Model Evaluation and Optimization

Oct 7th, 2025

Agenda

- Motivation
- Performance Metrics Classification
- Performance Metrics Regression
- Cross Validation
- Hyperparameter Tuning

Announcements:

- March 11th: NO CLASS
  - Work on projects (manuscript, midterm presentation)
  - Midterm presentation handout will be sent after class

- March 13th: Midterm presentation (15% final grade), 1st draft of manuscript (15%)
  - Conocophillips Visit (**network** - pizza)!
  - **12 minutes presentations**
  - Be on time (be professional)!

Motivation

1. **A great model isn't just about high accuracy!**
- A model that performs well on training data but fails in the real world is **useless**.
- How do we ensure our model generalizes well? **Evaluation metrics & cross-validation!**

2. **Optimizing for the Best Model**
- Choosing the right **hyperparameters** can **boost performance significantly**.
- Regularization techniques prevent models from learning noise instead of patterns.

3. **Real-World Impact**
- A poorly evaluated model in medicine, energy, or self-driving cars can have **serious consequences**.
- Understanding model evaluation helps us build **robust, fair, and trustworthy** AI systems.

Performance Metrics

What Are Performance Metrics?

Performance metrics are **quantitative measures** used to assess how well a machine learning model performs. They help us understand:

- **Accuracy** vs. **Error**
- **Model reliability**
- **Suitability for real-world applications**

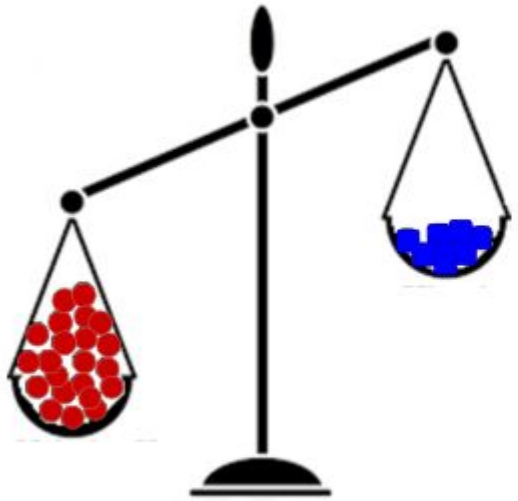
Why Are They Important?

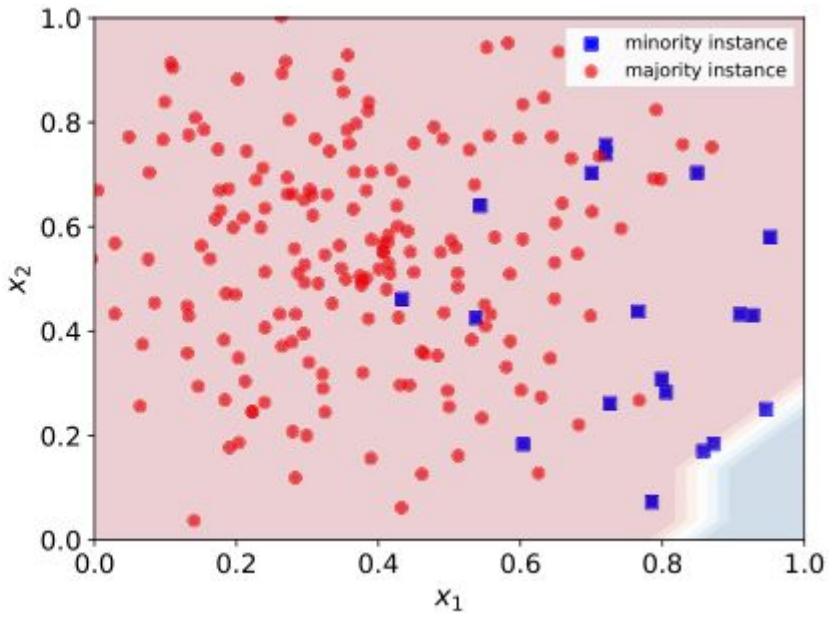
Avoid misleading results – Accuracy alone can be deceptive, especially with imbalanced data.
Compare models effectively – Helps in selecting the best model for deployment.
Optimize model performance – Guides hyperparameter tuning and improvements.
Ensure fairness & trustworthiness – Some models work better for certain groups, and metrics help identify biases.

Performance Metrics Classification

Why do we need specific metrics?

- Classification models predict **categories** (e.g., spam or not spam).
- Accuracy alone can be **misleading**, especially with imbalanced data.
- We need multiple metrics to understand model performance from different angles.

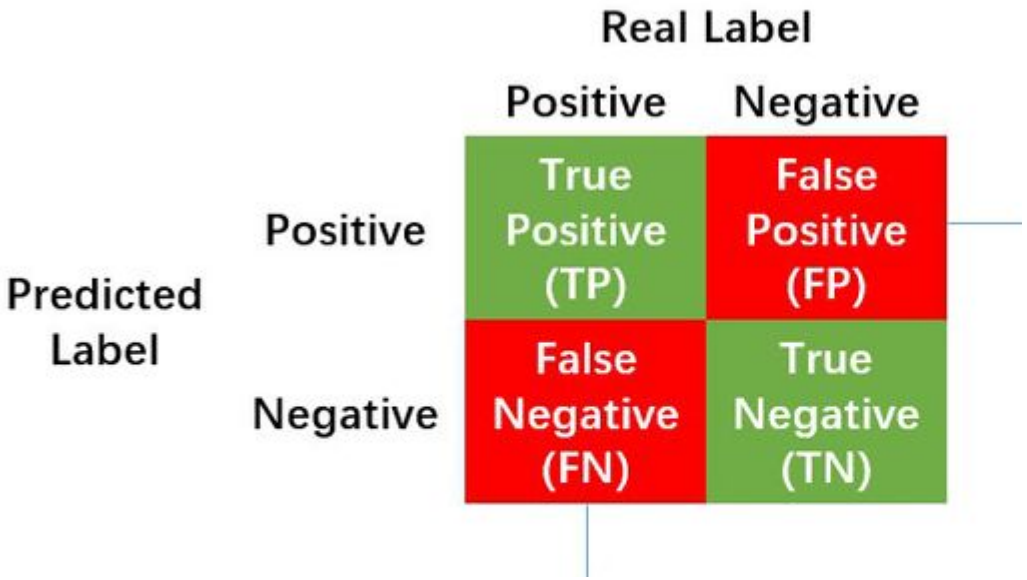




Performance Metrics Classification

Key Classification Metrics:

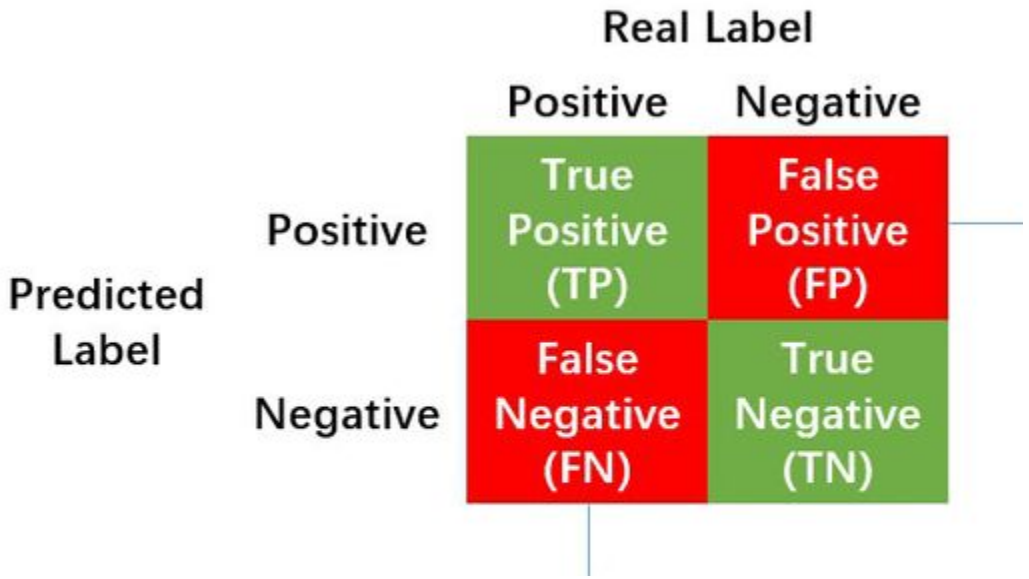
1. Confusion Matrix



Performance Metrics Classification

Key Classification Metrics:

1. Accuracy



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

- Measures the percentage of correctly classified instances.
- **Misleading when classes are imbalanced.**

Why is accuracy misleading for imbalanced classes?

Performance Metrics Classification

Key Classification Metrics:

1. Accuracy

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

✅ Works well when **class distribution is balanced**.
⚠️ **Fails when one class dominates!**

- **Dataset:** 10,000 transactions
- **Fraud Cases:** 100 (1%)
- **Legitimate Transactions:** 9,900 (99%)
- **Naïve Model:** Predicts **every transaction as "legitimate"**

Accuracy Calculation:

- **TP = 0**, **FN = 100** (missed all fraud cases)
- **TN = 9,900**, **FP = 0**

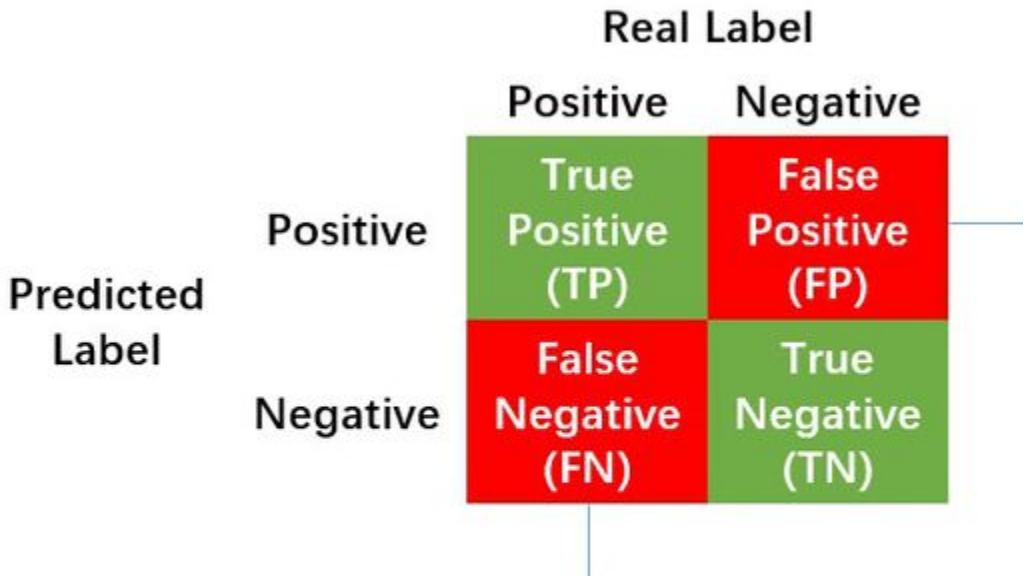
Accuracy=9,900/10,000=99%

99% accuracy, but the model is useless! It **never detects fraud**.

Performance Metrics Classification

Key Classification Metrics:

2. Precision



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

- Of all predicted positives, how many are actually correct?
- Useful when false positives are costly (e.g., spam detection, fraud detection).
- **A model can achieve high precision by predicting very few positives!**

Why is a model can achieve high precision by predicting very few positives?

Performance Metrics Classification

Key Classification Metrics:

2. Precision

- **Dataset:** 1,000 patients
- **Actual Sick Patients:** 100
- **Model's Predictions:** Only predicts **10 patients as sick**
  - 9 are actually sick (**TP = 9**)
  - 1 is a false positive (**FP = 1**)

Precision Calculation:

Precision=99+1=90%

High precision, but...

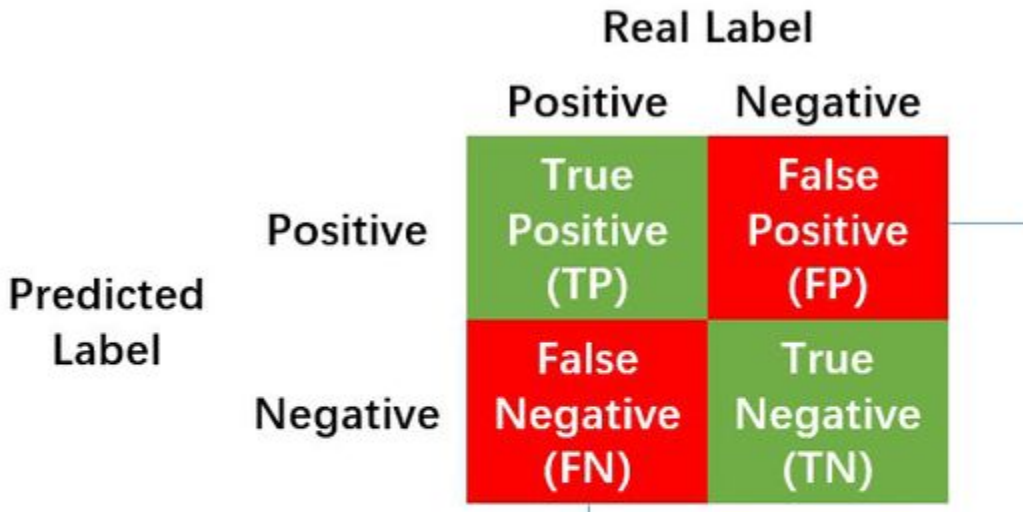
- The model only **caught 9 out of 100 sick patients → Missed 91 sick cases!**
- **Recall is terrible** → Patients remain undiagnosed.

Focusing **only on precision** can lead to **high false negatives** (missed cases). For critical applications, always **balance precision with recall**!

Performance Metrics Classification

Key Classification Metrics:

3. Recall



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

- Of all predicted positives, how many we correctly predict?
- Important for **minimizing false negatives** (e.g., disease detection).

Useful when missing positive cases is costly (e.g., medical diagnosis, fraud detection).
Problem: A model can achieve high recall by predicting everything as positive!

Why is a model can achieve high recall by predicting everything as positive?

Performance Metrics Classification

Key Classification Metrics:

3. Recall

- **Dataset:** 1,000 emails
- **Actual Spam:** 100
- **Model's Predictions:** Predicts **all 1,000 emails as spam**
  - **TP = 100** (caught all spam)
  - **FP = 900** (marked 900 good emails as spam)

Recall Calculation:

$$\text{Recall} = \frac{100}{100+0} = 100\%$$

Perfect recall, but…

- **Precision is terrible** → 900 good emails falsely labeled as spam.

Precision-Recall Tradeoff

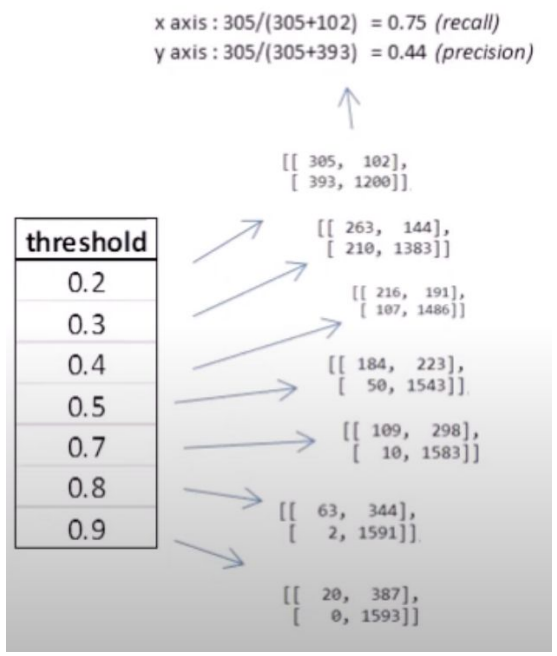
- **Precision** (Positive Predictive Value) → How many predicted positives are actually correct?
- **Recall** (Sensitivity) → How many actual positives were correctly predicted?

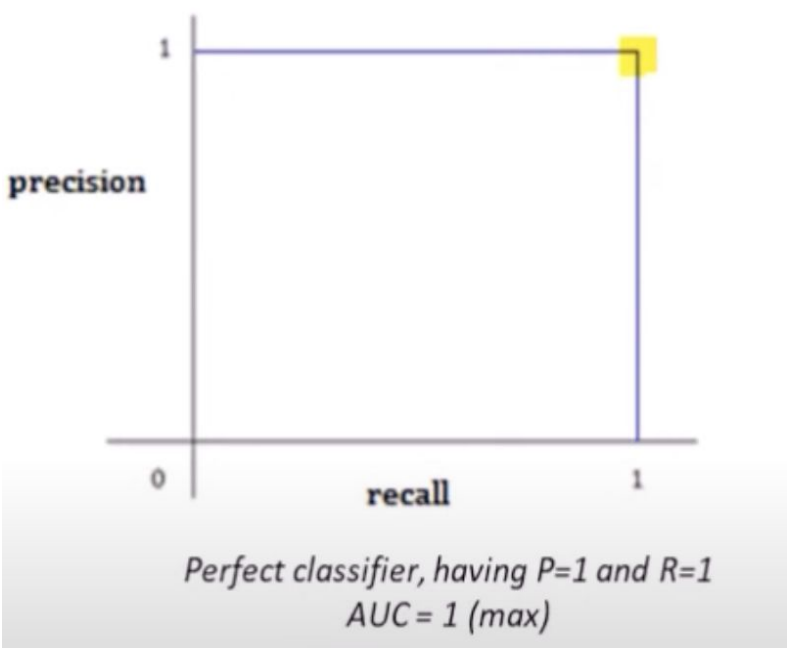
Key Concept:

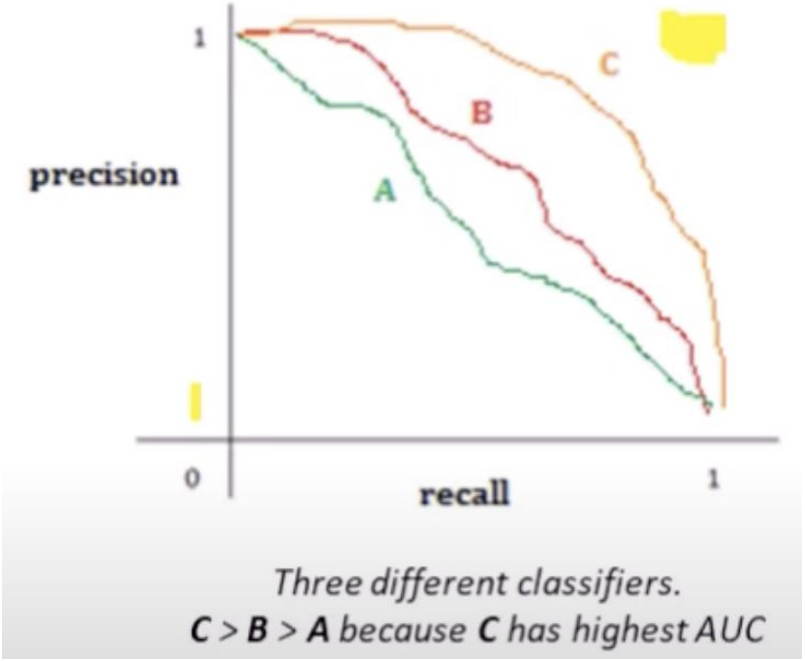
- Increasing **Recall** often **lowers Precision** (more false positives).
- Increasing **Precision** often **lowers Recall** (more false negatives).
- The goal is to **find the best balance** for a given application.

Precision-Recall Tradeoff : PR Curve

- A classifier (e.g., logistic regression, neural network) outputs **probabilities** rather than direct class labels.
- By changing the probability **threshold** (e.g., from 0.5 to 0.7), we adjust how the model classifies an instance as "positive."







F1 - Score

- Harmonic mean of precision and recall.
- Formula: **2 × (Precision × Recall) / (Precision + Recall)**
- Best when you need **a balance between precision and recall**.

Can I Always Use F1-Score Instead of Precision & Recall?

No, not always! F1-Score is a trade-off, and in some cases, you might care more about either Precision or Recall.

Use Precision when False Positives are costly

- Example: **Spam Detection**
  - You prefer not to mistakenly mark an important email as spam.
- Example: **Medical Diagnosis for a Non-Deadly Disease**
  - You don’t want to falsely tell a healthy person they are sick.

Use Recall when False Negatives are costly

- Example: **Cancer Diagnosis**
  - Missing a real cancer case (FN) is more dangerous than mistakenly diagnosing it (FP).
- Example: **Fraud Detection**
  - You don’t want fraudulent transactions to go undetected.

Use F1-Score when both FP & FN matter

- Example: **Search Engine Ranking**
  - You want to return relevant results (high Precision) while ensuring you don’t miss good results (high Recall).
- Example: **Autonomous Vehicles**
  - You need to both correctly identify obstacles (Recall) and avoid false alarms that stop the car unnecessarily (Precision).

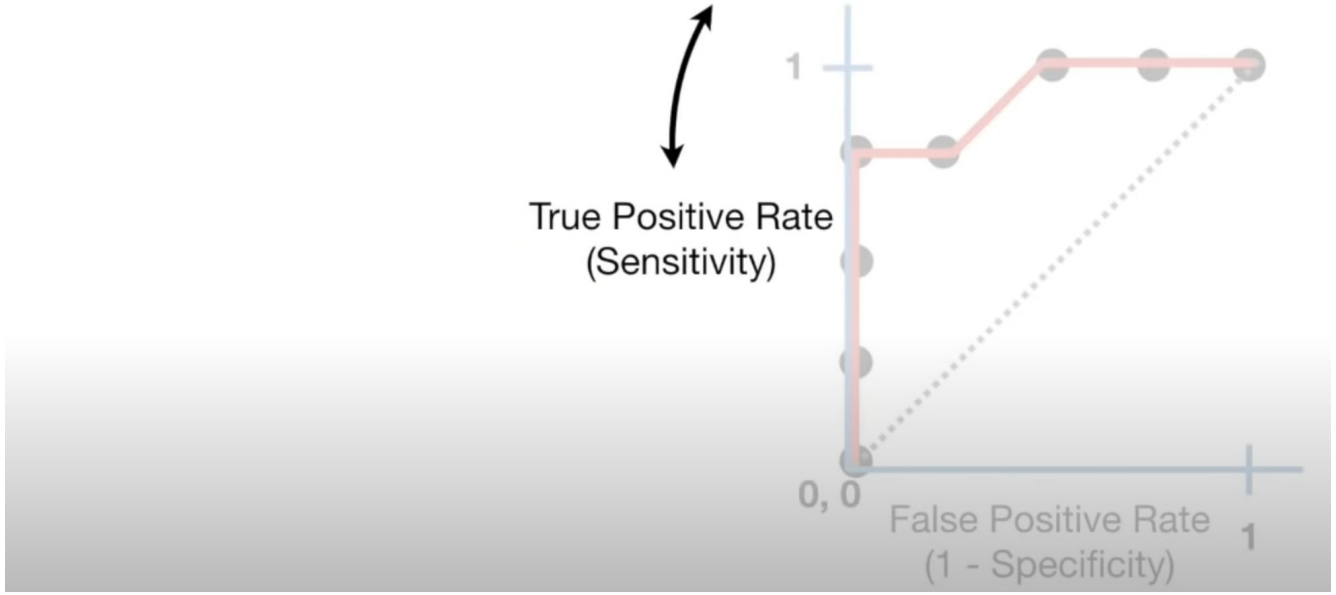
Advanced Classification Metrics

1. ROC curve

ROC (Receiver Operating Characteristic) Curve is a graphical representation that evaluates a classification model's performance across different classification thresholds.

y-axis: True Positive Rate, which is the same thing as sensitivity.

$$\text{True Positive Rate} = \text{Sensitivity} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$$



Advanced Classification Metrics

1. ROC curve

ROC (Receiver Operating Characteristic) Curve is a graphical representation that evaluates a classification model's performance across different classification thresholds.

y-axis: True Positive Rate, which is the same thing as sensitivity.

$$\text{True Positive Rate} = \text{Sensitivity} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$$

Predicted Label \ Real Label	Positive	Negative
Positive	True Positive (TP)	False Positive (FP)
Negative	False Negative (FN)	True Negative (TN)

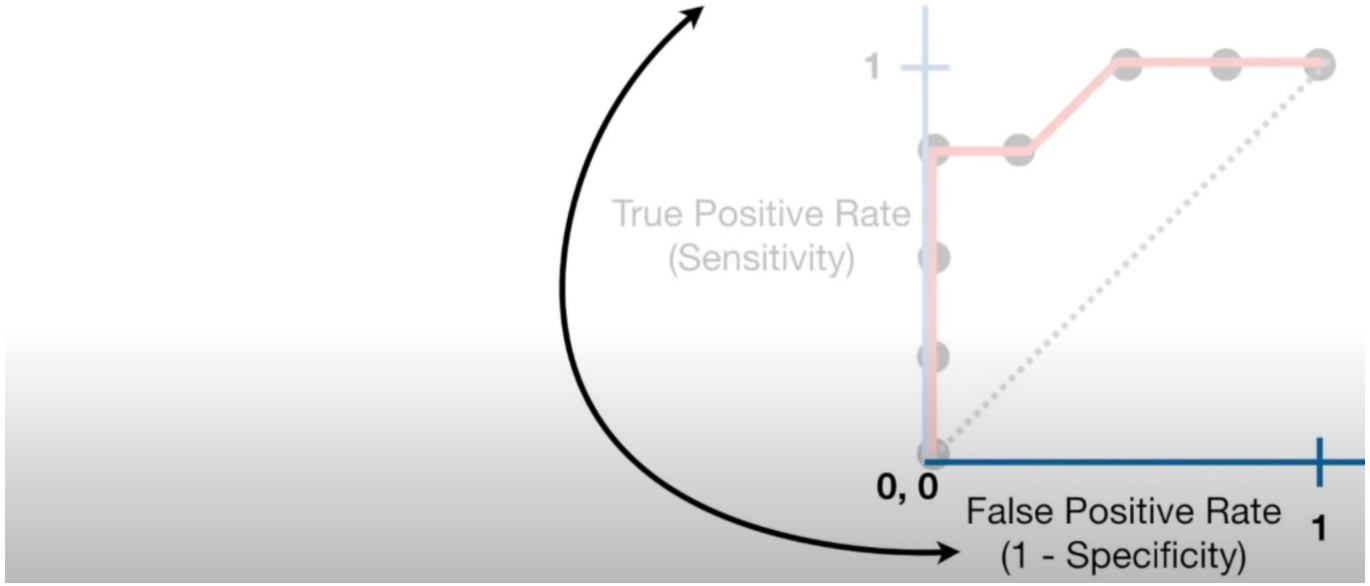
Advanced Classification Metrics

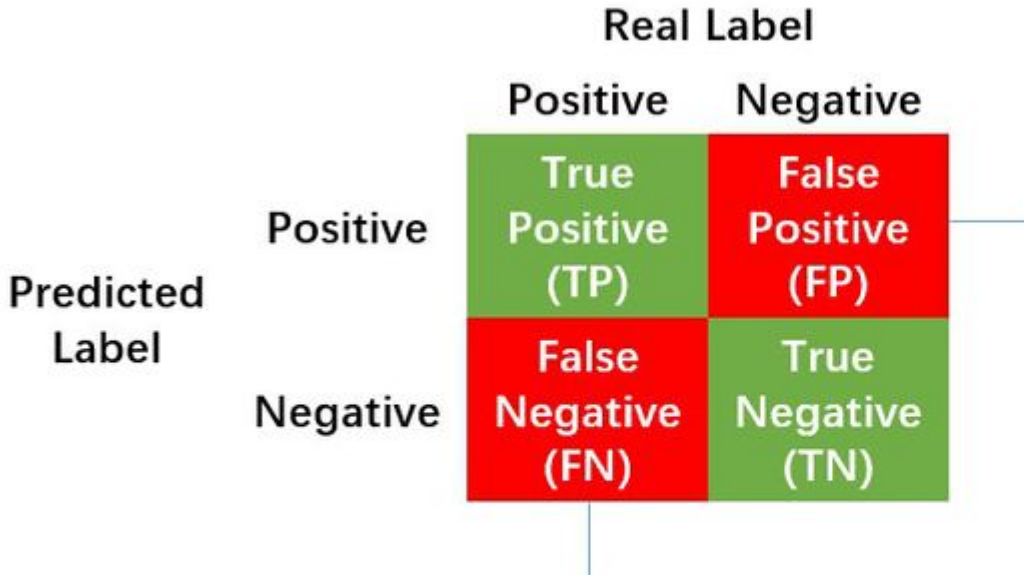
1. ROC curve

ROC (Receiver Operating Characteristic) Curve is a graphical representation that evaluates a classification model's performance across different classification thresholds.

x-axis: False Positive Rate, which is the same thing as sensitivity.

$$\text{False Positive Rate} = (1 - \text{Specificity}) = \frac{\text{False Positives}}{\text{False Positives} + \text{True Negatives}}$$





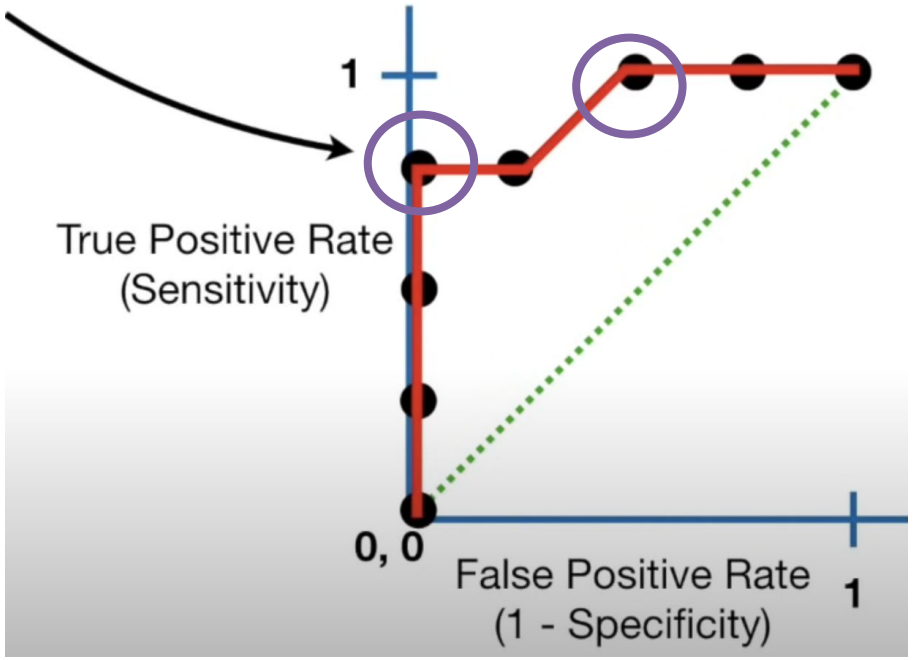
Advanced Classification Metrics

1. ROC curve

ROC (Receiver Operating Characteristic) Curve is a graphical representation that evaluates a classification model's performance across different classification thresholds.

A **classification model** (e.g., logistic regression, neural network) often outputs a **probability score** rather than a direct class label.

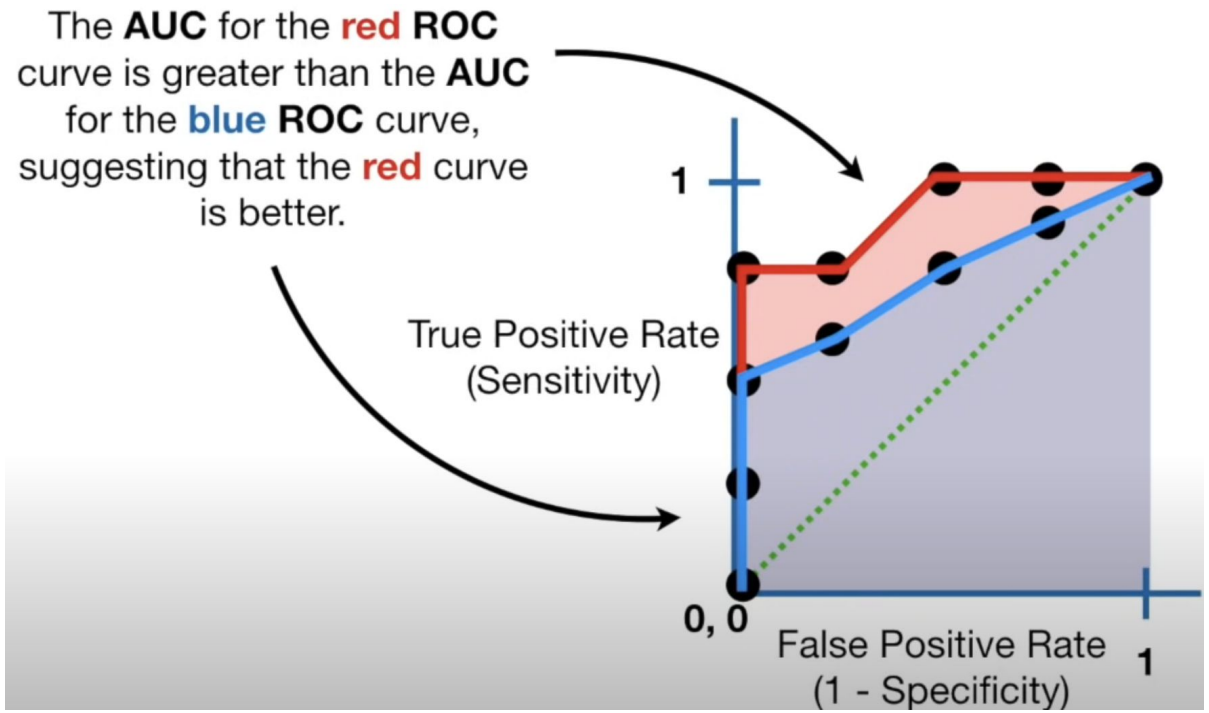
- Default threshold is usually **0.5**, meaning:
  - **Predicted Positive** if **Probability ≥ 0.5**.
  - **Predicted Negative** if **Probability < 0.5**.



Advanced Classification Metrics

1. ROC-AUC curve

AUC is a single scalar value that measures the entire two-dimensional area under the ROC curve, summarizing the model's ability to distinguish between classes.



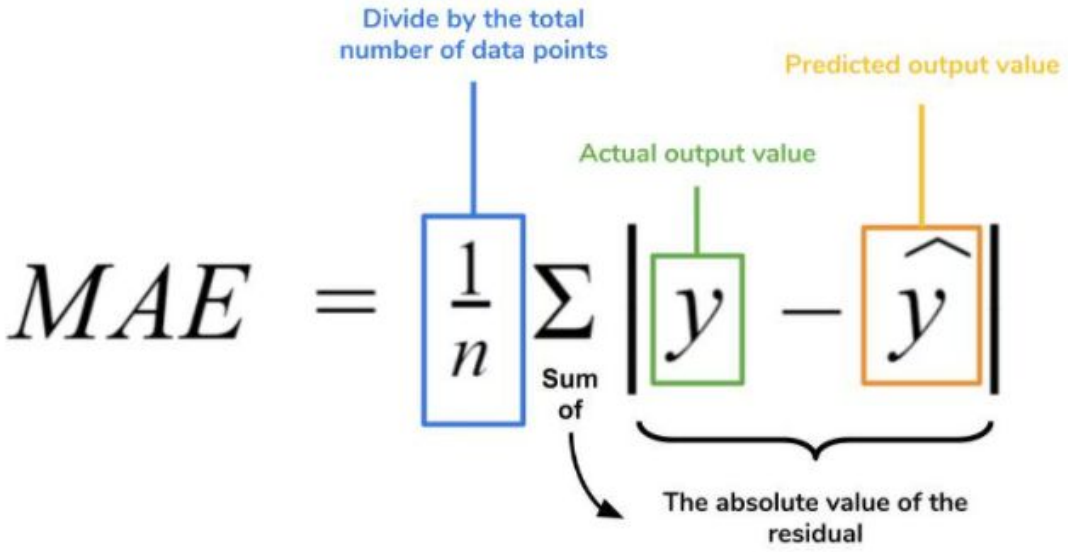
Choosing the Right Classification Metric (Classification)

Scenario	Recommended Metric(s)	Why?
Balanced Dataset	**Accuracy, F1-Score**	When False Positives (FP) and False Negatives (FN) are equally costly.
Imbalanced Dataset	**Precision, Recall, AUC-PR**	To avoid misleading high accuracy.
High Cost for False Positives	**Precision**	Example: Email spam filters, fraudulent transaction detection.
High Cost for False Negatives	**Recall (Sensitivity)**	Example: Cancer diagnosis, critical anomaly detection.
Need a Balance	**F1-Score**	When both FP and FN are important.
Probabilistic Models	**Log Loss**	When evaluating the confidence of predicted probabilities.

Performance Metrics (Regression)

Key Classification Metrics:

1. Mean Absolute Error (MAE)



Strengths:

- **Simple and Intuitive:** Represents the average magnitude of errors.
- **Interpretable in the same units** as the target variable.
- **Less sensitive than MSE** to large errors (but not entirely robust)

Pitfalls:

- **Not Completely Robust to Outliers**
- **Equal Weighting of All Errors:**
  - Small and large errors **contribute equally** to the final score.
  - Might **underestimate the impact** of large deviations.

Performance Metrics (Regression)

Key Regression Metrics:

1. Mean Absolute Error (MAE)

Actual Values	100	150	200	250	300	350	400
Good Predictions	110	145	210	240	310	360	390
Outlier Prediction	110	145	210	240	310	360	700

MAE (Good Predictions): 10.0
MAE (With Outlier): 74.3

Performance Metrics (Regression)

Key Classification Metrics:

2. Mean Square Error (MSE)

$$MSE = \frac{1}{N} \sum_{i=1}^{N} (y_i - \hat{y}_i)^2$$

- $\frac{1}{N}$: average over all results
- $(y_i - \hat{y}_i)^2$: makes result quadratic; y_i = true y, $\hat{y}_i$ = estimate of y

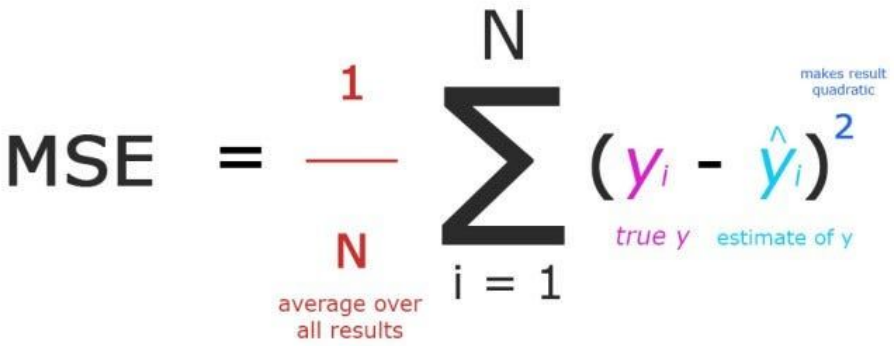
Strengths:

- **Penalizes Larger Errors:** The **squared term** increases the impact of **larger deviations**, making it useful when **large errors are particularly costly**.
- Commonly used in **optimization objectives** for many machine learning algorithms (e.g., **linear regression**, **neural networks**).
- Encourages models to be **more cautious with large deviations**.

Performance Metrics (Regression)

Key Classification Metrics:

2. Mean Square Error (MSE)



Pitfalls:

- **Highly Sensitive to Outliers:**
  - Squaring the error **magnifies the impact of large errors**.
  - Example: An error of **10** contributes **100** to MSE, while an error of **20** contributes **400**.
- **Harder to Interpret:**
  - The **units of MSE** are the **square of the target variable's units**, making it **less intuitive**.

Performance Metrics (Regression)

Key Classification Metrics:

3. Root Mean Squared Error (RMSE)

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N} (Predicted_i - Actual_i)^2}{N}}$$

Strengths:

- **Same Units as the Target Variable:**
  - Unlike MSE, RMSE provides an error metric in the **original scale**, making it **easier to interpret**.
- **Balances Sensitivity to Outliers:**
  - Still **penalizes large errors** (due to squaring), but the **square root softens** the impact compared to MSE.
- **Widely Used in Regression Problems:**
  - Especially when the model's goal is to **minimize prediction errors**.

Performance Metrics (Regression)

Key Classification Metrics:

3. Root Mean Squared Error (RMSE)

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N} (Predicted_i - Actual_i)^2}{N}}$$

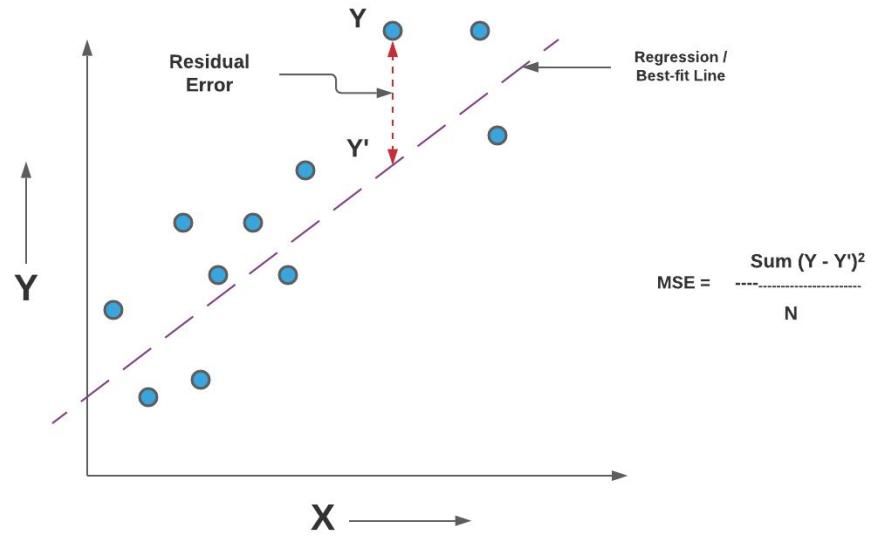
Pitfalls:

- **Still Sensitive to Outliers:**
  - Although not as extreme as MSE, RMSE can still be **significantly impacted by large errors**.
- **Not Ideal for Skewed Data:**
  - When data has many **outliers**, RMSE might not provide a **reliable error metric**.
- **Can Be Misleading with Mixed Scales:**
  - Not suitable when target variable has a **wide range of values**, as it can **overemphasize high-value errors**.

Performance Metrics (Regression)

Key Classification Metrics:

4. R^2 (Coefficient of Determination)



$$R^2 = 1 - \frac{SS_{RES}}{SS_{TOT}} = 1 - \frac{\sum_i (y_i - \hat{y}_i)^2}{\sum_i (y_i - \overline{y})^2}$$

Interpretable: Shows the **proportion of variance explained** by the model.

- **$R^2 = 1$:** Perfect fit (model explains **all** variance).
- **$R^2 = 0$:** Model performs no better than predicting the **mean** of the data.
- **$R^2 < 0$:** Model is **worse than the mean prediction** (e.g., when predictions are very poor).

Versatile: Can be used with **any regression model**, not just **linear regression**.

Good for Model Comparison: When comparing models with the **same data**, a **higher R^2** indicates a **better fit**.

Performance Metrics (Regression)

Key Classification Metrics:

3. R^2 (Coefficient of Determination)

Misleading with Non-Linear Data:

- A high R^2 does not always mean a **good model**.
- With **non-linear relationships**, a high R^2 might be due to **overfitting**.

Not Suitable for Evaluating Predictive Performance:

- R^2 only measures the **fit on the training data**, not the **generalization to unseen data**.
- In some cases, **Adjusted R^2** is better, especially when **adding more predictors** to avoid misleading improvements in R^2.

Sensitive to Outliers:

- Like many metrics, outliers can **skew R^2**, showing an **artificially high or low fit**.

Performance Metrics (Regression)

Key Classification Metrics:

4. Mean Absolute Percentage Error (MAPE)

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\frac{|A_i - F_i|}{A_i}$$

A_i is the actual value

F_i is the forecast value

n is total number of observations

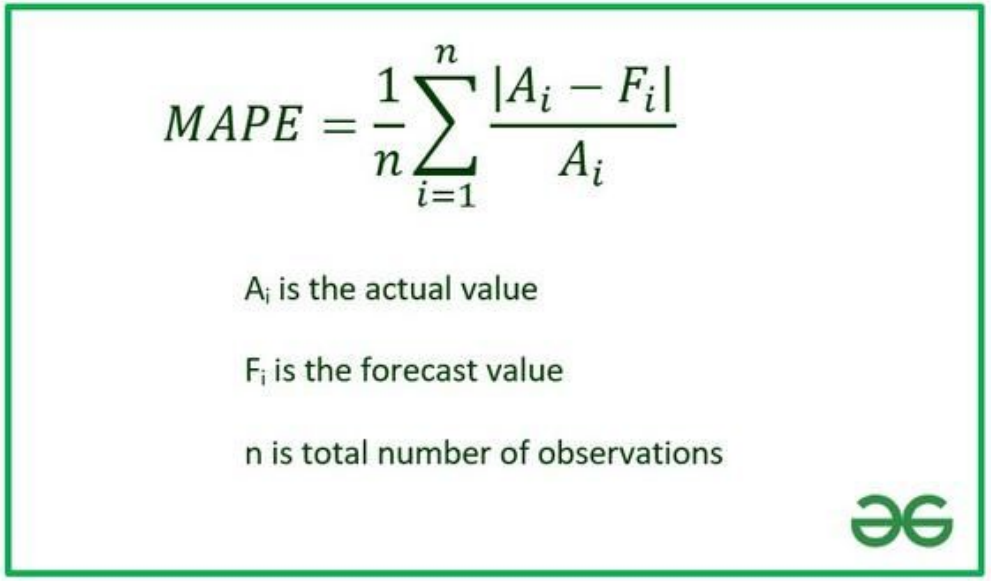
Strengths:

- **Easy to Interpret:** Shows the **average prediction error as a percentage**, making it useful for stakeholders.
- **Scale-Invariant:** Works well with **different ranges of data**, allowing for **easy comparison** between models or datasets.
- **Good for Business Use Cases:** Helpful when you need to explain model accuracy in **real-world terms**, e.g., "On average, predictions are off by **10%**."

Performance Metrics (Regression)

Key Classification Metrics:

4. Mean Absolute Percentage Error (MAPE)



Undefined for Zero Values:

- When $y_i = 0$, MAPE becomes **undefined**, which is problematic for datasets with **zero values**.

Biased Towards Small Values:

- Overestimates error when the **actual values are small**.
- Example: Predicting **5 instead of 1** gives a **400% error**, which might **overemphasize** minor discrepancies.

Not Symmetric:

- Over-predictions and under-predictions of the **same magnitude** produce **different MAPE values**.

Choosing the Right Regression Metric for Different Scenarios

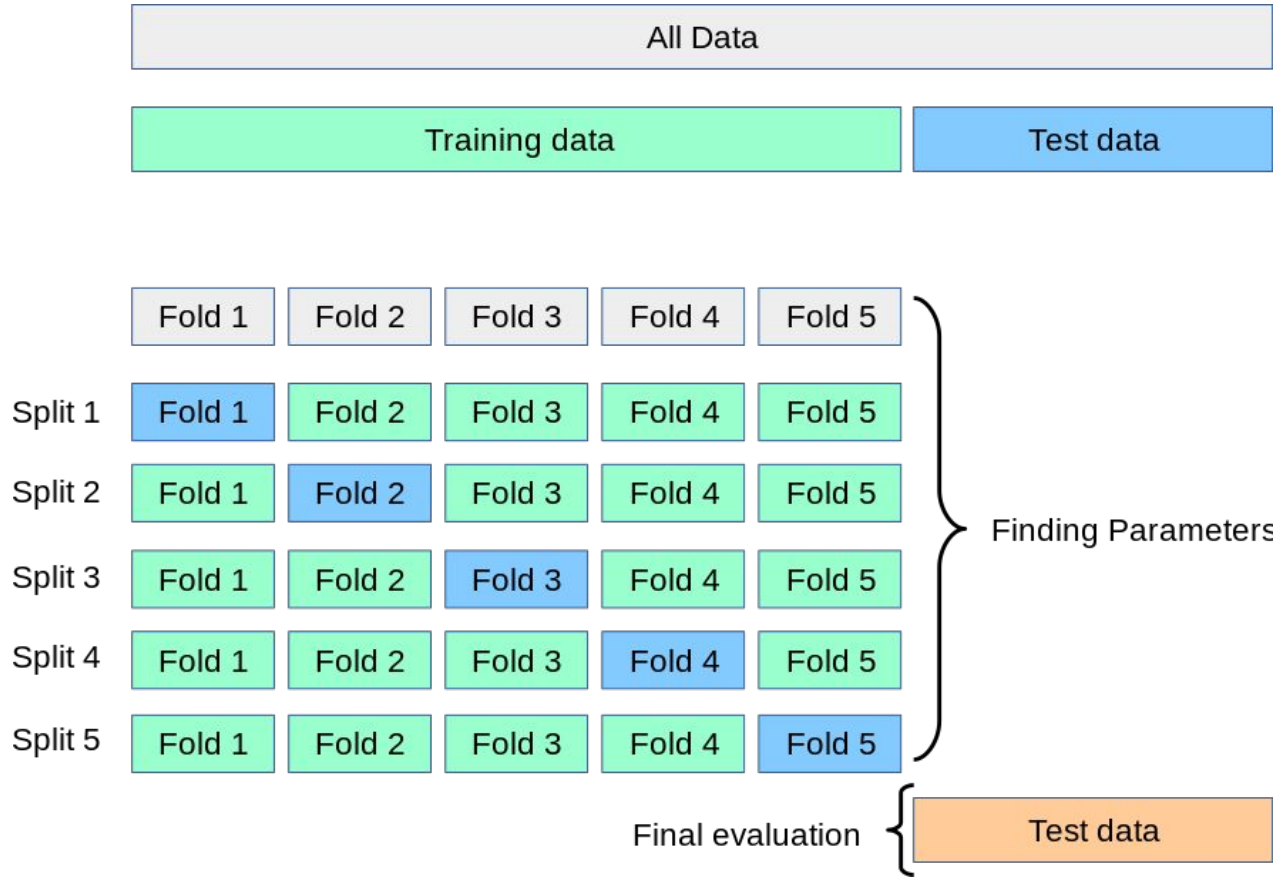
Scenario	Recommended Metric(s)	Why?
When Outliers are Important	MAE (Mean Absolute Error)	Less sensitive to outliers, provides a **balanced view of average error**.
When Outliers are Not Critical	MSE (Mean Squared Error), RMSE (Root Mean Squared Error)	Penalizes **large errors more heavily**, useful for **safety-critical systems**.
Need Interpretation in Same Units	RMSE	Shows the **error in the same units** as the target variable, balances **error sensitivity**.
Understanding Variance Explained	R^2 (Coefficient of Determination)	Shows how much **variance** in the data is **captured by the model**, good for **model fit assessment**.
When Target Scale Varies Widely	MAPE (Mean Absolute Percentage Error)	Provides **error as a percentage**, allowing for **easy comparison** across different scales.
When Data Contains Zero or Small Values	MAE, RMSE, SMAPE (Symmetric MAPE)	Avoids **undefined values** or **inflated errors** that can occur with **MAPE**.
When Comparing Models with Different Features	Adjusted R^2	Adjusts for the **number of predictors**, helping avoid **overestimation of model performance**.

Choosing the Right Regression Metric for Different Scenarios

Forecasting and Time Series	MAPE, MAE, RMSE	Provides **scale-independent error metrics**, helps evaluate **forecast accuracy**.
Evaluating Model Generalization	R^2 on Test Data, Cross-Validation	Shows whether the model is **overfitting** by comparing performance on **training and unseen data**.

Cross Validation

Cross-validation is a technique for evaluating the **generalization performance** of a machine learning model by testing it on **multiple subsets** of the data. It helps ensure that the model is not just learning the **training data** but can also perform well on **unseen data**.



Cross Validation - K folds

Split Data into 'K' Folds:

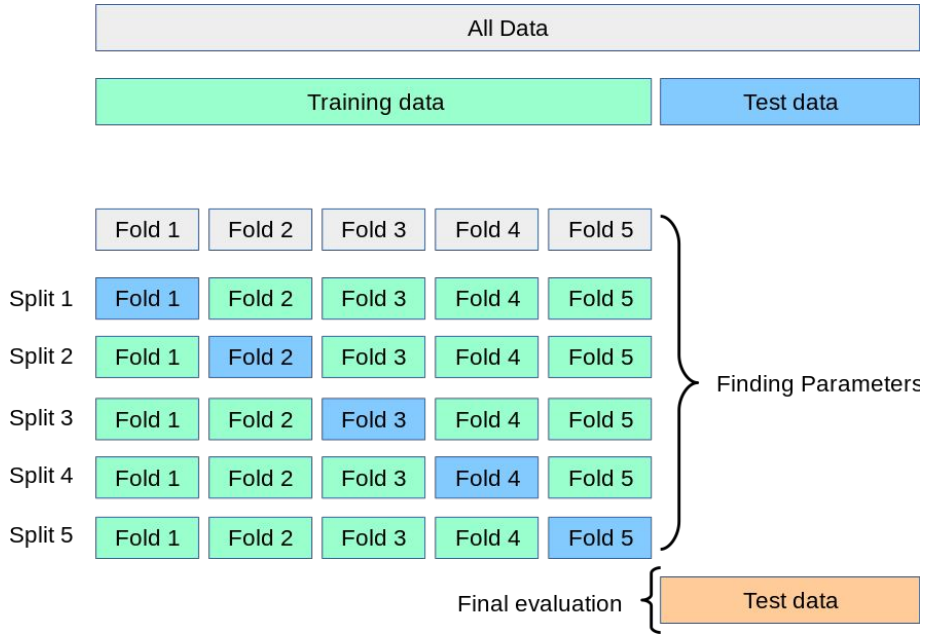
- The dataset is divided into **K equal-sized subsets** (**folds**).
- **Common choices:** 5-fold, 10-fold cross-validation.

Iterative Training and Validation:

- For each iteration, **K-1 folds** are used for **training**, and **1 fold** is used for **validation**.
- The process repeats **K times**, with a **different fold** used for validation each time.

Average the Results:

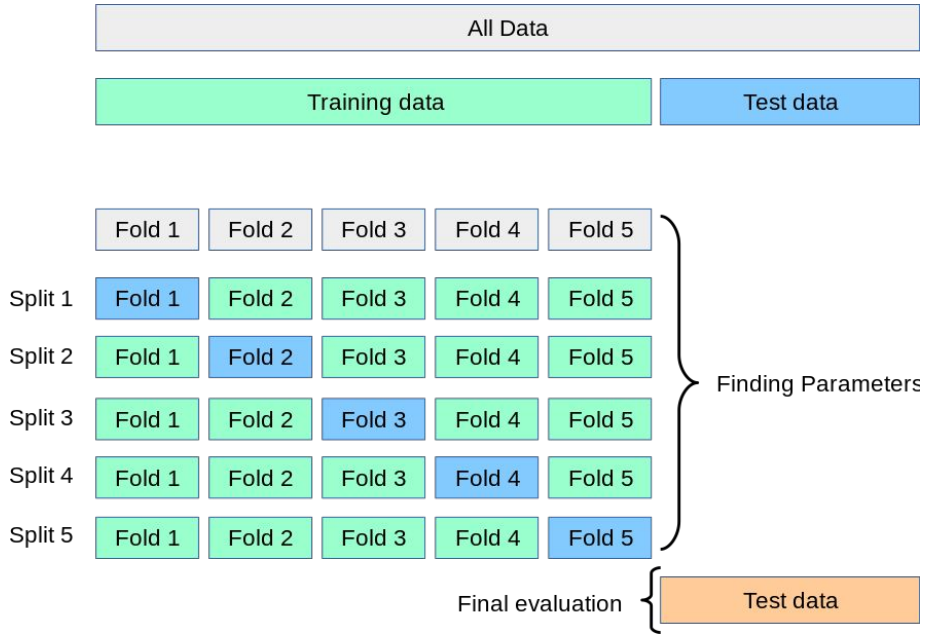
- Performance metrics (e.g., **accuracy**, **MAE**, **R^2**) are calculated for each iteration.
- The final model performance is the **average of all K results**, providing a **more reliable estimate**.



Cross Validation - K folds

Strengths:

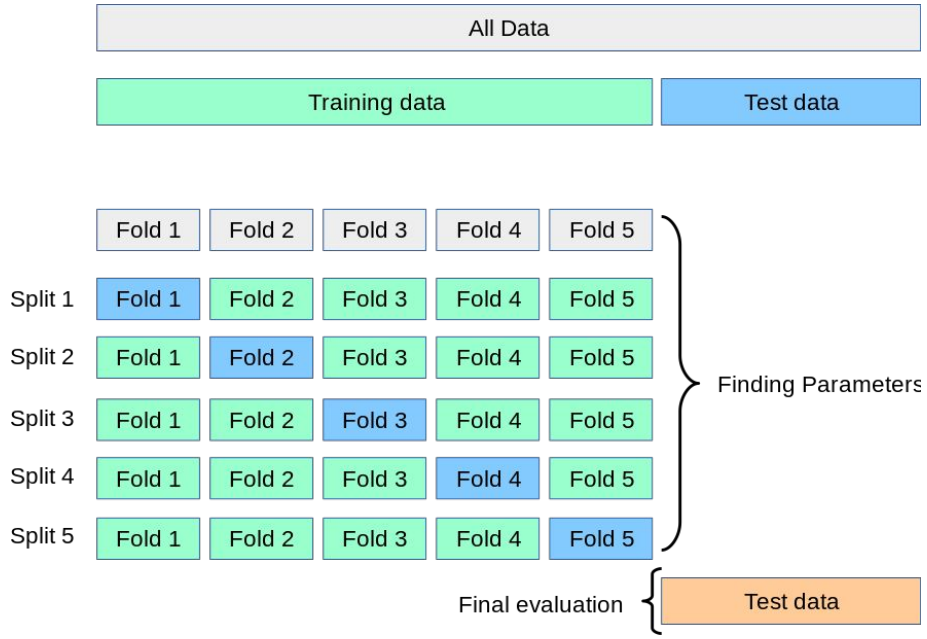
- **Reduces Overfitting:** The model is tested on **different subsets**, providing a **robust performance estimate**.
- **More Data Utilization:** Each data point is used for both **training** and **validation**, maximizing the dataset's **value**.
- **Applicable to Small Datasets:** When the dataset is **limited**, cross-validation helps get a **better model performance estimate** without a large **test set**.



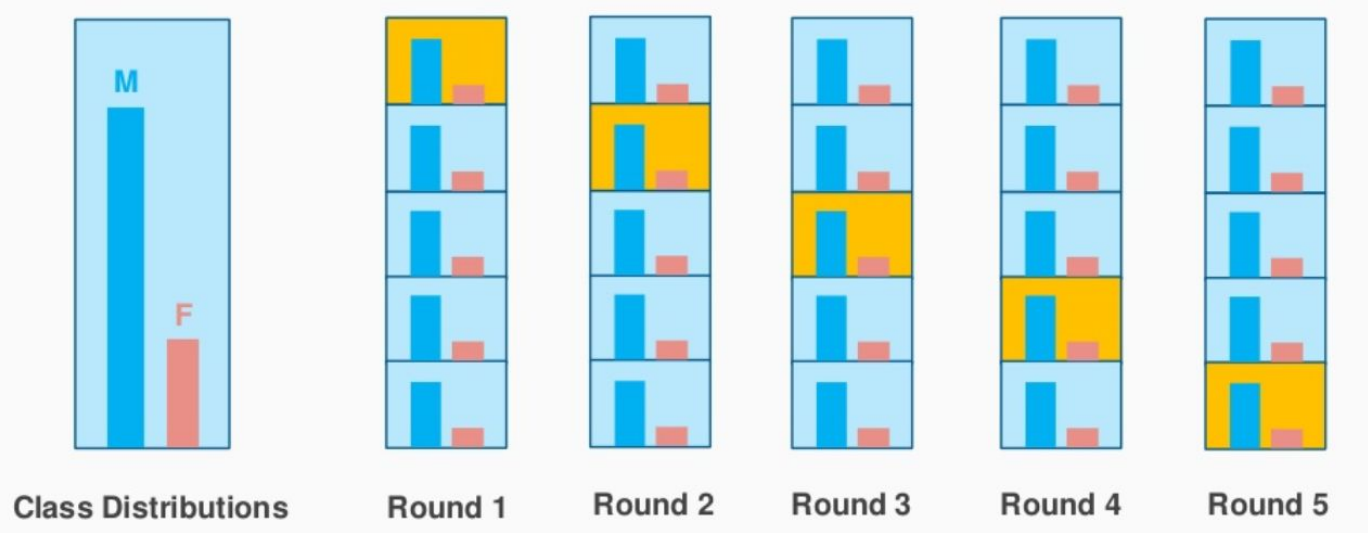
Cross Validation - K folds

Time-Consuming: For large datasets, the process can be **computationally expensive**, especially with **complex models**.

Not Ideal for Time Series Data: When **temporal order matters**, traditional cross-validation can **violate the sequence**. Instead, use **Time Series Split** methods.



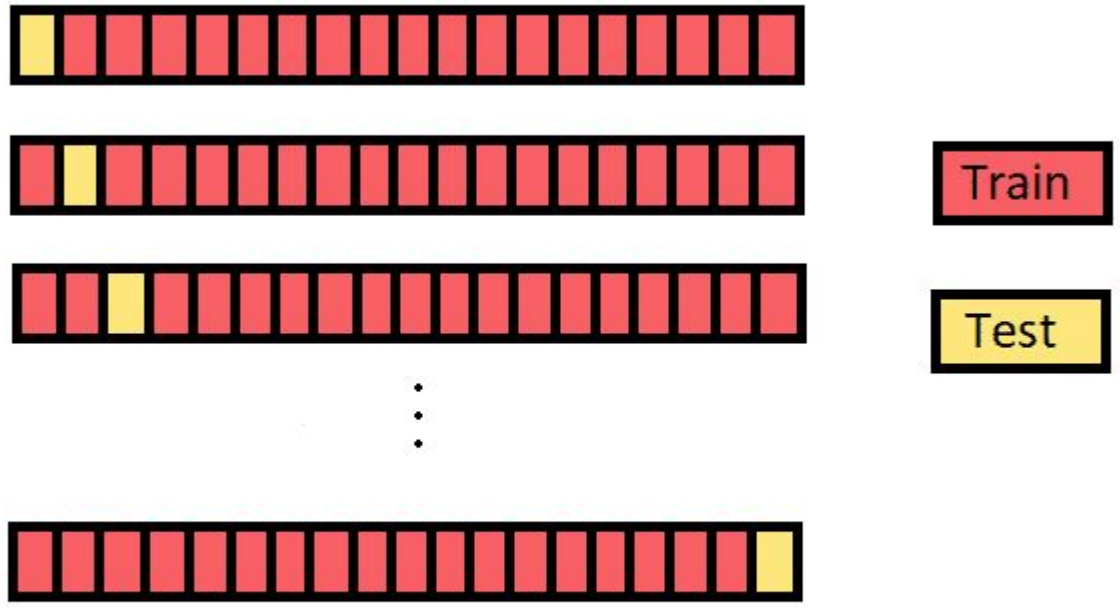
Cross Validation - Stratified K folds



Stratified K-Fold is a variation of **K-Fold Cross-Validation** that ensures each fold has approximately the **same distribution of class labels** as the entire dataset. It is particularly useful for **classification problems** with **imbalanced classes**.

Leave-One-Out Cross-validation (LOOCV)

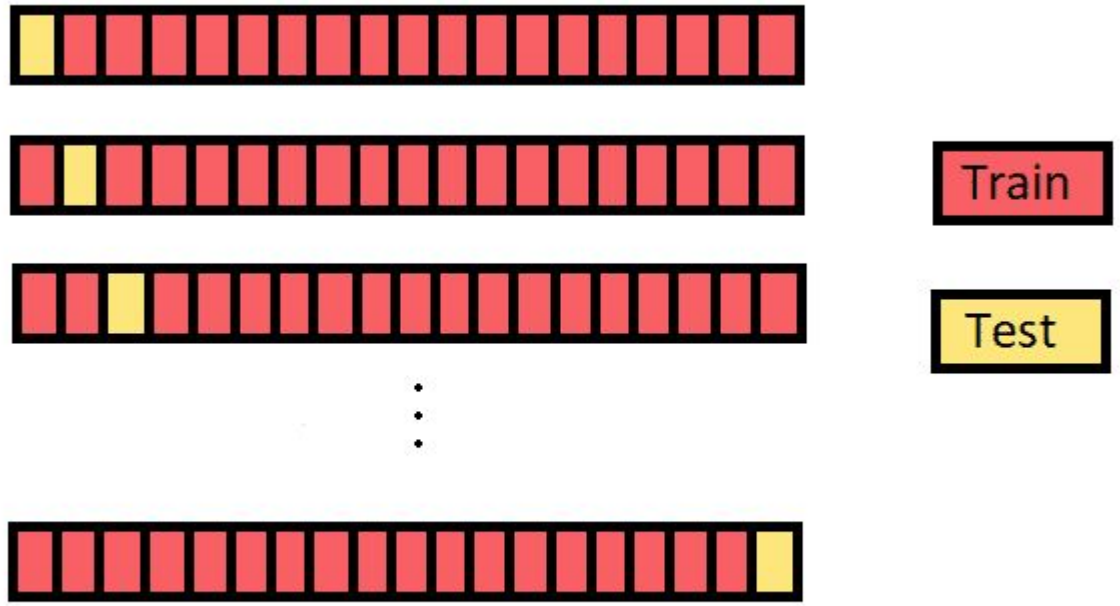
Leave-One-Out Cross-Validation (**LOOCV**) is an extreme form of **K-Fold Cross-Validation** where the number of **folds (K)** is equal to the **number of samples (N)** in the dataset. This means that during each iteration, **one data point is used for validation**, while the **remaining (N-1) points** are used for **training**.



Leave-One-Out Cross-validation (LOOCV)

Strengths:

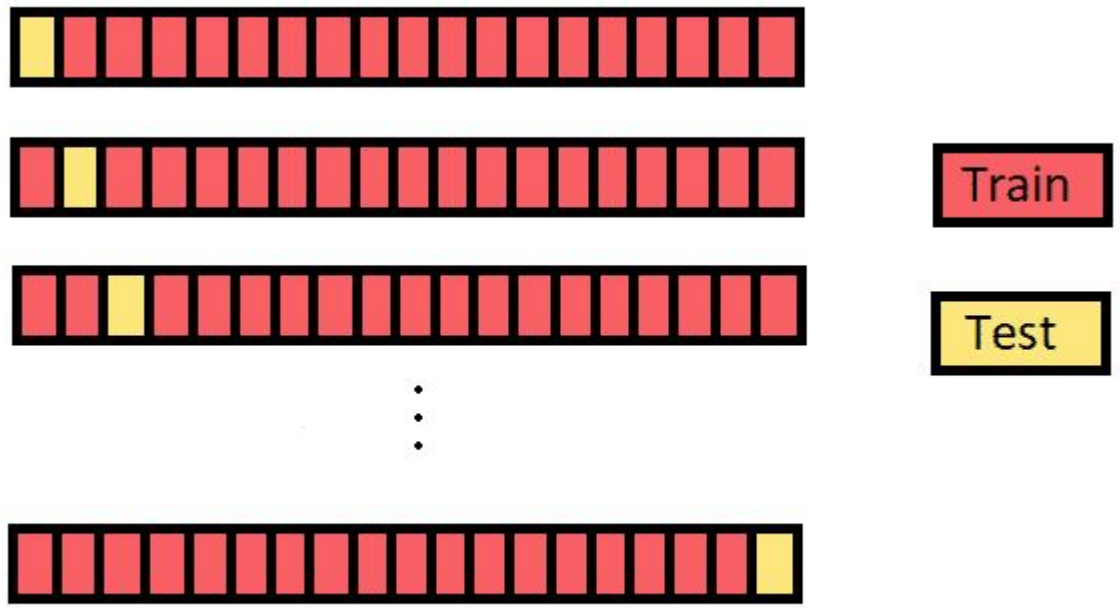
- **Maximum Data Utilization:** The model is trained on **N-1 samples** in every iteration, meaning it **learns from almost the entire dataset**.
- **No Data Wastage:** Every **data point** is used for **validation** exactly **once**, ensuring a **comprehensive evaluation**.
- **Less Bias in Performance Estimate:** Since each data point is **tested independently**, the performance estimate is often **unbiased**.



Leave-One-Out Cross-validation (LOOCV)

Pitfalls:

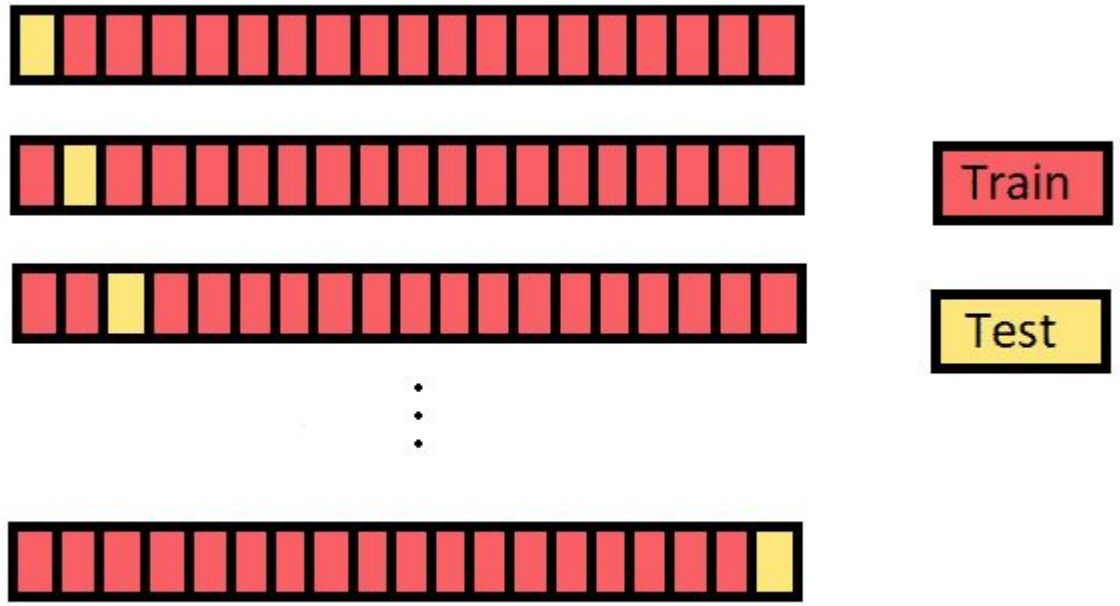
- **Computationally Expensive:** For **large datasets**, the model needs to be **trained N times**, which can be **time-consuming**.
- **Model Variance:** The **validation set** contains only **one sample**, which can lead to **high variance** in the model's performance if the data is **noisy**.
- **Risk of Overfitting:** Since the model is trained on **almost all the data** in each iteration, it may not **generalize well** to **new data**.



Leave-One-Out Cross-validation (LOOCV)

When to Use LOOCV:

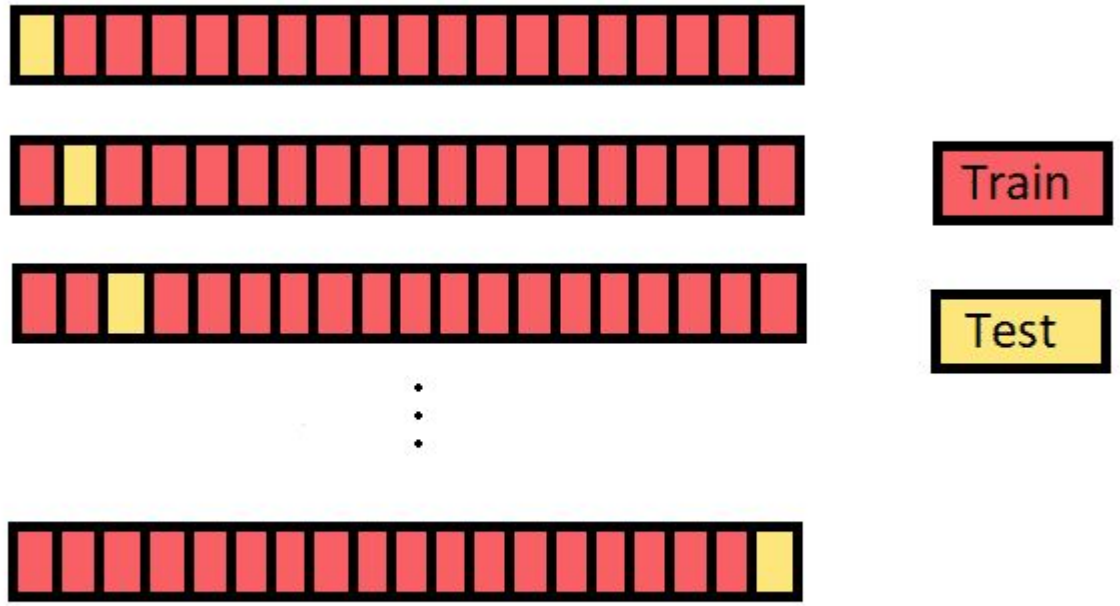
- When the **dataset is very small**, and you want to **maximize training data**.
- When you need a **low-bias performance estimate**, but are **not concerned about computational cost**.
- When evaluating **simple models** where the **training process is quick**, such as **linear regression**.



Leave-One-Out Cross-validation (LOOCV)

When to Use LOOCV:

- When the **dataset is very small**, and you want to **maximize training data**.
- When you need a **low-bias performance estimate**, but are **not concerned about computational cost**.
- When evaluating **simple models** where the **training process is quick**, such as **linear regression**.



Cross-Validation Methods: When to Use Each Type

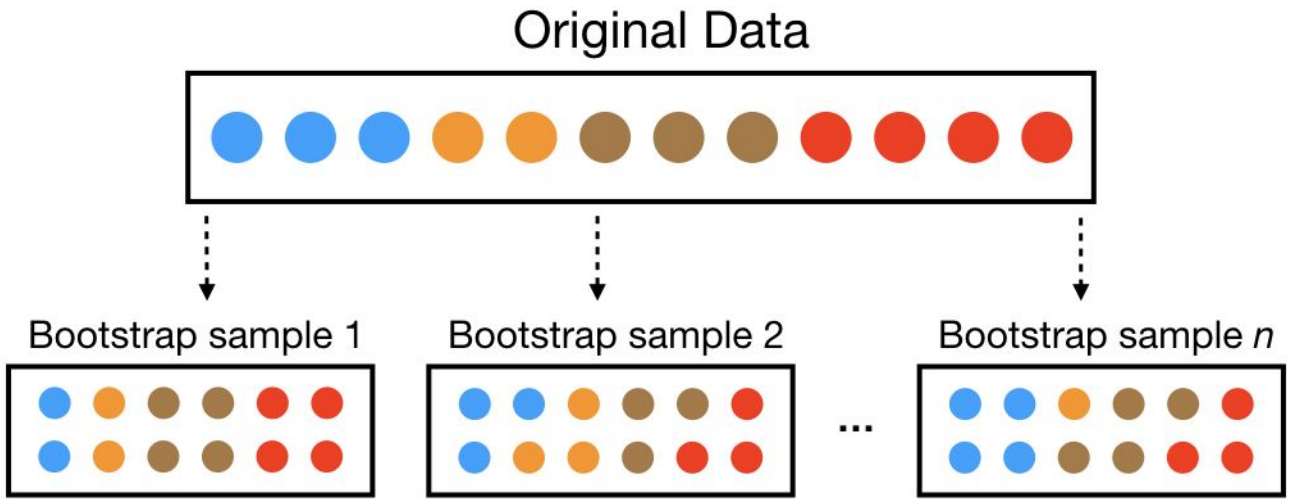
Cross-Validation Method	Description	Best Scenario	Strengths	Pitfalls
K-Fold Cross-Validation	Splits data into **K equal-sized folds**. Each fold is used as validation once, and **K-1** for training.	**General use**, works well with **balanced datasets**.	Reduces **variance**, good **balance of bias-variance**.	**Risk of overfitting** if **K** is too large.
Stratified K-Fold	Like **K-Fold**, but maintains the **class distribution** in each fold.	**Classification problems** with **imbalanced classes**.	Ensures **minority classes** are well represented.	Limited to **classification** tasks, not for **regression**.
Leave-One-Out (LOOCV)	Uses **N folds**, where **N** is the number of samples. Trains on **N-1** samples, validates on **1**.	When the **dataset is very small** or when **maximum data utilization** is needed.	**Unbiased estimates**, **maximum data use**.	**Computationally expensive**, **high variance** in results.

Bootstrap

What is Bootstrap?

Bootstrap is a **resampling method** that involves **randomly sampling** from the **original dataset with replacement** to create **multiple "bootstrap samples"**.

- Each **bootstrap sample** is the **same size** as the **original dataset**, but some **samples may appear multiple times**, while others may be **omitted**.
- Typically, **100 to 1000 bootstrap samples** are generated, and the model is **trained and evaluated** on each sample.



Bootstrap vs cross validation

Aspect	Bootstrap	Cross-Validation
Resampling Method	**With replacement**, can duplicate samples.	**Without replacement**, each sample is used once per fold.
Data Usage	Evaluates on **out-of-bag samples**.	Evaluates on **validation folds**.
Computational Cost	Generally **higher**, but offers **uncertainty estimation**.	Generally **lower**, but may not provide **variance estimates**.
Best For	When you need **confidence intervals** for **model performance**.	When you need a **robust estimate** of **generalization error**.
Common Use Cases	**Small datasets**, **confidence interval estimation**.	**General model validation**, **large datasets**.

Hyperparameter Tuning

Hyperparameter tuning is the process of **selecting the best hyperparameters** for a machine learning model to **optimize performance**.

- **Hyperparameters:** Model parameters set **before training**, such as:
  - **Learning Rate** (e.g., 0.01, 0.001)
  - **Number of Trees** in Random Forest (e.g., 100, 200)
  - **Depth of a Neural Network** (e.g., 3 layers, 5 layers)
  - **Regularization Strength** (e.g., L1, L2 penalties)

Why is Hyperparameter Tuning Important?

Improve Model Performance:

- Hyperparameters can significantly impact a model's **accuracy**, **precision**, **recall**, or **MSE**.

Prevent Overfitting and Underfitting:

- Helps to find the **balance** between **model complexity** and **generalization**.

Enhance Model Robustness:

- Reduces the risk of the model being **too sensitive** to specific **training data**.

Grid Search

What is Grid Search?

Grid Search is an **exhaustive search method** for hyperparameter tuning that evaluates all possible **combinations** of a **predefined grid of hyperparameter values** to find the **best model configuration**.

```python
from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import RandomForestClassifier

# Define model
model = RandomForestClassifier()

# Define hyperparameter grid
param_grid = {
    'n_estimators': [50, 100, 200],
    'max_depth': [10, 20, None],
    'min_samples_split': [2, 5, 10]
}

# Initialize GridSearchCV
grid_search = GridSearchCV(estimator=model, param_grid=param_grid, cv=5,
                           scoring='accuracy')

# Fit the model
grid_search.fit(X_train, y_train)

# Display the best parameters and score
print(f"Best parameters: {grid_search.best_params_}")
print(f"Best cross-validation score: {grid_search.best_score_:.4f}")
```

Random Search

Random Search is a **hyperparameter tuning method** that selects **random combinations** of hyperparameters from a predefined **search space**. Unlike **Grid Search**, which evaluates **all possible combinations**, Random Search explores the space **randomly and efficiently**

```python
from sklearn.model_selection import RandomizedSearchCV
from sklearn.ensemble import RandomForestClassifier

# Define model
model = RandomForestClassifier()

# Define parameter distribution
param_dist = {
    'n_estimators': [50, 100, 200],
    'max_depth': [10, 20, None],
    'min_samples_split': [2, 5, 10]
}

# Initialize RandomizedSearchCV
random_search = RandomizedSearchCV(estimator=model, param_distributions=param_dist,
                                   n_iter=10, cv=5, scoring='accuracy', random_state=42)

# Fit the model
random_search.fit(X_train, y_train)

# Display the best parameters and score
print(f"Best parameters: {random_search.best_params_}")
print(f"Best cross-validation score: {random_search.best_score_:.4f}")
```

Random Search vs Grid Search

Feature	Grid Search	Random Search
Search Strategy	Evaluates **all possible combinations**.	Evaluates a **random subset** of combinations.
Efficiency	**Inefficient** with **large parameter grids**.	**More efficient** with **complex models**.
When to Use	When the **parameter space is small**.	When the **search space is large**.
Computational Cost	**High** with **many parameters**.	Allows **flexibility** in **number of iterations**.