Model Selection and Evaluation

Qing-Yuan Jiang

The school of Computer Science and Engineering, NJUST

2026.09.24

Review of Previous Class

- Why optimization matters in machine learning?
- What is convexity?
- Convex sets and convex functions
- Convex optimization problems
- Gradient and gradient descent
- Stochastic gradient descent (SGD)
- Mini-batch SGD

Why Model Selection?

The Machine Learning Pipeline

- The main process of machine learning:

Data Collection → Model → Optimization → Learned Parameters

- After optimization, we obtain a trained model.
- There exist many machine learning models, such as: LR, SVM, Decision Tree, DNN
- **How do we know whether the learned models is actually good?**

Training Performance is Not Enough!

- Suppose we have two models trained on **the same** datasets
- **Model A:** 95% training accuracy
- **Model B:** 90% training accuracy
- Which one should we use?
- Can you design a model with 100% accuracy on **training set**.
  - **Hint:** no unclear label for training data

Can We Always Fit the Training Data?

- Suppose a model: $\mathcal{M} = \{\theta = \mathcal{X} = \{(x_i, y_i)\}_{i=1}^{n}\}$
- Given any data, if $x = x_i \in \theta$, the output of the model is

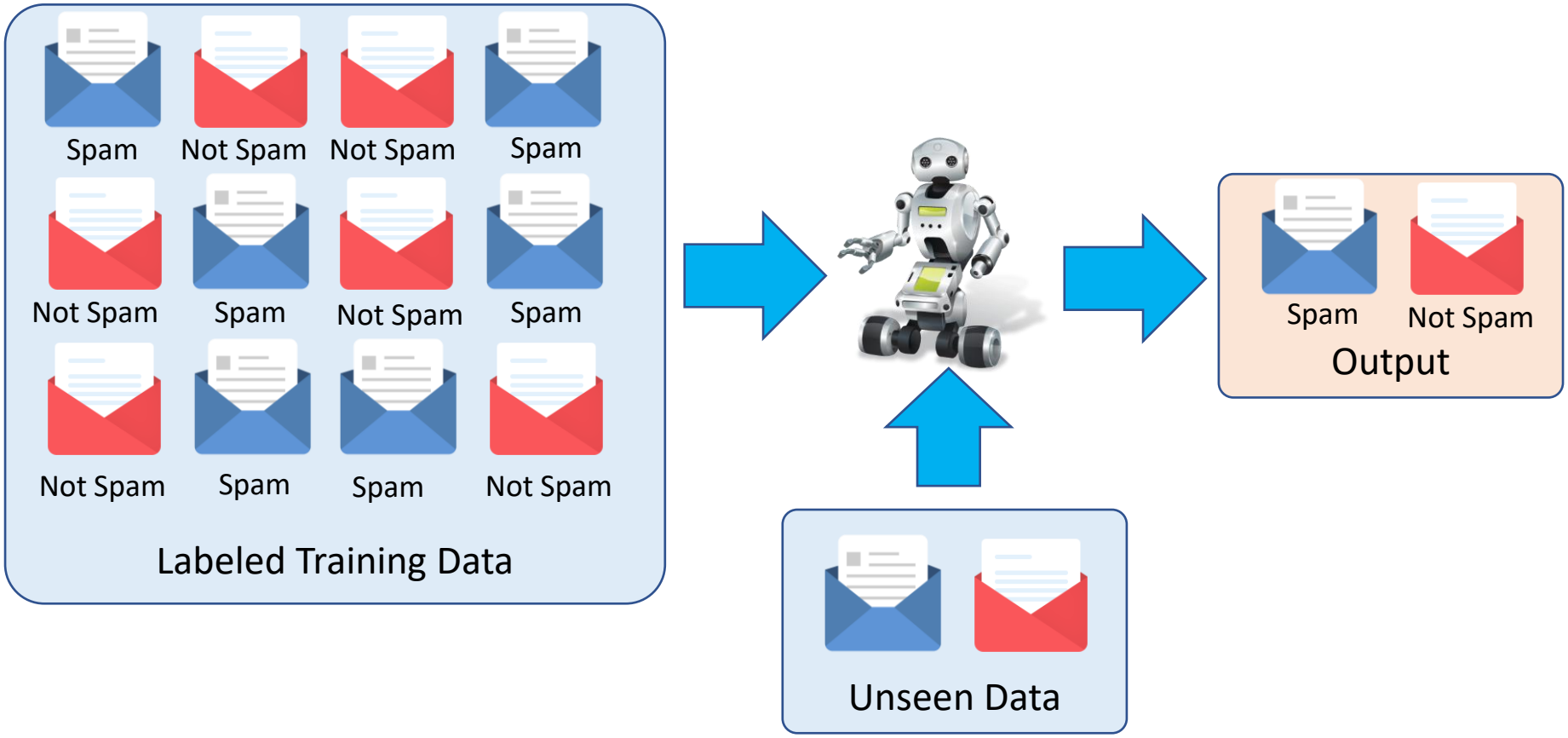
  defined: y_i.

  else: output a random label.
- What is the training accuracy of the model?
- How about the testing accuracy?

- What we really care about is **performance on unseen data**.

Generalization

- Review the spam email detection example:



- Training on **observed data**, testing on **unseen data**
- What ultimately matters is not how well the model fits the training data, but **how well it performs on unseen examples**.

Empirical Risk vs. Generalization Risk

- Empirical Risk:

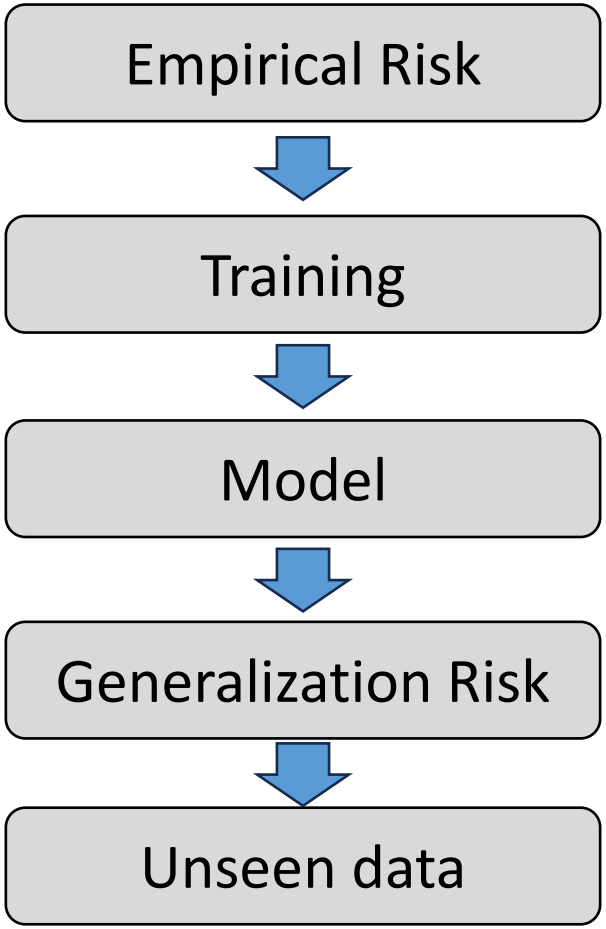
$$R_{emp}(f) = \frac{1}{n} \sum_{i=1}^{n} L(f(x_i), y_i)$$

- Generalization Risk:

$$R(f) = \mathbb{E}_{(x,y)\sim\mathcal{D}} L(f(x_i), y_i)$$

- Empirical risk measures performance on observed samples.
- Generalization risk measures expected performance on the underlying data distribution.
- We **optimize empirical risk**, but we **care about generalization risk**.

Generalization Gap

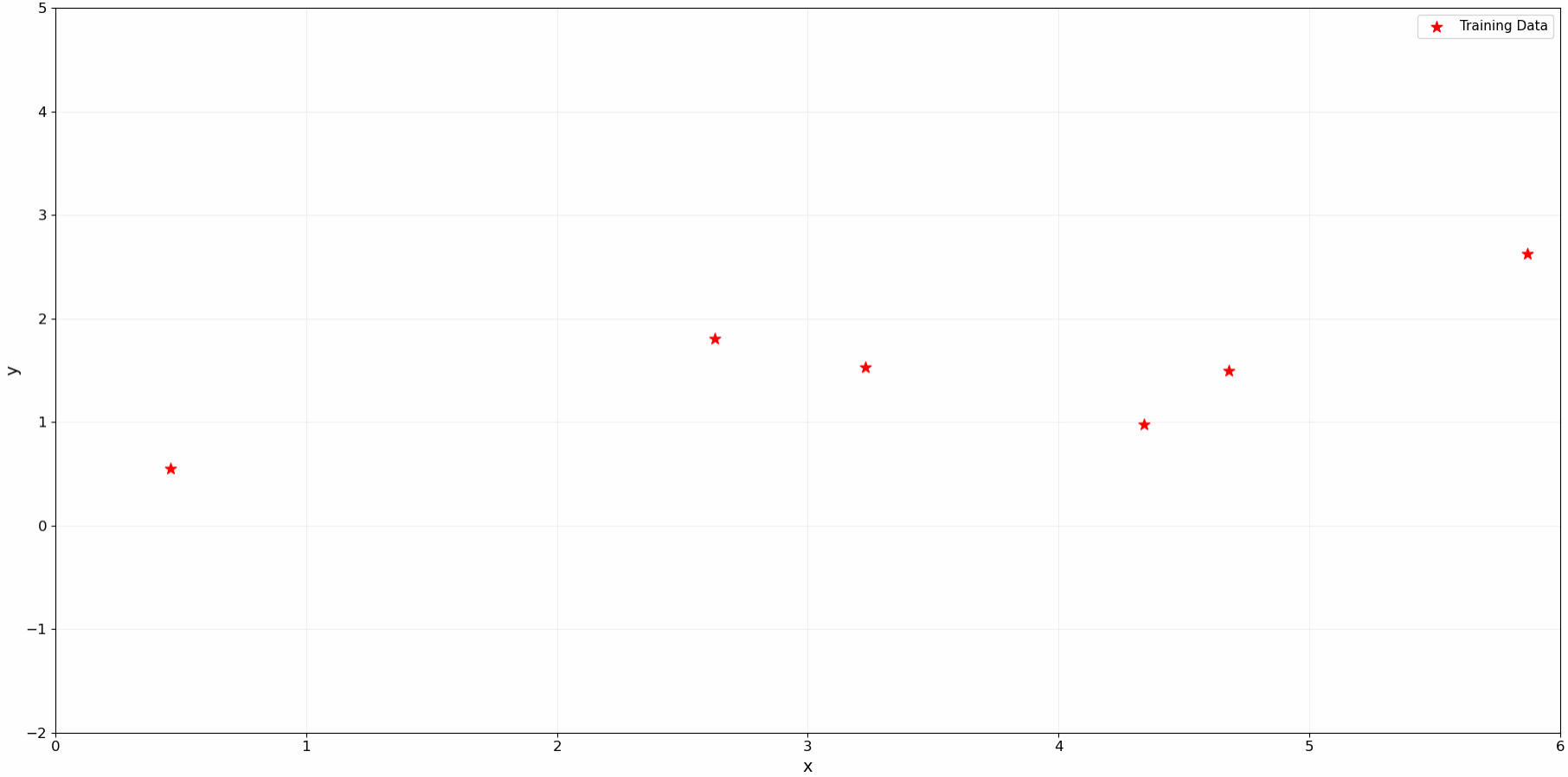


The fundamental challenge is the gap between what we observe on training data and what we care about on unseen data.

Model Selection Pipeline

Why Training Error Can Be Misleading

- Polynomial fitting example



- Increasing model complexity can improve the fit to training data while simultaneously hurting generalization.

Underfitting & Overfitting

Underfitting:

- The model cannot capture important patterns in the data.
- Both training and test errors remain high.
- The model has insufficient expressive capacity.

Overfitting:

- Model fits training examples extremely well.
- It may also fit noise or idiosyncrasies in the training data.
- Performance deteriorates on unseen data.

Examples of Underfitting and Overfitting

- Train a model to learn “what is the tree”
- **Underfitting:**
  - “If it is **green**, it is a tree.”
  - The model is too simple and fails to capture the underlying pattern.
- **Overfitting:**
  - If it is **green**, has **serrated leaves**, has **exactly five leaves**, and the leaves have this specific shape, it is a tree.
  - The model is too specific and memorizes the training examples rather than learning general patterns.

Bias and Variance Trade-Off

- Given a dataset $\mathcal{D} = \{(x_i, \hat{y}_i)\}_{i=1}^n$, suppose its ground-truth as: $\{y_i\}_{i=1}^n$
  - What is the different between $\hat{y}$ and y?
- Define the prediction as: $\bar{y}_i = f(x_i)$
- The generalization error is defined as:

$$E(f, \mathcal{D}) = \frac{1}{n} \sum_{i=1}^n \mathcal{I}(f(x_i) \neq y_i)$$

- We have:

$$\begin{aligned} E(f, \mathcal{D}) =& \mathbb{E}[\bar{f}(x) - y]^2 + (f(x, \mathcal{D}) - \bar{f}(x))^2 + \mathbb{E}[\hat{y} - y]^2 \\ =& bias^2(x) + var(x) + \epsilon^2 \end{aligned}$$

Train, Validation, and Test Sets

- Dataset: train set + validation set + test set
- Train set:
  - Learn model parameters
- Validation set:
  - Select models and hyperparameters.
- Test set (unseen data):
  - Estimate final performance
- **The validation set supports decisions; the test set supports final evaluation.**

Parameters vs. Hyperparameters

- Parameters:
  - Learned by optimization:

$$\omega^* = \text{argmin}_\omega L(\omega)$$

- Examples:
  - Linear weights
  - Neural network weights
  - SVM parameters

Parameters vs. Hyperparameters

- Hyperparameters: specified before/during training
  - Learning rate
  - Regularization strength
  - Tree depth
  - Number of neighbors
  - Kernel parameters
- Parameters are learned from training data
- Hyperparameters are selected.
- Hyperparameters determine how the learning process is configured.

Examples of Hyperparameter Selection

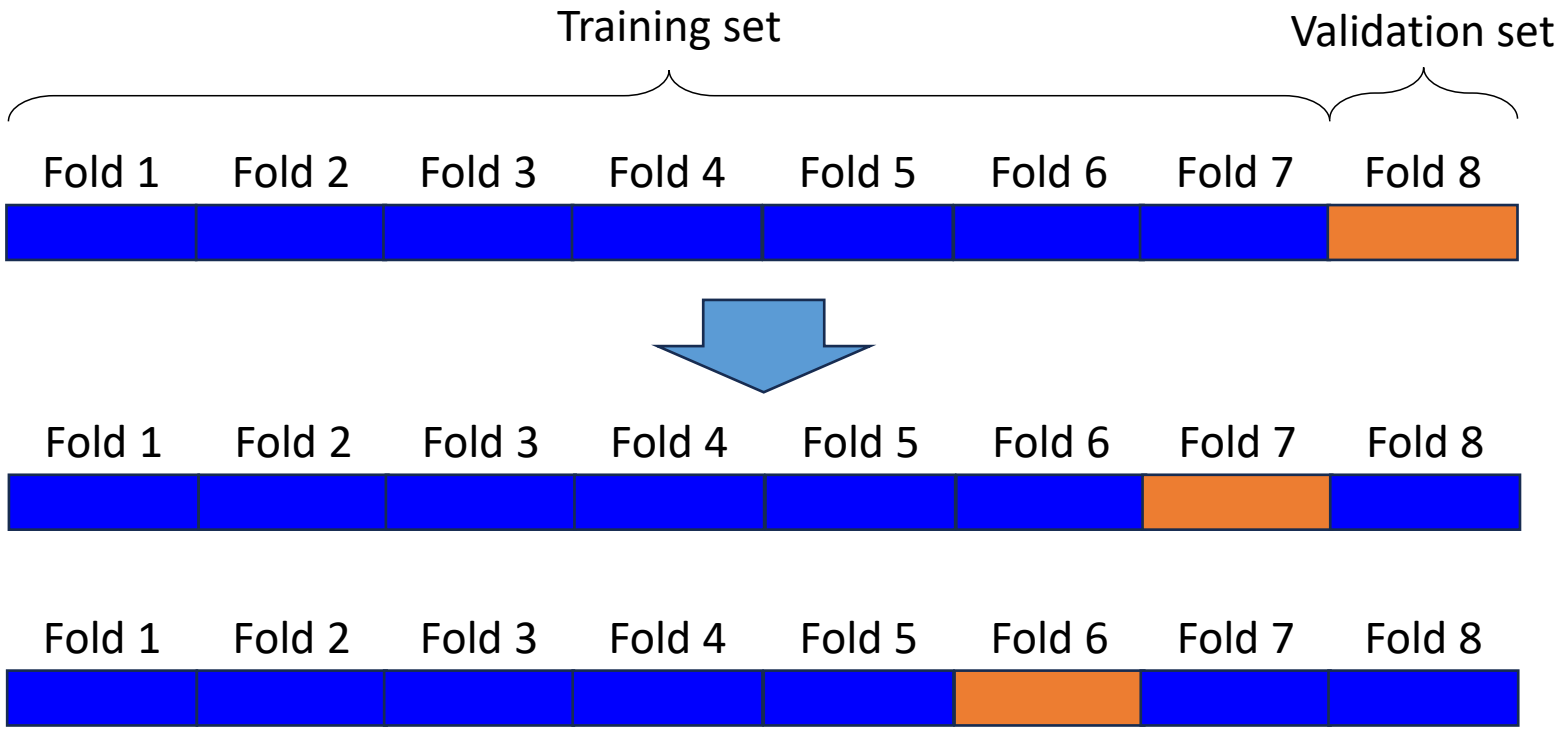
- Suppose we have a model containing a hyperparameter λ

$$\lambda = 0.001, \ \text{accuracy} = 98\%,$$
$$\lambda = 0.01, \ \text{accuracy} = 99\%,$$
$$\lambda = 0.1, \ \text{accuracy} = 92\%,$$

- Which λ should we choose?
- Choose the one that performs **best** on the validation set.

K-Fold Cross-Validation

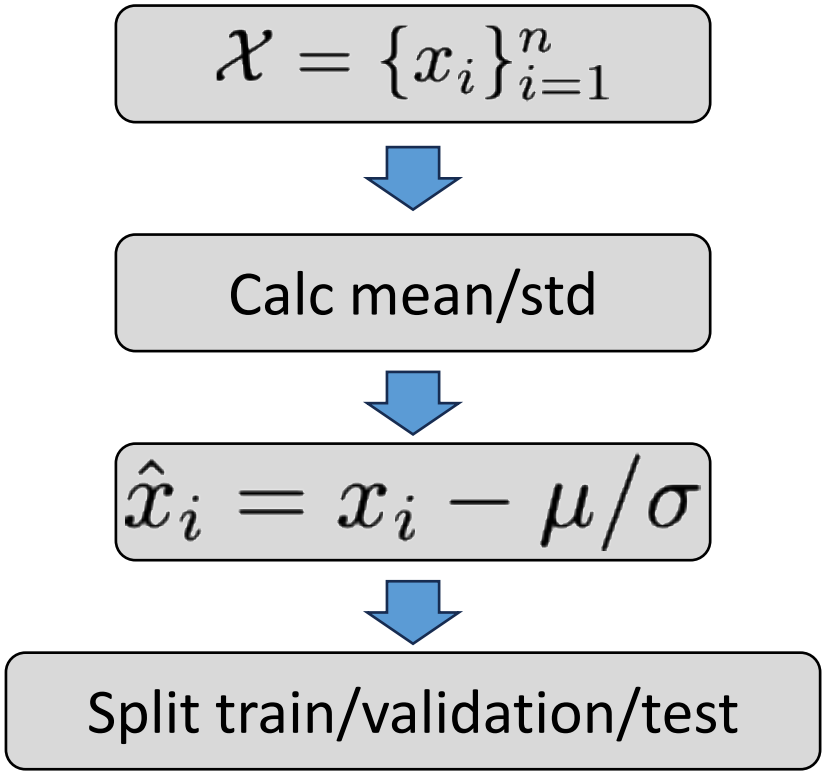
- What if the dataset is small?



$$\text{CV score} = \frac{1}{K}\sum_{k=1}\text{score}_k$$

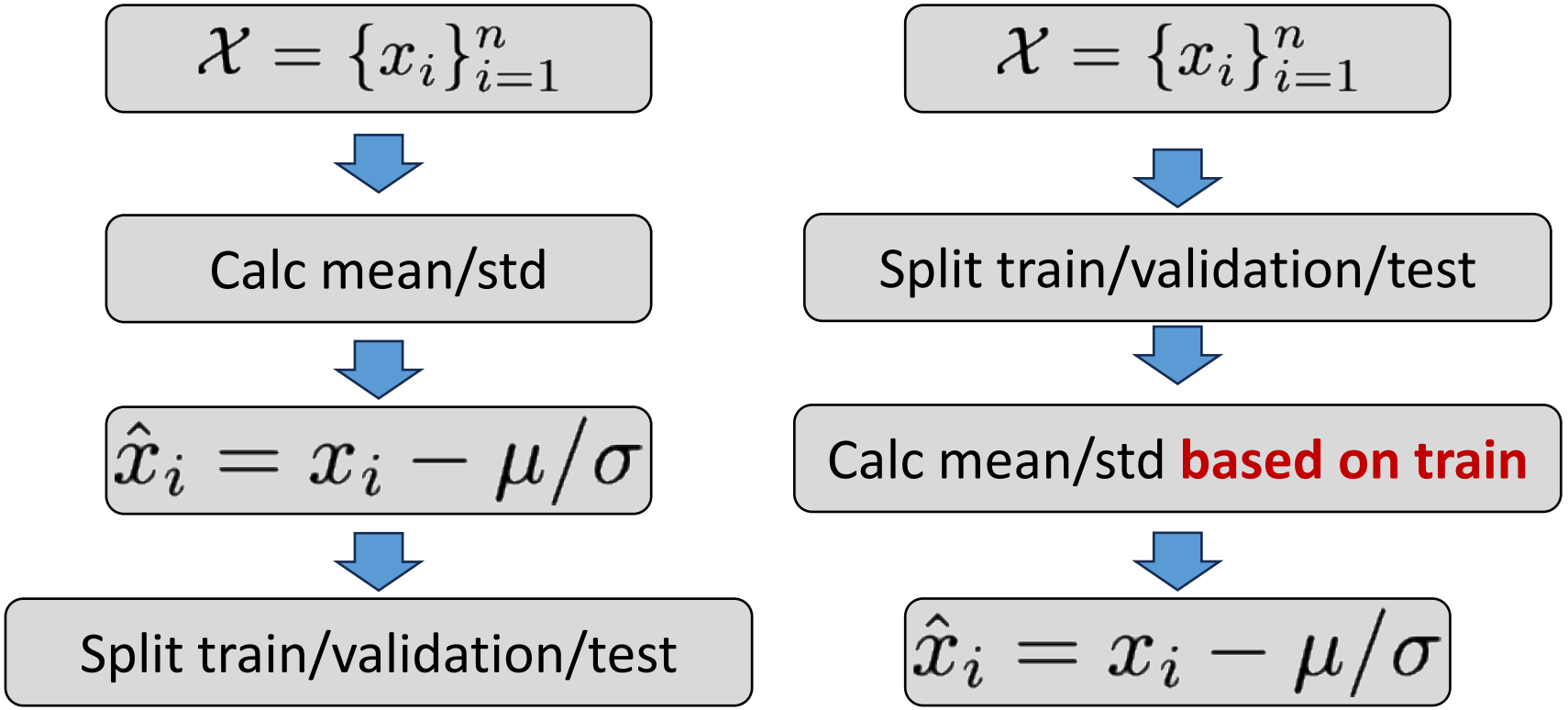
Data Leakage

- Some operations are employed over **entire dataset**, e.g. normalization.
- How should we do?



Data Leakage

- Some operations are employed over **entire dataset**, e.g. normalization.
- How should we do?



Other Format of Data Leakage

- Using test labels during model development
- Feature engineering using the entire dataset
- Oversampling before data splitting
- Selecting features using validation/test data
- Using future information to predict the past

Model Selection Pipeline

- Split the dataset
  - Training Set | Validation Set | Test Set
- Train Candidate Models
  - Learn model parameters using the **training set**
- Select the Model
  - Model architectures
  - Hyperparameters
- Retrain the Selected Model
  - Train using the available training data
- Final Evaluation
  - Evaluate **once on the test set**

Model Evaluation

What Should We Measure?

- Accuracy alone cannot describe every learning problem
- Metric depend on:
  - Task
  - Application
  - Class distribution
  - Cost of different errors
- A performance metric should reflect what matters in the target task.

Metrics for Regression Task

- Project X to Y
  - Loss function: $\ell = ||XA - Y||_2^2$
- Mean Squared Error:

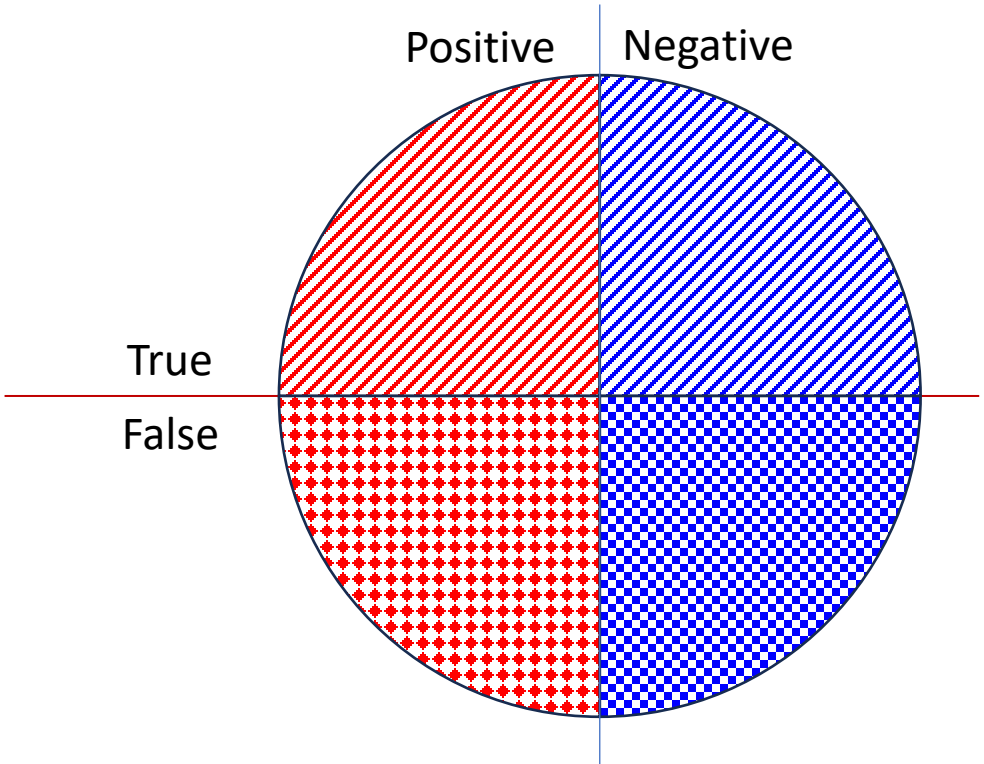
$$\text{error} = \frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2$$

- Mean Absolute Error:

$$\text{error} = \frac{1}{n}\sum_{i=1}^{n}|\hat{y}_i - y_i|$$

Confusion Matrix

- Suppose we have a dataset $\mathcal{X} = \{x_i, y_i\}_{i=1}^n$, where each data is associated with a positive/negative label, i.e., $y_i \in \{+1, -1\}$.
- We have a machine learning model $\phi : \mathcal{X} \mapsto \{+1, -1\}$
- Given x_i , the prediction can be defined: $\hat{y}_i = \phi(x_i)$



Accuracy, Precision, and Recall

- Accuracy

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

- Precision

$$\text{Precision} = \frac{TP}{TP + FP}$$

- Recall

$$\text{Recall} = \frac{TP}{TP + FN}$$

Why Accuracy Can Be Misleading?

- Consider a highly imbalanced dataset
- 1000 samples
  - 990 negative
  - 10 positive
- Suppose a model predicts “Negative” for every sample.
- Accuracy = 990 / 1000 = 99%
- But: (1). recall = 0; (2). The model detects **none of the positive samples**.
- A high accuracy does not necessarily mean that a model performs well.

Precision-Recall Trade-Off

- For example, for spam email detection:
- High Precision
  - Few legitimate emails are incorrectly classified as spam.
- High Recall
  - Few spam emails are missed.
- Improving one metric may come at the cost of the other.

Randomness in Model Evaluation

- Why can the same model obtain different results?
- Machine learning experiments often involve randomness:
  - Random train/validation/test splits
  - Random parameter initialization
  - Mini-batch sampling
  - Data augmentation
  - Other stochastic training procedures
- A single run may not provide a reliable estimate of model performance.

Thanks