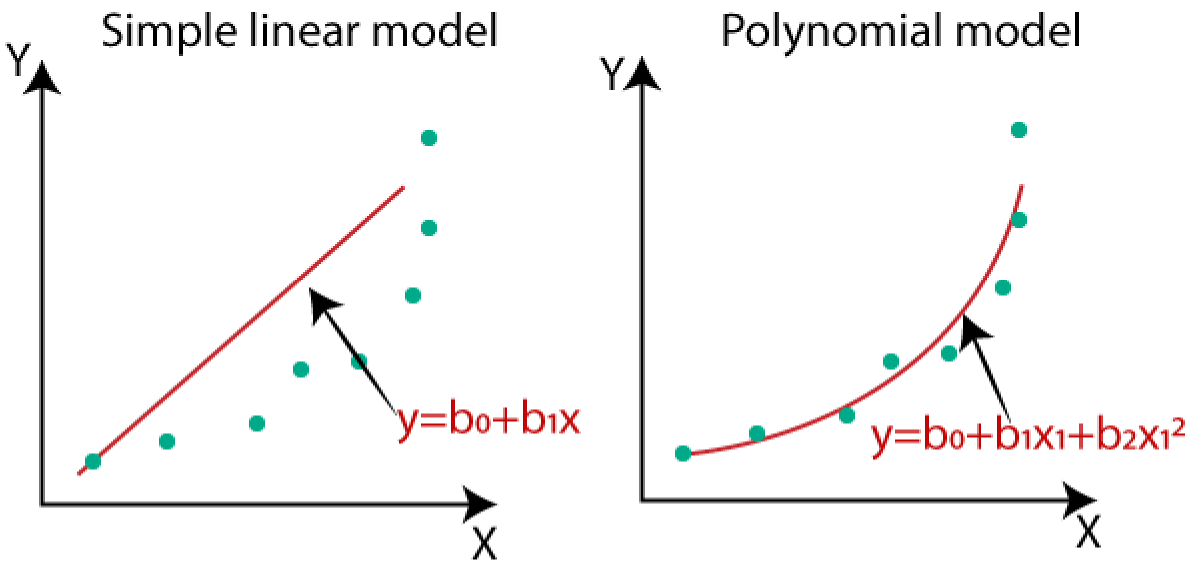
Non-linear Regression

- **Polynomial**: $y = \beta_0 + \beta_1 x + \beta_2 x^2 + \cdots + \beta_n x^n$
- **Exponential**: $y = \beta_0 e^{\beta_1 x}$
- **Logarithmic**: $y = \beta_0 + \beta_1 \ln(x)$
- **Sigmoidal**: $y = \frac{L}{1+e^{-k(x-x_0)}}$ (used in logistic growth models)
- **Power Law**: $y = \beta_0 x^{\beta_1}$

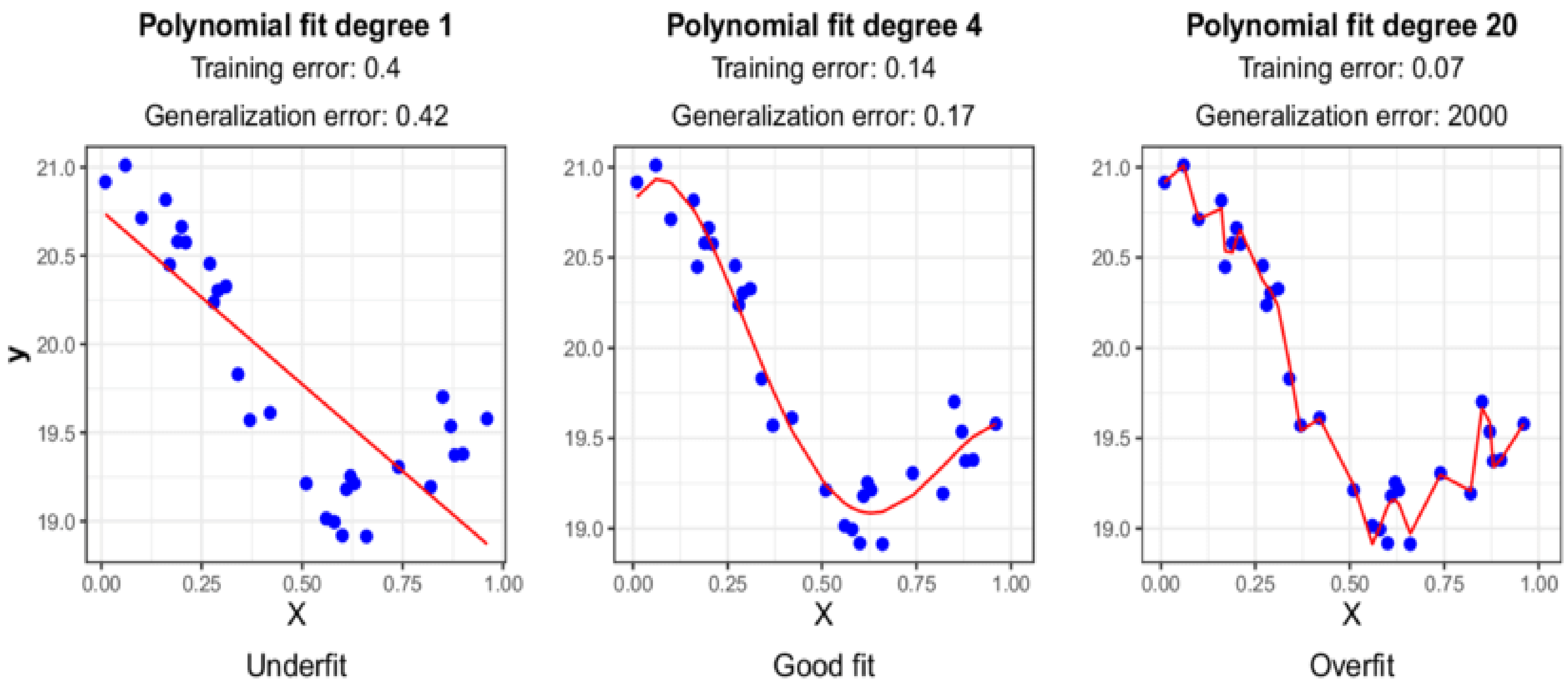


Non-linear Regression

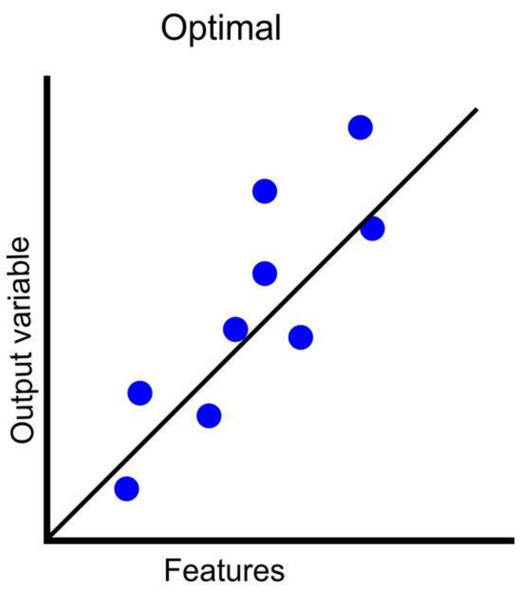
- Transform data
- Find optimal parameters
- Transform parameters

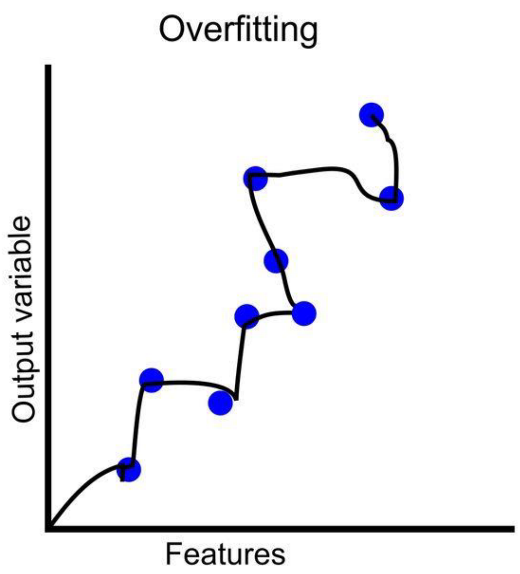


Non-linear Regression (Overfitting)

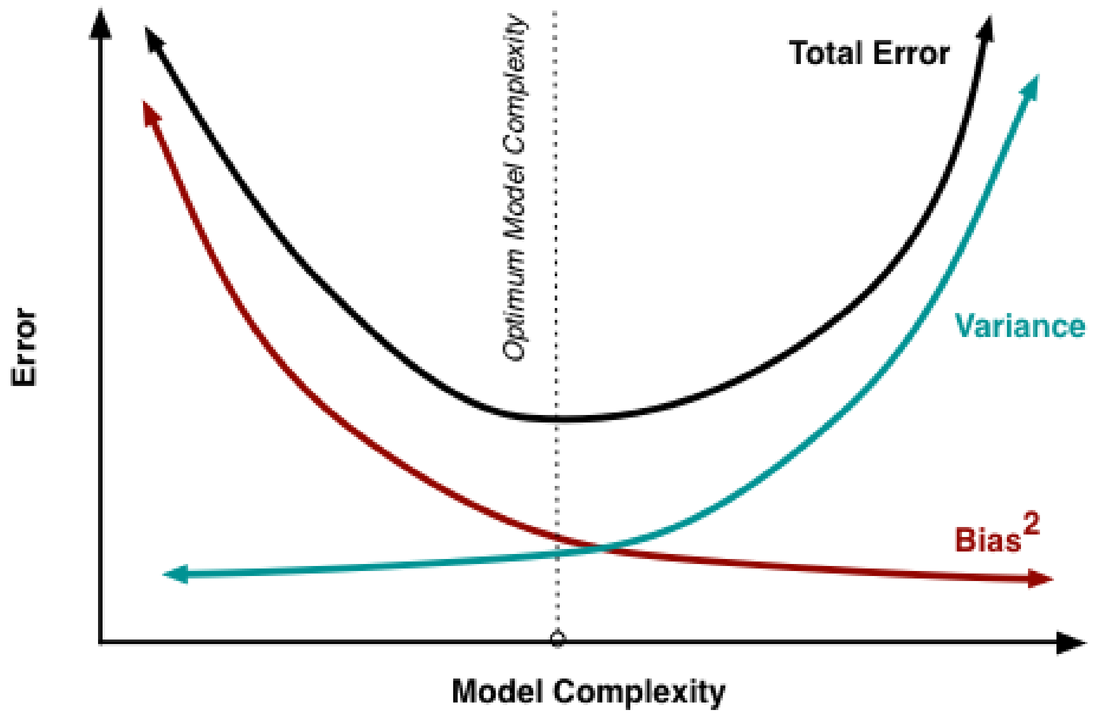


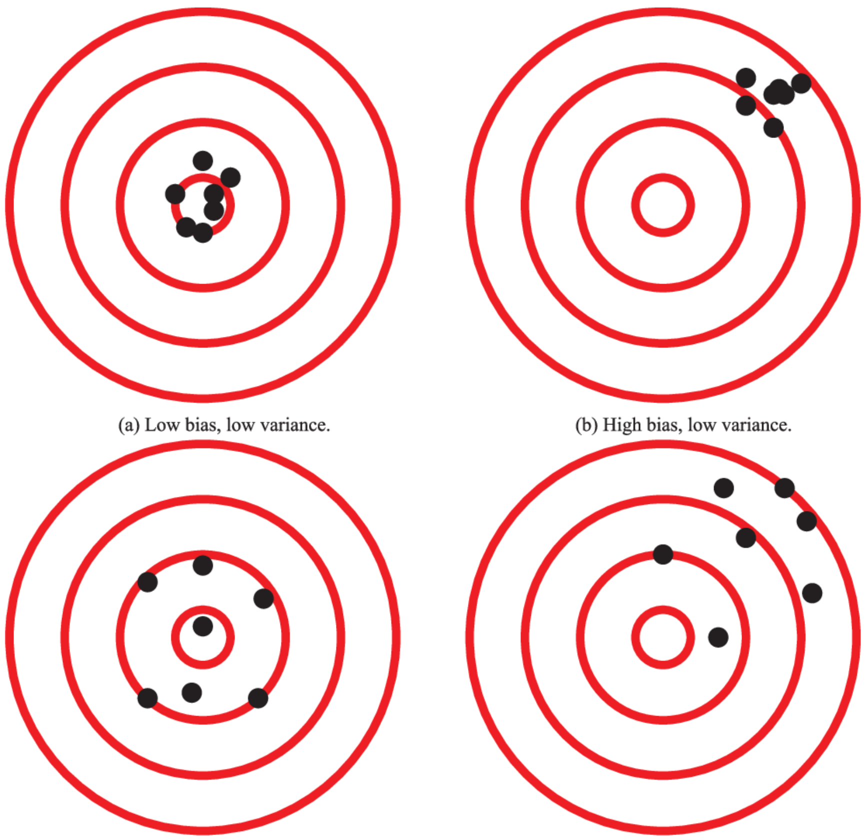
Overfitting





Bias and Variance Tradeoff





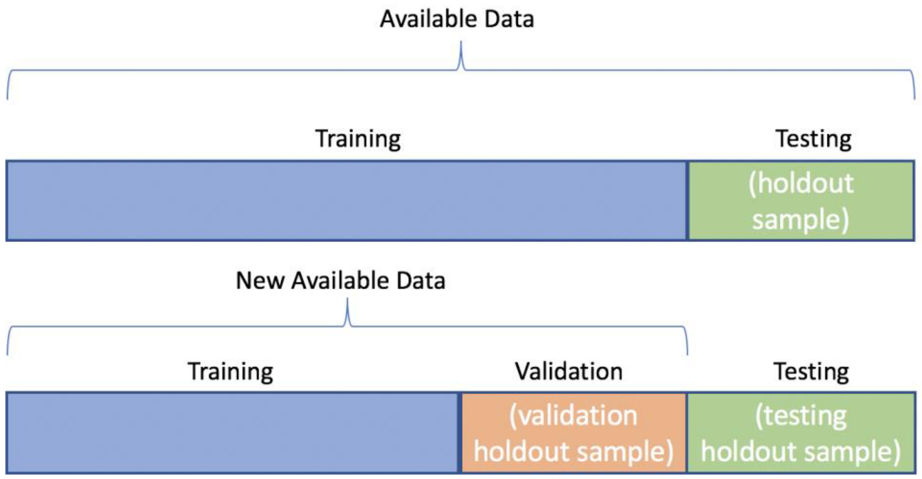
(a) Low bias, low variance.

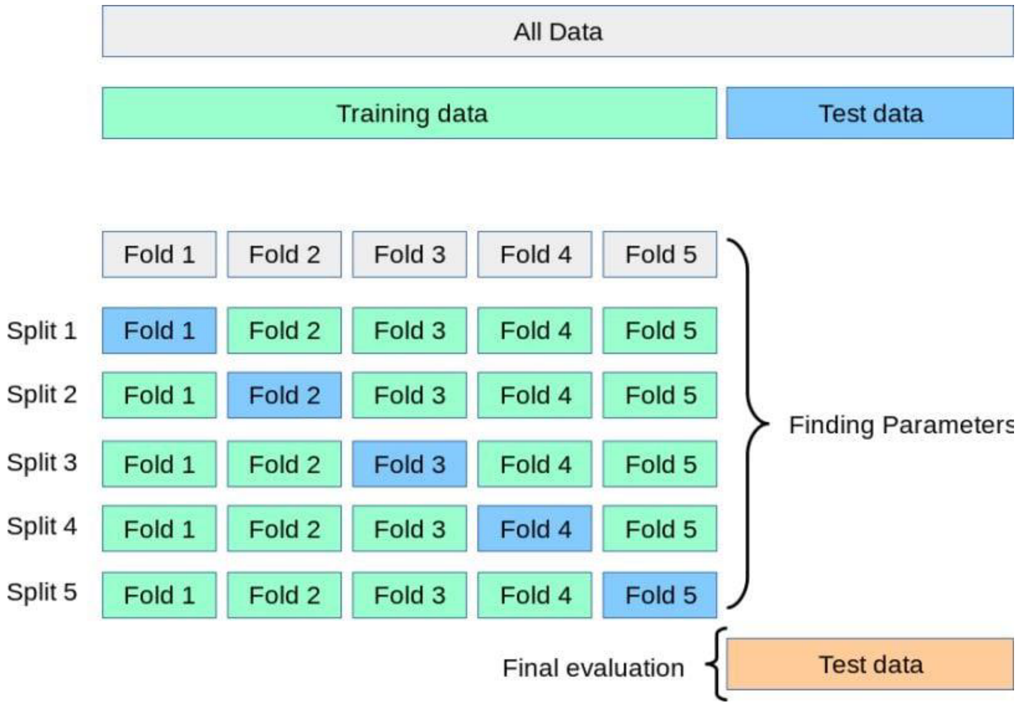
(b) High bias, low variance.

(c) Low bias, high variance.

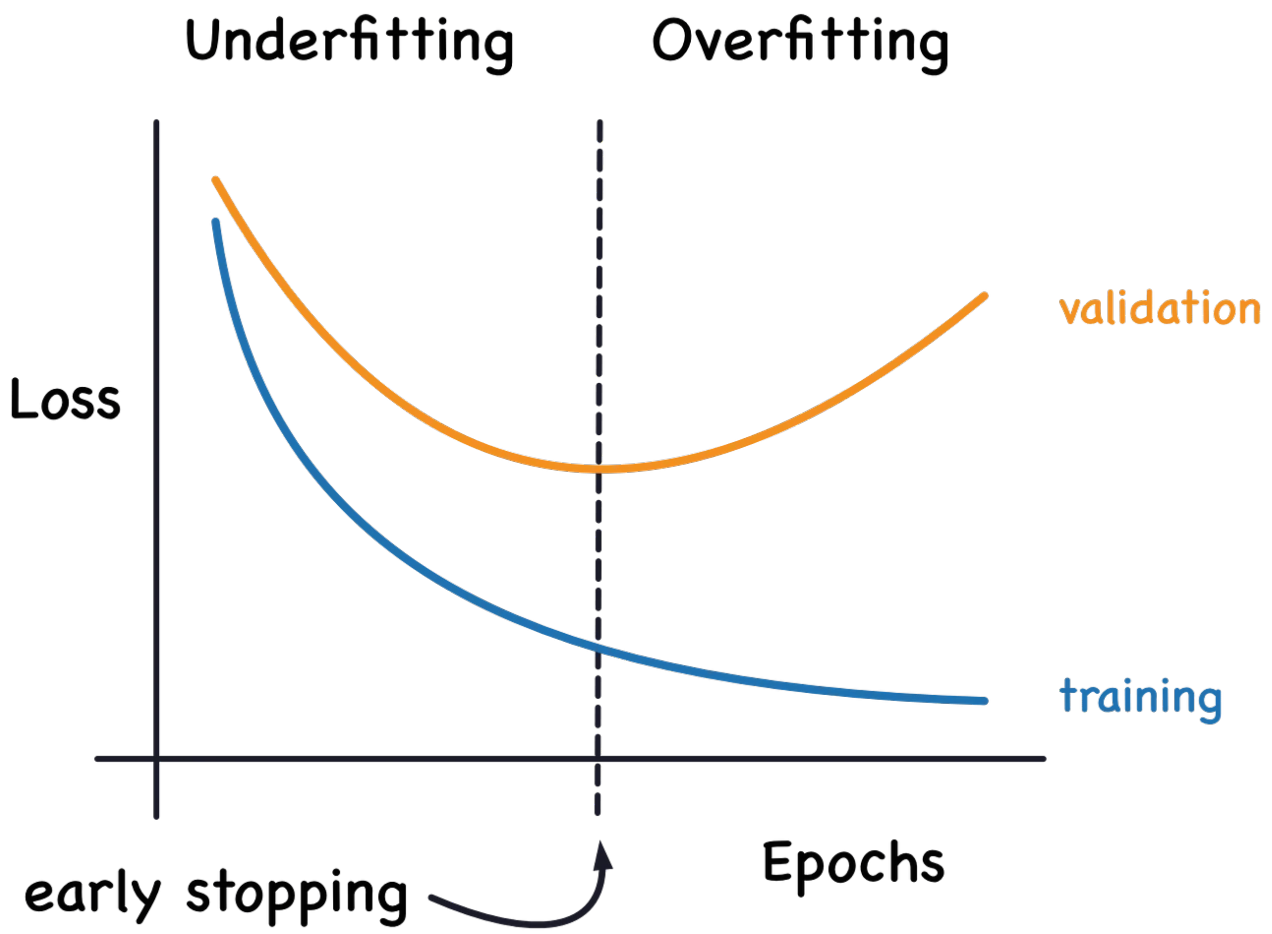
(d) High bias, high variance.

Train-Test Split-Cross Validation





Early Stopping



L2 Regularization

$$\text{Loss} = \frac{1}{2n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 + \frac{\lambda}{2}\sum_{j=1}^{p}\beta_j^2$$

$$\frac{\partial \text{Loss}}{\partial \beta_j} = -\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)x_{ij} + \lambda\beta_j$$

$$\beta_j = \beta_j - \eta\frac{\partial \text{Loss}}{\partial \beta_j}$$

Python

- Non-linear Regression
- L2 Regularization

Homework

- Download dataset from github (data-regression.csv)
- Split data into train and test.
- Train your model based-on one of common mathematical function.
- Use test data for evaluation your model.
- All code should write from scratch.