Lecture 6:

Supervised Learning Evaluation

Ho-min Park

homin.park@ghent.ac.kr

powersimmani@gmail.com

Lecture Contents

Part 1: Importance and Fundamentals of Evaluation

Part 2: Regression Evaluation Metrics

Part 3: Classification Evaluation Metrics

Part 4: Model Validation Techniques

Part 1/4:

Importance and Fundamentals of Evaluation

1. Why is Evaluation Important?
2. Train vs Validation vs Test
3. Overfitting and Underfitting
4. Bias-Variance Tradeoff
5. Data Splitting Strategies
6. Stratified Sampling
7. Time Series Data Splitting
8. Preventing Data Leakage

Why is Evaluation Important?

1. Validates model performance before deployment in real-world scenarios
2. Prevents costly errors from poorly performing models
3. Guides model selection and hyperparameter tuning decisions
4. Identifies overfitting and underfitting issues early
5. **Builds trust and confidence in ML systems**
6. Ensures models generalize well to unseen data
7. Enables comparison between different algorithms objectively

Train vs Validation vs Test

Training Set

Fit model parameters

Used to learn patterns and train the model's weights and parameters

Validation Set

Tune hyperparameters

Acts as proxy for real-world performance; prevents overfitting during development

Test Set

Final unbiased evaluation

Remains completely unseen until final evaluation; never used during development

⚠ Best Practices

- Never use test data during model development
- Proper splitting ensures honest performance assessment
- Keep test set completely isolated until final evaluation

Overfitting and Underfitting

Underfitting

Too simple to capture patterns

Train Error:

Val Error:

Good Fit

Balance between bias & variance

Train Error:

Val Error:

Overfitting

Memorizes training data

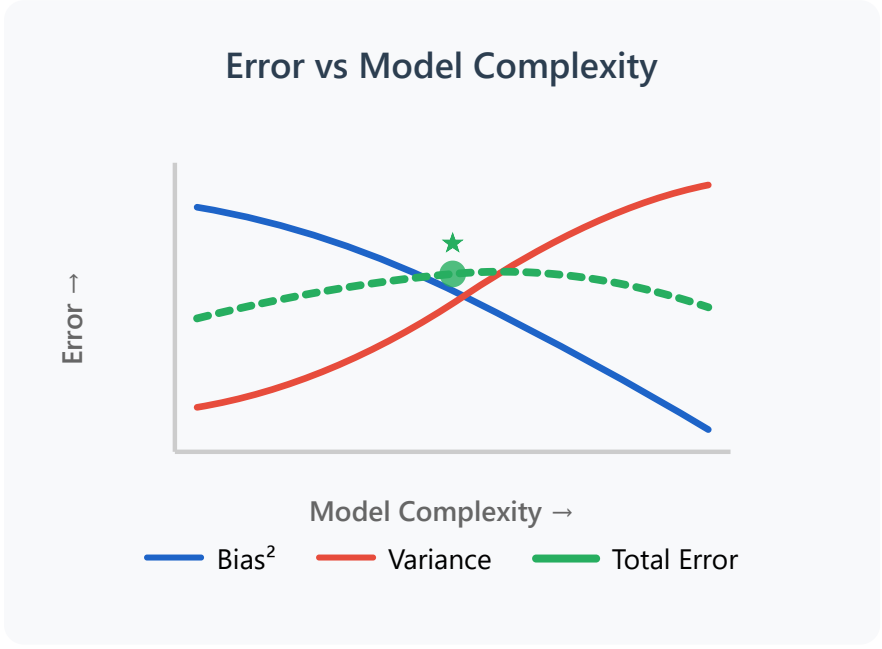
Train Error:

Val Error:

Key Insight: Finding the Sweet Spot

Model complexity should match problem complexity • Use regularization techniques to prevent overfitting • Monitor both training and validation errors

Bias-Variance Tradeoff



📊 Bias

Error from oversimplified assumptions in the model

→ *High bias leads to underfitting (too rigid)*

📈 Variance

Error from sensitivity to training data fluctuations

→ *High variance leads to overfitting (too flexible)*

🎯 **Goal: Minimize both bias and variance simultaneously**

Total Error Decomposition

$$\text{Total Error} = \text{Bias}^2 + \text{Variance} + \text{Irreducible Error}$$

Data Splitting Strategies

Random Split

Simplest approach for large, i.i.d. datasets

✓ Best for: Large, independent datasets

Stratified Split

Maintains class distribution in each subset

✓ Best for: Imbalanced classification

Time-based Split

Essential for temporal or sequential data

✓ Best for: Time series, forecasting

Group-based Split

Prevents data leakage from related samples

✓ Best for: Patient/user-level data

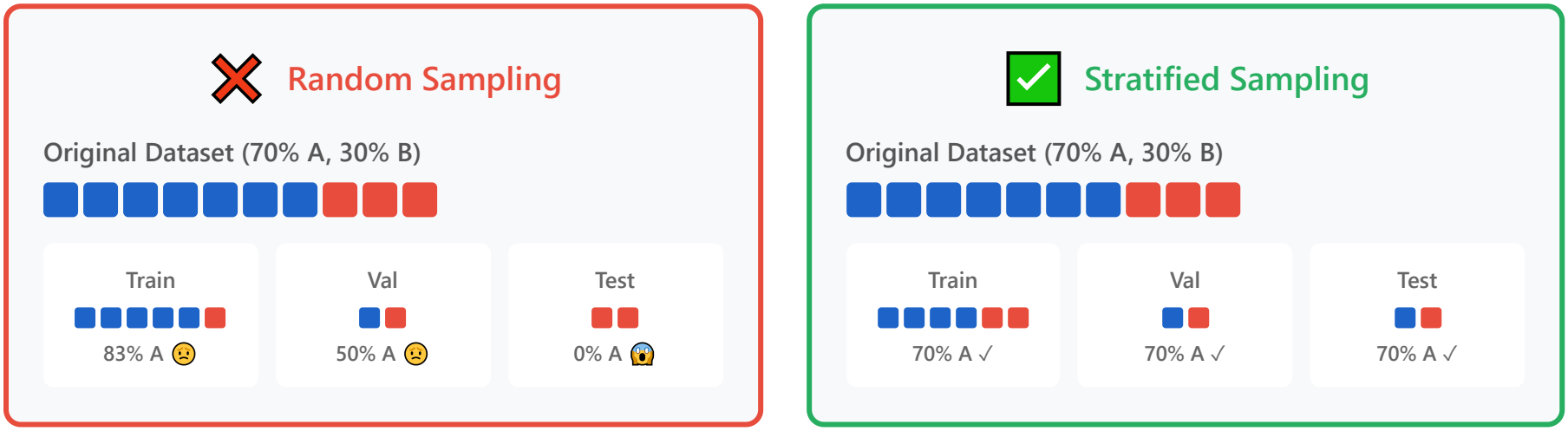
⚠ Dataset Size Consideration

Larger datasets allow smaller validation/test percentages

✓ Key Principle

All splits must be representative of population

Stratified Sampling



Why Stratified Sampling?

- 🎯 Critical for imbalanced datasets
- ⚖️ Prevents biased evaluation
- 📊 Reduces variance in estimates

Time Series Data Splitting

Random Shuffle (Wrong!)

⚠ Violates temporal order

Jan Mar Jun Sep Dec

⛔ Creates look-ahead bias

⛔ Causes data leakage

Forward Chaining (Correct)

→ Time flows forward →

Past ← Present → Future

✓ Preserves chronological order

✓ Train on past, validate on future

Two Common Approaches

Sliding Window

Fixed window moves forward in time

Expanding Window

Training set grows over time

⚡ Key Principles

- Never shuffle temporal data
- Respect temporal dependencies
- Model drift & seasonality

Preventing Data Leakage

⚠ What is Data Leakage?

Information from test set influences training, causing overly optimistic performance estimates

✗ Wrong Workflow

1. Entire Dataset
2. Normalize all data
   ⚠ Uses test statistics!
3. Split into Train/Test
4. Train Model

✓ Correct Workflow

1. Entire Dataset
2. Split into Train/Test
   ✓ Split FIRST!
3. Normalize using train stats only
4. Train Model

🚨 Common Leakage Sources

Using test set statistics

Features with future info

Duplicate samples

Improper CV folds

⚡ Consequence: Overly optimistic performance → Model fails in production

Part 2/4:

Regression Evaluation Metrics

9. MSE, RMSE, MAE
10. R^2, Adjusted R^2
11. MAPE, SMAPE
12. Residual Analysis and Diagnostics

MSE, RMSE, MAE

MSE

Mean Squared Error

$$\Sigma(y_i - \hat{y}_i)^2 / n$$

- 🎯 Squares differences
- ⚡ Heavily penalizes large errors
- 📊 Squared units

RMSE

Root Mean Squared Error

$$\sqrt{(\Sigma(y_i - \hat{y}_i)^2 / n)}$$

- 🎯 Square root of MSE
- ⚡ Same units as target
- 📊 More interpretable

MAE

Mean Absolute Error

$$\Sigma|y_i - \hat{y}_i| / n$$

- 🎯 Absolute differences
- ⚡ Treats all errors equally
- 📊 Robust to outliers

Quick Comparison

Property	MSE	RMSE	MAE
Outlier Sensitivity	High ⚠	High ⚠	Low ✓
Interpretability	Low	High ✓	High ✓
Units	Squared	Original ✓	Original ✓

🎯 Key Insight: Lower values = Better performance • Choose based on outlier importance

R^2, Adjusted R^2

R^2

Coefficient of Determination

$$R^2 = 1 - (SS_{res} / SS_{tot})$$

Proportion of variance explained

- Range: -∞ to 1 (typically 0 to 1)
- R^2 = 1: Perfect predictions
- R^2 = 0: No better than mean
- Negative R^2: Worse than baseline

Adjusted R^2

Penalized R^2

$$R^2_{adj} = 1 - [(1-R^2)(n-1)/(n-p-1)]$$

Adjusts for number of predictors

- Penalizes adding features
- Decreases if feature doesn't help
- Use for model comparison
- Better for high-dimensional data

R^2 Interpretation Scale

< 0	0.25	0.50	0.75	1.0
Terrible	Weak	Moderate	Strong	Perfect

⚡ When to Use Which?

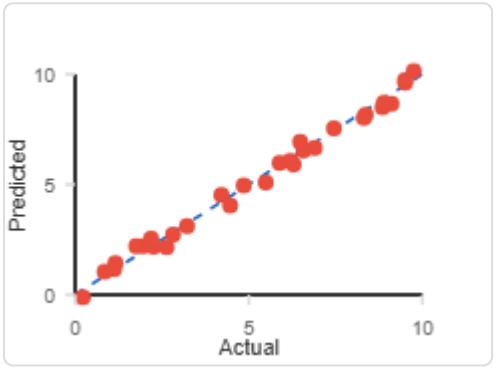
R^2
Single model evaluation

Adjusted R^2
Comparing multiple models

Predicted vs Actual Values

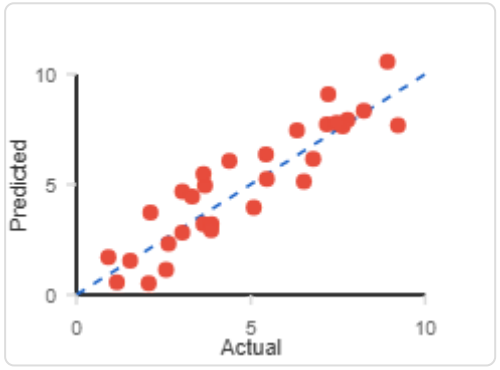
$R^2 \approx 0.95$

Strong Fit



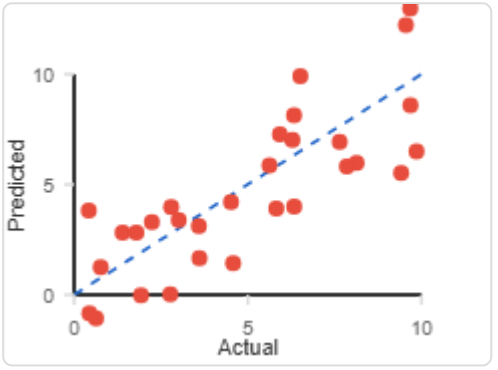
$R^2 \approx 0.50$

Moderate Fit



$R^2 \approx 0.10$

Weak Fit



MAPE, SMAPE

MAPE

Mean Absolute Percentage Error

$$\frac{100}{n} \times \sum \left| \frac{y_i - \hat{y}_i}{y_i} \right|$$

Expressed as percentage (%)

- Scale-independent metric
- Easy to interpret & communicate
- Undefined when actual = 0
- Asymmetric (penalizes over-prediction more)

SMAPE

Symmetric Mean Absolute Percentage Error

$$\frac{100}{n} \times \sum \frac{|y_i - \hat{y}_i|}{[(|y_i| + |\hat{y}_i|)/2]}$$

Bounded: 0% to 200%

- Scale-independent metric
- Easy to interpret & communicate
- ✓ More stable than MAPE
- Symmetric (treats over/under equally)

Key Differences

Feature	MAPE	SMAPE
Handles Zero Values	✗	✓

Symmetric Treatment	✗	✓
Bounded Range	0% - ∞	0% - 200%
Stability	Lower	Higher

💼 Why Use Percentage Metrics?

Compare across different scales

Easy business communication

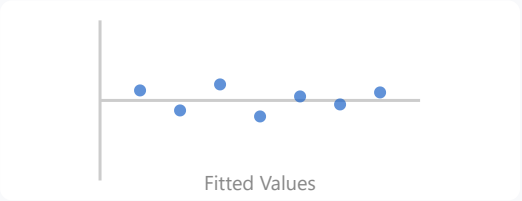
Intuitive interpretation

Residual Analysis and Diagnostics

Residuals: Differences between observed and predicted values

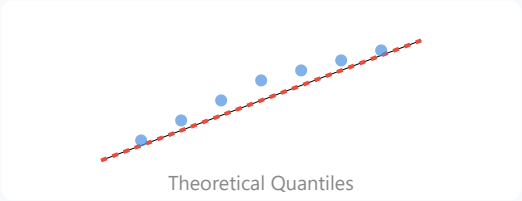
$$\text{Residual} = y - \hat{y}$$

Residual Plot



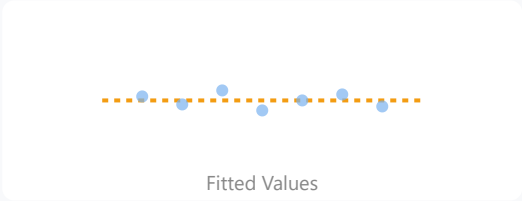
Random scatter around zero indicates good fit

Q-Q Plot



Checks normal distribution of residuals

Scale-Location Plot



Detects heteroscedasticity (non-constant variance)

Ideal vs Problematic Patterns

✓ Good Pattern

Random scatter, constant variance

✗ Bad Pattern

Curved pattern, non-linearity

🔍 What to Check

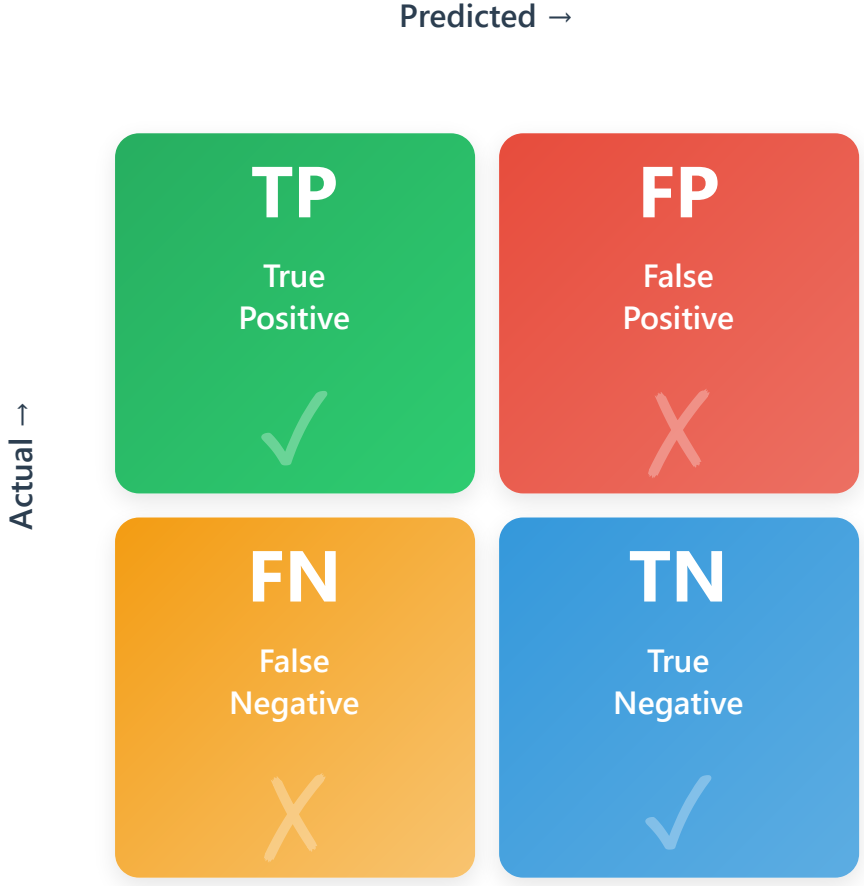

Non-linearity
Heteroscedasticity
Outliers
Normal distribution

Part 3/4:

Classification Evaluation Metrics

13. Confusion Matrix
14. Accuracy and Its Limitations
15. Precision and Recall
16. F1 Score and Fβ
17. ROC Curve and AUC
18. Precision-Recall Curve
19. Multi-class Metrics

Confusion Matrix



True Positive (TP)

Correctly predicted positive cases

False Positive (FP)

Incorrectly predicted as positive

Type I Error

False Negative (FN)

Incorrectly predicted as negative

Type II Error

True Negative (TN)

Correctly predicted negative cases

🎯 Foundation: All classification metrics derive from these four components

Accuracy and Its Limitations

📊 Definition

Accuracy = (TP + TN) / Total

Percentage of correct predictions

✓ Advantages

- ✓ Simple to calculate
- ✓ Easy to interpret
- ✓ Intuitive metric (0-100%)

⚠ The Accuracy Paradox: Rare Disease Detection

Dataset

1%

Disease prevalence
(99% healthy)

→

Naive Model

99%

"Always predict healthy"
Useless accuracy!

⚖

💰

🎯

Imbalanced Data

Misleading with skewed class distribution

Different Costs

Ignores varying costs of FP vs FN

No Error Context

Doesn't reveal error patterns

⚡ **Critical: Always consider class distribution before relying on accuracy alone**

Precision and Recall

Precision

Positive Predictive Value

```
Precision = TP / (TP + FP)
```

"When model says positive, how often is it right?"

Recall

Sensitivity / True Positive Rate

```
Recall = TP / (TP + FN)
```

"Of all actual positives, how many did we find?"

Focus Comparison

Precision Focus

Predicted Positives

Among what we predicted as positive, how many are correct?

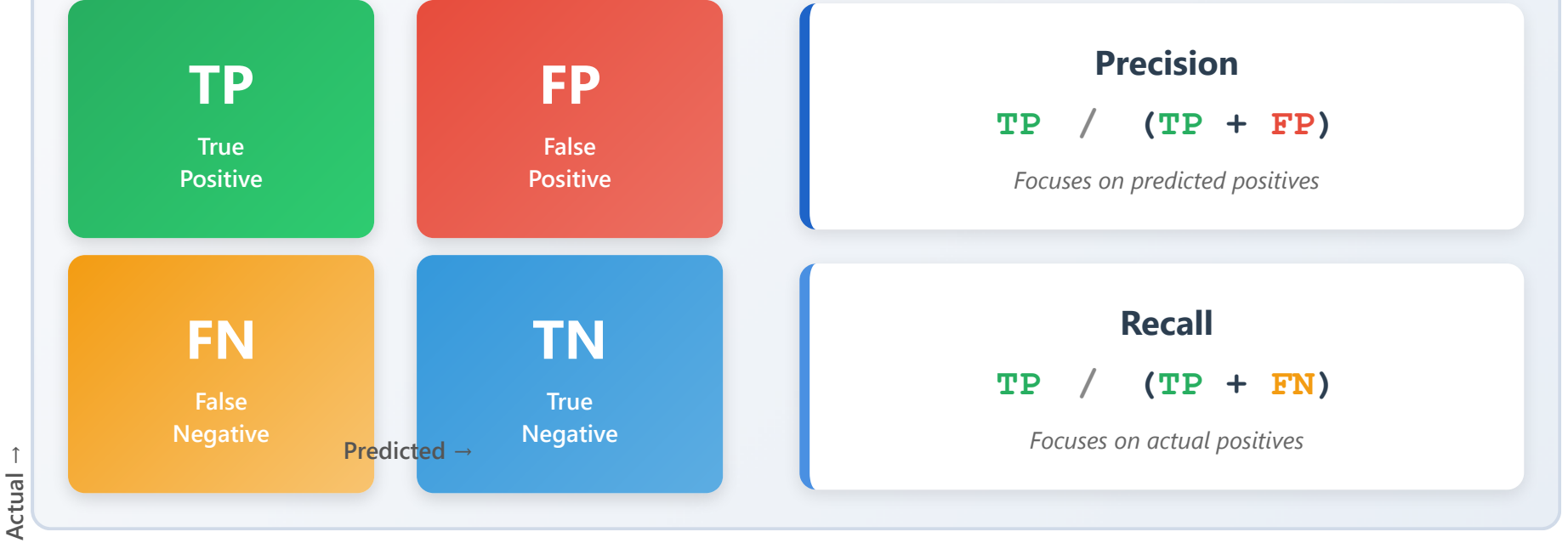
VS

Recall Focus

Actual Positives

Among what is actually positive, how many did we catch?

📊 How They're Calculated from Confusion Matrix



⚖️ **The Precision-Recall Tradeoff**

Improving precision often decreases recall and vice versa

High Precision Needed

When false positives are costly

Example: Spam filter

High Recall Needed

When false negatives are costly

Example: Cancer screening

F1 Score and Fβ

F1 Score

Harmonic Mean

$$F1 = 2 \times P \times R / (P + R)$$

Balances Precision & Recall

- Single metric (0 to 1)
- Equal weight to P and R
- Penalizes extreme values
- Use when balance needed

Fβ Score

Weighted F-Score

$$F\beta = (1+\beta^2) \times P \times R / (\beta^2 P + R)$$

Allows emphasis control

- β controls weight
- β > 1: Favor recall
- β < 1: Favor precision
- β = 1: Standard F1

β Parameter Spectrum

F0.5

β = 0.5

Emphasizes Precision

F1

β = 1

Balanced

F2

β = 2

Emphasizes Recall

Common Fβ Variants

F0.5

Precision > Recall

Minimize false alarms

F1

Precision = Recall

Equal importance

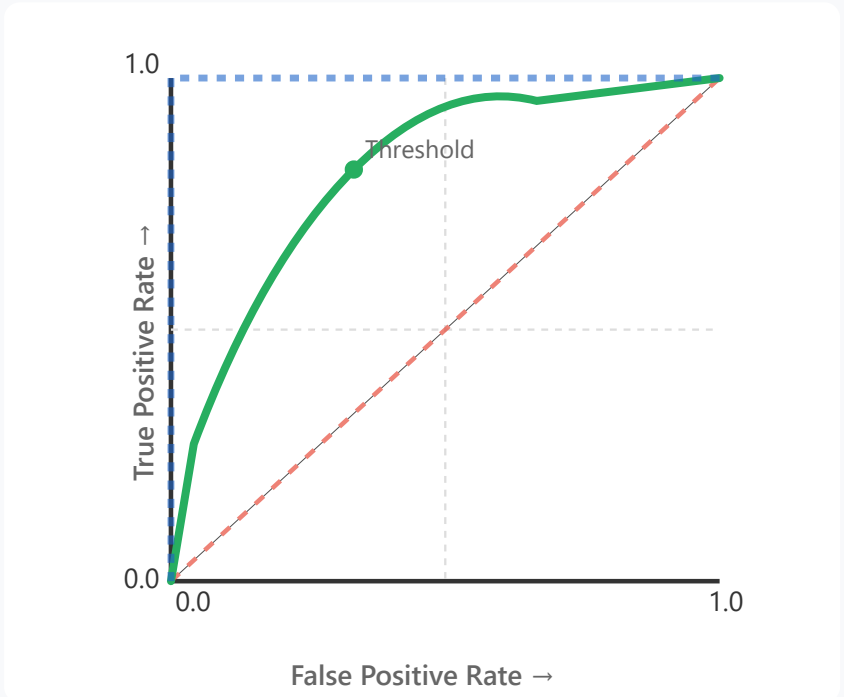
F2

Recall > Precision

Minimize misses

ROC Curve and AUC

ROC Curve Visualization



Perfect (AUC=1.0)

Good (AUC≈0.85)

Random (AUC=0.5)

📈 ROC Curve

Plots TPR vs FPR at various classification thresholds. Shows model performance across all possible threshold settings.

📊 AUC Score

Area Under the ROC Curve. Single number (0.5 to 1.0) summarizing overall classification performance.

AUC Interpretation Scale

0.5	0.7	0.8	0.9	1.0
Random	Fair	Good	Excellent	Perfect

✨ Key Advantages

Threshold-independent metric | Robust to class imbalance | Easy model comparison

ROC Curve Construction: Step-by-Step Guide

1 Predicted Probabilities and Actual Labels

Sample	Predicted Probability	Actual Label
A	0.95	1 (Positive)
B	0.85	1 (Positive)
C	0.75	0 (Negative)
D	0.65	1 (Positive)
E	0.55	0 (Negative)
F	0.45	0 (Negative)
G	0.35	1 (Positive)
H	0.15	0 (Negative)

2 Calculate TPR and FPR at Each Threshold

Threshold = 0.80

Confusion Matrix

	A+	A-
P+	TP 2	FP 0
P-	FN 2	TN 4

TPR

$2/(2+2)=0.5$

Threshold = 0.60

Confusion Matrix

	A+	A-
P+	TP 3	FP 1
P-	FN 1	TN 3

TPR

$3/(3+1)=0.75$

Threshold = 0.40

Confusion Matrix

	A+	A-
P+	TP 4	FP 2
P-	FN 0	TN 2

TPR

$4/(4+0)=1.0$

50%

FPR

0/(0+4)=0.0

(0.00, 0.50)

75%

FPR

1/(1+3)=0.25

25%

(0.25, 0.75)

100%

FPR

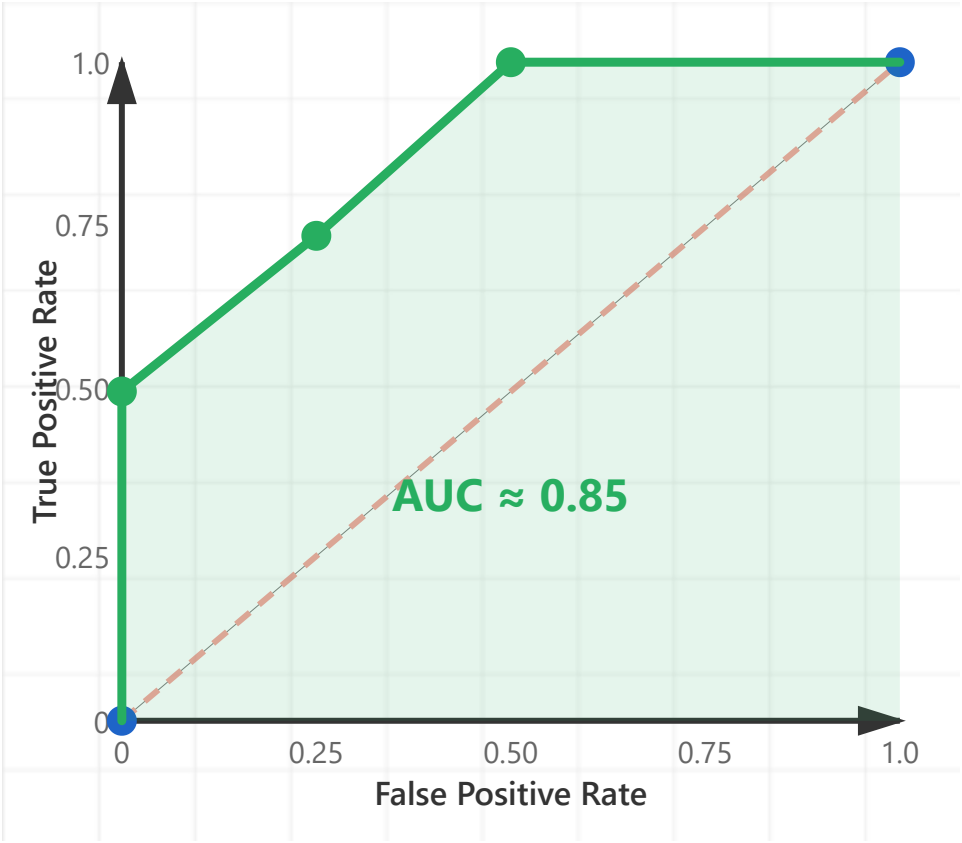
2/(2+2)=0.5

50%

(0.50, 1.00)

3 Connect Points to Create ROC Curve

- Calculate (FPR, TPR) coordinates for all thresholds
- Sort points by FPR in ascending order
- Connect points with lines to complete ROC curve
- Evaluate model performance using AUC



💡 Key Summary

Step 1:
Predicted Probability과 Actual Label 준비

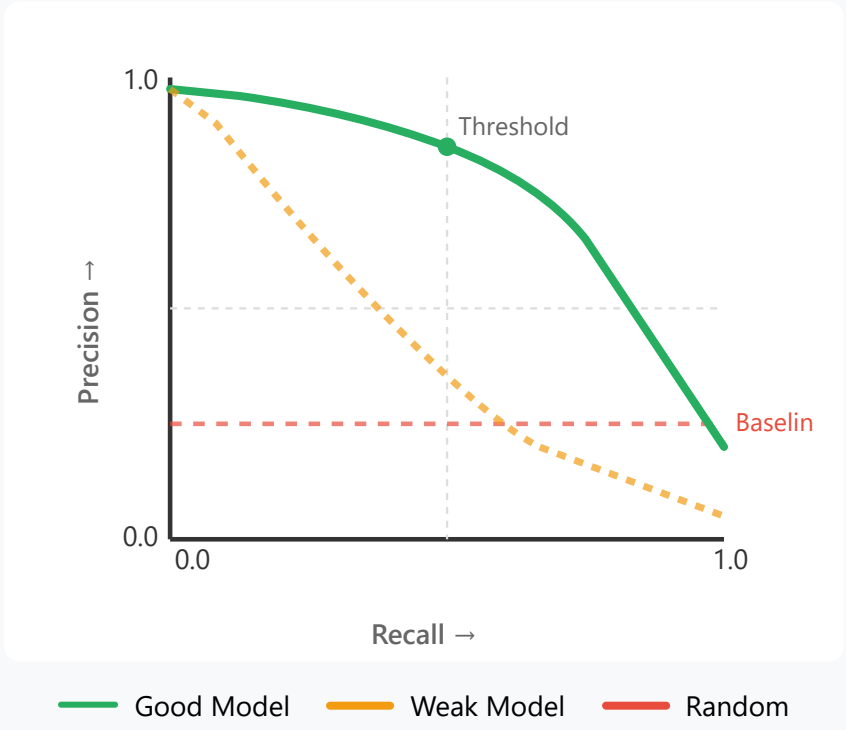
Step 2:
Calculate TPR and FPR at each threshold

Step 3:
Connect coordinates to create curve

Result:
Higher AUC indicates better model

Precision-Recall Curve

PR Curve Visualization



📊 PR Curve

Plots Precision vs Recall at different thresholds. High area indicates both metrics are maintained well.

⚖️ vs ROC Curve

More informative for imbalanced datasets. Better reveals performance on minority class.

Key Insights

Steep Decline
Difficult P-R tradeoff

High Area
Strong classifier

Baseline
= Positive class %

Complement
Use with ROC

💡 Best Used For

Fraud Detection

Rare Disease Detection

Imbalanced Datasets

How to Construct a PR Curve: Step-by-Step

1 Model Predictions & Thresholds

Sample	Score	True Label	@ 0.9	@ 0.7	@ 0.5	@ 0.3
A	0.95	✓ Positive	✓	✓	✓	✓
B	0.85	✓ Positive	✗	✓	✓	✓
C	0.65	✗ Negative	✗	✗	✓	✓
D	0.60	✓ Positive	✗	✗	✓	✓
E	0.40	✓ Positive	✗	✗	✗	✓
F	0.25	✗ Negative	✗	✗	✗	✗

At each threshold, predict Positive if score ≥ threshold

2 Calculate Precision & Recall at Each Threshold

Threshold = 0.9

TP:	1	(A)
FP:	0	(none)
FN:	3	(B,D,E)

Precision: 1.00
Recall: 0.25

Threshold = 0.7

TP:	2	(A,B)
FP:	0	(none)
FN:	2	(D,E)

Precision: 1.00
Recall: 0.50

Threshold = 0.5

TP:	3	(A,B,D)
FP:	1	(C)
FN:	1	(E)

Precision: 0.75
Recall: 0.75

Threshold = 0.3

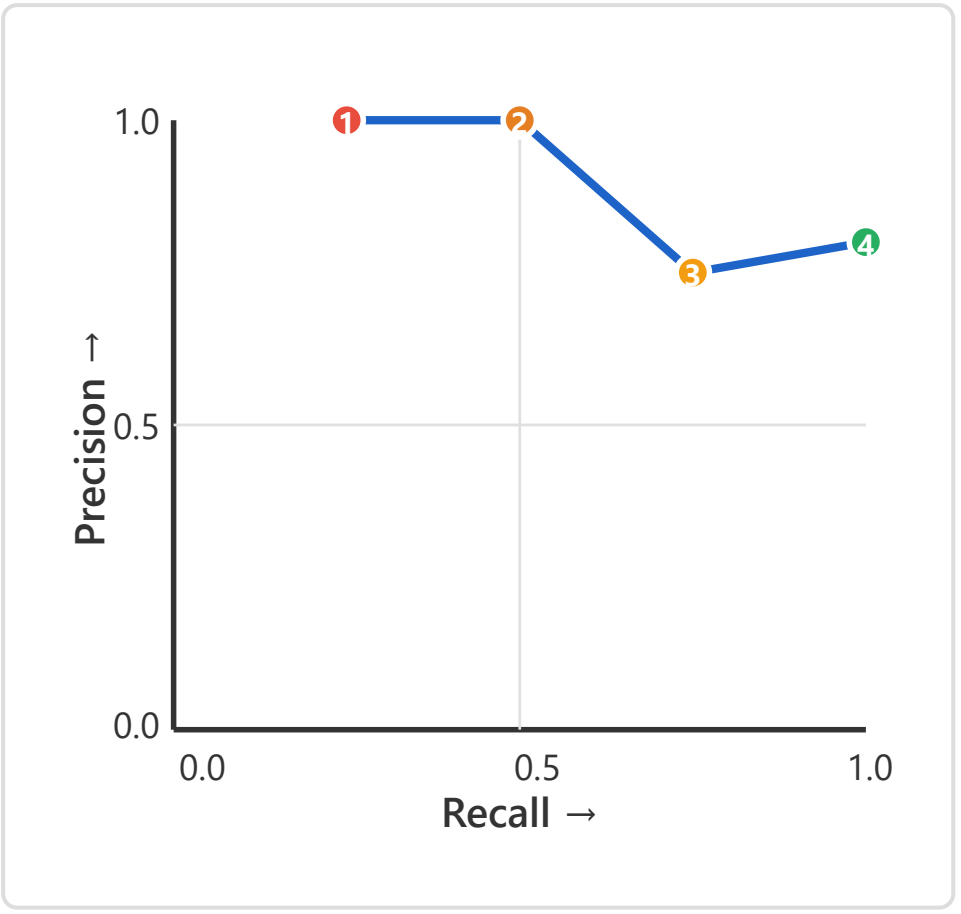
TP:	4	(A,B,D,E)
FP:	1	(C)
FN:	0	(none)

Precision: 0.80
Recall: 1.00

Precision = `TP / (TP + FP)` Recall = `TP / (TP + FN)`

3 Plot Points & Draw the Curve

1. `(0.25, 1.00)` Threshold 0.9
2. `(0.50, 1.00)` Threshold 0.7
3. `(0.75, 0.75)` Threshold 0.5
4. `(1.00, 0.80)` Threshold 0.3



Plot (Recall, Precision) coordinates for each threshold and connect them to form the PR curve

Key Insight

As we lower the threshold → More samples are predicted as Positive → Recall increases (capturing all positive samples) → But False Positives also increase → Precision decreases. The PR curve visualizes this trade-off.

Multi-class Metrics

Macro-Average

$$\text{avg}(\text{metric}_1, \text{metric}_2, \ldots)$$

Calculate metric for each class, then average. Equal class weight.

Micro-Average

$$\text{metric}(\Sigma\ TP, \Sigma\ FP, \Sigma\ FN)$$

Aggregate all classes first, then calculate. Favors majority.

Weighted-Average

$$\Sigma(\text{weight} \times \text{metric})$$

Weight by class support. Balances macro & micro.

Example: F1 Score Calculation

Class A (100 samples)	F1 = 0.90
Class B (50 samples)	F1 = 0.70
Class C (50 samples)	F1 = 0.60

Macro-Average
(0.90 + 0.70 + 0.60) / 3 → **0.73**

Micro-Average
Aggregate TP/FP/FN → **0.82**

Weighted-Average
(100×0.90+50×0.70+50×0.60)/200 → **0.78**

🔑 Key Insights

Macro
Treats all classes equally, good for balanced view

Micro
Similar to accuracy, biased to majority class

Weighted
Balanced approach considering class sizes

Confusion Matrix (5 Classes)

	C0	C1	C2	C3	C4
C0	**45**	3	2	0	0
C1	2	**28**	5	0	0
C2	1	4	**18**	2	0
C3	0	0	3	**12**	0
C4	0	0	1	1	**8**

Rows: Actual | Columns: Predicted | Total: 135 samples

Per-Class Metrics

Class	Support	Precision	Recall	F1
C0	50	0.938	0.900	**0.918**
C1	35	0.800	0.800	**0.800**
C2	25	0.621	0.720	**0.667**
C3	15	0.800	0.800	**0.800**
C4	10	1.000	0.800	**0.889**

Example Calculation for C0:

- TP = 45, FP = 3, FN = 5
- Precision = 45/(45+3) = 0.938
- Recall = 45/(45+5) = 0.900
- F1 = 2×(0.938×0.900)/(0.938+0.900) = 0.918

📊 Final Averaged Metrics

Macro-Average F1

0.815

(0.918 + 0.800 + 0.667
+ 0.800 + 0.889) / 5

Micro-Average F1

0.822

ΣTP = 111, ΣFP = 24, ΣFN = 24
F1 = 2×111 / (2×111+24+24)

Weighted-Average F1

0.835

(50×0.918 + 35×0.800 + 25×0.667
+ 15×0.800 + 10×0.889) / 135

💡 **Interpretation:** Weighted (0.835) > Micro (0.822) > Macro (0.815) indicates the model performs better on majority classes (C0, C1) than on minority classes

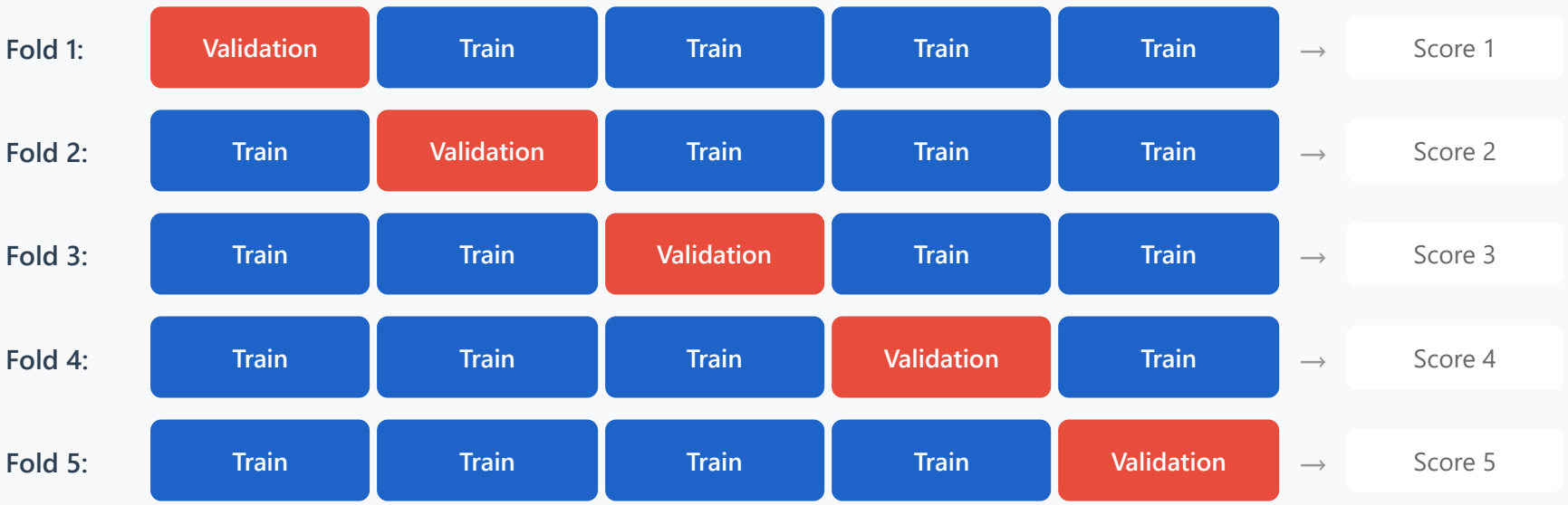
Part 4/4:

Model Validation Techniques

20. K-fold Cross Validation
21. Stratified K-fold
22. Leave-One-Out CV
23. Bootstrapping
24. Hyperparameter Tuning
25. Model Selection Strategy

K-fold Cross Validation (K=5)

🔄 Validation Process



📊 Final Score

$$\text{Mean} = \Sigma(\text{scores}) \;/\; K$$

Average performance across all K folds for robust estimate

📈 Standard Deviation

$$\text{Std} = \sqrt{\Sigma(\text{score} - \text{mean})^2 \;/\; K}$$

Variance across folds indicates model stability

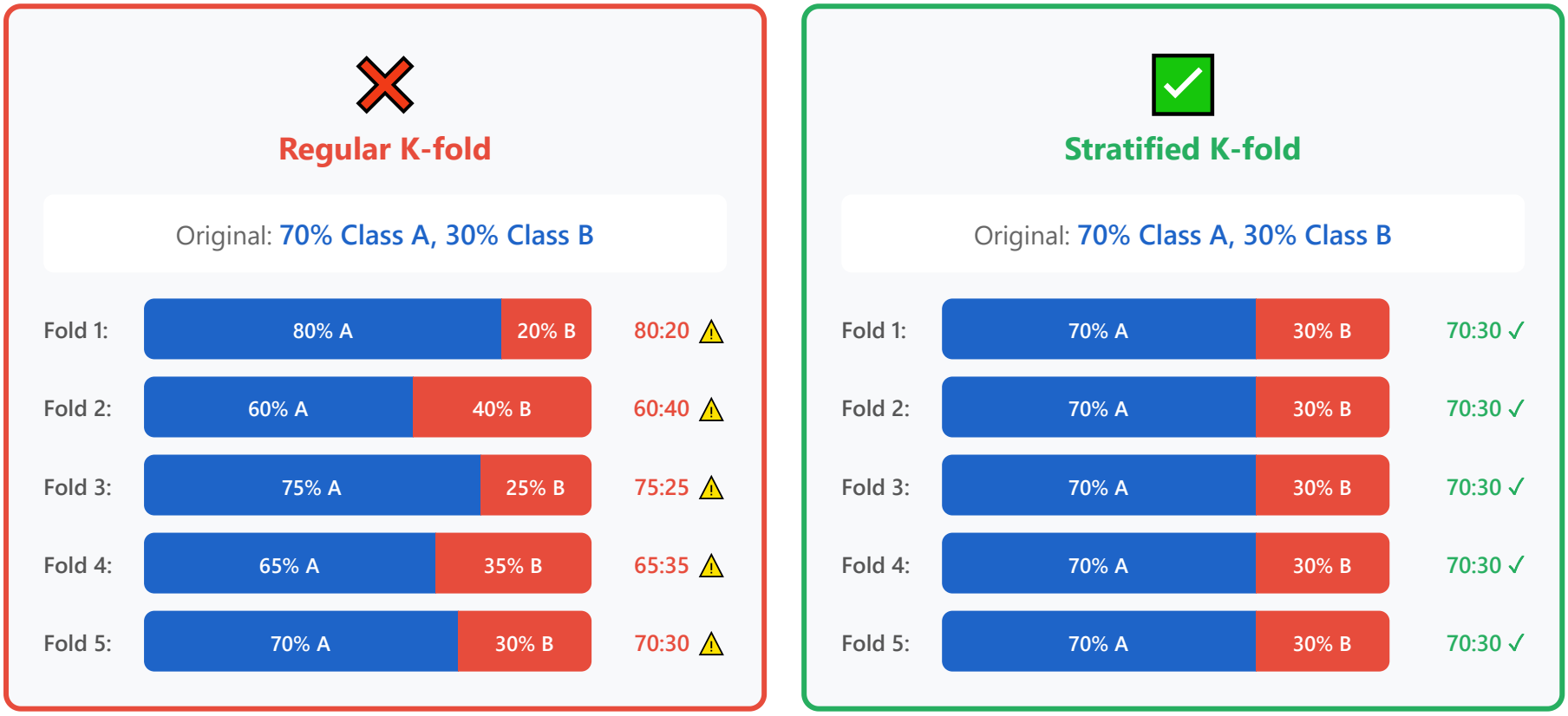
✨ Key Benefits

Every sample used for both training & validation

More reliable performance estimate

Reduces variance in evaluation

Stratified K-fold Cross Validation



Key Benefits

- Maintains class distribution in each fold
- Reduces variance in performance estimates

🎯 Essential for imbalanced datasets

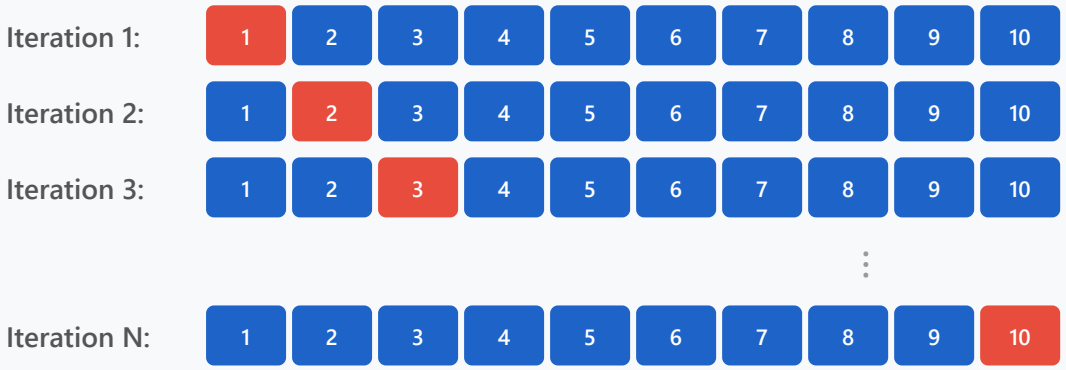
✓ More reliable for classification tasks

💡 When to Use

Always prefer stratified K-fold for classification problems, especially with imbalanced classes

Leave-One-Out Cross Validation (LOOCV)

🔄 Process: K = N (where N = dataset size)



📊 Final Performance

$$\text{Performance} = \Sigma(\text{score}_1 + \text{score}_2 + \ldots + \text{score}_n) \;/\; N$$

Average of N individual predictions where N = total samples

✅ Advantages

- 📊 Maximum use of data (N-1 samples for training)
- 🎯 Nearly unbiased estimate of model performance

❌ Disadvantages

- ⏱️ Computationally expensive (N model trainings)
- 📈 High variance in performance estimates

✓ Deterministic (no randomness in splits)

⚠ Not practical for large datasets

💡 When to Use

Best for small datasets where data is precious • Avoid for large datasets due to computational cost

Bootstrapping

Bootstrap Process

Original Data

N samples

→

Sample with Replacement

Create B bootstrap samples

→

Train Models

One model per sample

→

Aggregate

Mean & confidence interval

Sampling with Replacement (Example)

Original (N=10)

1	2	3	4	5
6	7	8	9	10

→

Bootstrap Samples

Sample 1:	3	3	7	1	9	2	2	5	10	8
Sample 2:	5	6	1	1	4	8	9	9	3	7
Sample B:	...									

Key Feature

Statistical Power

Samples with replacement: same data point can appear multiple times in a bootstrap sample

Provides confidence intervals and variance estimates for model performance

💡 Common Uses

- Estimate model uncertainty
- Build ensemble methods (Bagging)
- Small dataset validation

Hyperparameter Tuning

Grid Search

Exhaustively searches through specified parameter combinations

✓ Advantages:

Comprehensive, guaranteed to find best in grid

Random Search

Randomly samples from parameter space for specified iterations

✓ Advantages:

More efficient, explores wider range

Bayesian Optimization

Uses probabilistic model to guide search toward promising regions

✓ Advantages:

Smart search, fewer iterations needed

📋 Typical Tuning Workflow

1. Define parameter space & ranges
2. → Choose search strategy
3. → Use cross-validation for each config
4. → Select best parameters
5. → Evaluate on test set

🎯 Best Practices

Use validation set, never test set for tuning

Combine tuning with cross-validation

Start broad, then narrow search range

Consider computational budget

Log-scale for learning rates

✓

Automate with tools (sklearn, optuna)

Model Selection Strategy

🎯 Complete Model Selection Workflow

1. Split data into train/val/test
2. Select validation method (K-fold, etc.)
3. Train multiple models with CV
4. Tune hyperparameters on validation
5. Select best model
6. Final evaluation on test set

📊 Recommended Data Split

Training: Model learning • Validation: Hyperparameter tuning & model selection • Test: Final unbiased evaluation

Never Touch Test Set

Test set must remain unseen until final evaluation

Use Cross-Validation

Apply CV on training set for robust estimates

Consider Data Balance

Use stratified splits for imbalanced datasets

Monitor Overfitting

Compare train vs validation performance

✨ Key Principles

- 🎯 Choose metrics aligned with business goals
- 📊 Use multiple metrics for comprehensive view
- ⏱️ Balance performance with computational cost
- 🔍 Analyze errors to understand model behavior
- 📝 Document all decisions and experiments
- 🔄 Iterate based on validation results

📋 Model Performance Comparison

Model	Best Hyperparameters	Accuracy	Precision	Recall	F1 Score	Training Time	Status
Random Forest	`n_estimators: 200` `max_depth: 15` `min_samples_split: 5`	0.942	0.938	0.945	0.941	2.3s	✓ Best
XGBoost	`learning_rate: 0.1` `max_depth: 8` `n_estimators: 150`	0.935	0.932	0.937	0.934	1.8s	2nd
SVM (RBF)	`C: 10` `gamma: 0.01` `kernel: rbf`	0.918	0.915	0.920	0.917	5.1s	-
Logistic Regression	`C: 1.0` `penalty: l2` `solver: lbfgs`	0.887	0.883	0.891	0.887	0.5s	⚡ Fastest
Decision Tree	`max_depth: 12` `min_samples_split: 10` `criterion: gini`	0.865	0.861	0.869	0.865	0.3s	-

💡 All metrics evaluated on validation set using 5-fold cross-validation. Best model selected based on F1 score and generalization performance.

Thank you

Ho-min Park

homin.park@ghent.ac.kr

powersimmani@gmail.com