



Model Evaluation and Selection

Performance Metrics: Accuracy, Precision, Recall, F1-Score, ROC Curves, AUC

Sarwan Ali

Department of Computer Science
Georgia State University

 Evaluating Machine Learning Models 

Today's Learning Journey

- 1 Introduction to Model Evaluation
- 2 Basic Performance Metrics
- 3 Precision and Recall
- 4 F1-Score: Harmonic Mean
- 5 ROC Curves and AUC
- 6 Choosing the Right Metric
- 7 Multiclass Extensions
- 8 Advanced Topics
- 9 Practical Implementation
- 10 Best Practices and Common Pitfalls
- 11 Summary and Key Takeaways

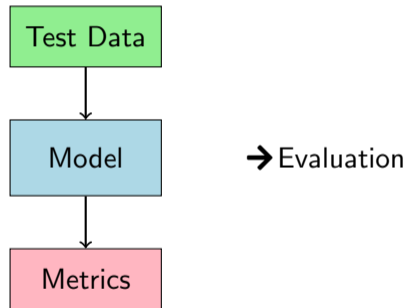
Why Model Evaluation Matters

Key Questions:

- How well does our model perform?
- Which model is better for our problem?
- Will our model generalize to new data?
- What types of errors is our model making?

The Goal:

- Select the best performing model
- Understand model strengths and weaknesses
- Make informed decisions about deployment



Binary Classification Confusion Matrix:

	Predicted	
Actual	Positive	Negative
Positive	TP	FN
Negative	FP	TN

Where:

- TP: True Positives
- TN: True Negatives
- FP: False Positives (Type I Error)
- FN: False Negatives (Type II Error)

Example: Email Spam Detection

	Predicted	
Actual	Spam	Not Spam
Spam	85	15
Not Spam	10	890

- TP = 85 (Correctly identified spam)
- TN = 890 (Correctly identified not spam)
- FP = 10 (Wrongly flagged as spam)
- FN = 15 (Missed spam emails)

Accuracy: The Most Intuitive Metric

Definition:

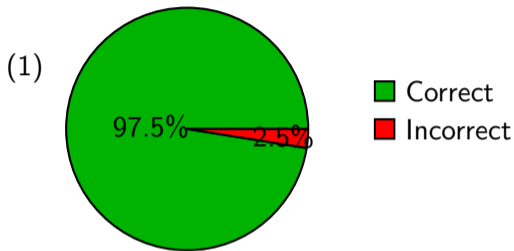
$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

Interpretation:

- Fraction of predictions that are correct
- Range: [0, 1] or [0%, 100%]
- Higher values indicate better performance

Example Calculation:

$$\text{Accuracy} = \frac{85 + 890}{85 + 890 + 10 + 15} = \frac{975}{1000} = 0.975 \quad (2)$$



97.5% Accuracy

(2) **Warning:** Accuracy can be misleading with imbalanced datasets!

The Accuracy Paradox: When High Accuracy Misleads

Scenario: Medical diagnosis for a rare disease (1% prevalence)

Model A (Always predicts "Healthy"):

Actual	Sick	Healthy
Sick	0	10
Healthy	0	990

Accuracy: $\frac{0+990}{1000} = 99\%$

But misses ALL sick patients!

Model B (Balanced):

Actual	Sick	Healthy
Sick	8	2
Healthy	100	890

Accuracy: $\frac{8+890}{1000} = 89.8\%$

Catches 80% of sick patients!

Lesson: Accuracy alone is insufficient for imbalanced datasets!

Precision: Quality of Positive Predictions

Definition:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \quad (3)$$

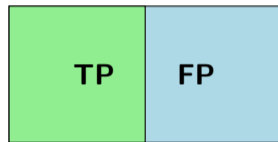
Interpretation:

- "Of all positive predictions, how many were correct?"
- Measures the quality of positive predictions
- High precision = Low false positive rate

When to prioritize precision:

- Spam detection (minimize false positives)
- Medical screening (avoid unnecessary worry)
- Recommendation systems (avoid bad recommendations)

True Positives



All Positive Predictions

Example:

$$\text{Precision} = \frac{85}{85 + 10} = \frac{85}{95} = 0.895 \quad (4)$$

Recall: Coverage of Actual Positives

Definition:

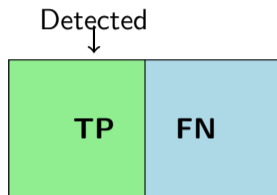
$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \quad (5)$$

Interpretation:

- "Of all actual positives, how many did we catch?"
- Also called **Sensitivity** or **True Positive Rate**
- High recall = Low false negative rate

When to prioritize recall:

- Medical diagnosis (catch all diseases)
- Fraud detection (catch all fraudulent transactions)
- Search engines (find all relevant documents)



Example:

$$\text{Recall} = \frac{85}{85 + 15} = \frac{85}{100} = 0.85 \quad (6)$$

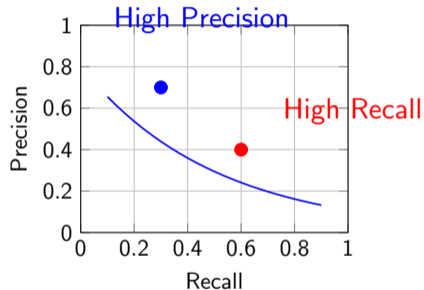
The Precision-Recall Trade-off

The Dilemma:

- Increasing precision often decreases recall
- Increasing recall often decreases precision
- Need to balance based on application needs

Example Scenarios:

- **High Precision Priority:** Email spam filter
- **High Recall Priority:** Cancer screening
- **Balance Both:** Search engines



Question: How do we find the right balance? **Answer:** F1-Score!

F1-Score: Balancing Precision and Recall

Definition:

$$\text{F1-Score} = 2 \cdot \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \quad (7)$$

Why Harmonic Mean?

- Harmonic mean is more sensitive to low values
- Forces both precision and recall to be reasonably high
- Single metric that balances both concerns

Properties:

- Range: $[0, 1]$
- $\text{F1} = 1$ when $\text{Precision} = \text{Recall} = 1$
- F1 approaches 0 when either metric is very low

Example Calculation:

$$\text{Precision} = 0.895 \quad (8)$$

$$\text{Recall} = 0.85 \quad (9)$$

$$\text{F1} = 2 \cdot \frac{0.895 \times 0.85}{0.895 + 0.85} \quad (10)$$

$$= 2 \cdot \frac{0.761}{1.745} \quad (11)$$

$$= 0.872 \quad (12)$$

$$\text{Recall} = 0.85$$

$$\text{Precision} = 0.895$$

$$\text{F1} = 0.872$$

F1-Score vs Arithmetic Mean: Why the Difference Matters

Comparison of Different Scenarios:

Scenario	Precision	Recall	Arithmetic Mean	F1-Score
Balanced	0.8	0.8	0.8	0.8
High Precision	0.9	0.3	0.6	0.45
High Recall	0.3	0.9	0.6	0.45
Very Unbalanced	0.95	0.05	0.5	0.095

Key Observations:

- F1-Score penalizes extreme imbalances
- Arithmetic mean can be misleading
- F1-Score encourages balanced performance

When to Use F1-Score:

- When you need a single metric
- When both precision and recall matter
- For model comparison and selection

ROC Curve: Visualizing Performance Across Thresholds

ROC = Receiver Operating Characteristic

Key Components:

- **True Positive Rate (TPR):** $\frac{TP}{TP+FN}$
(Recall)
- **False Positive Rate (FPR):** $\frac{FP}{FP+TN}$

What ROC Shows:

- Performance at different classification thresholds
- Trade-off between sensitivity and specificity
- Model's ability to discriminate between classes

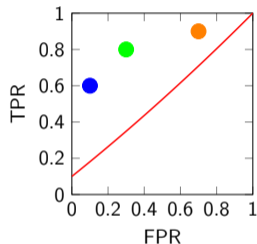


Understanding ROC Curve Interpretation

ROC Curve Interpretation:

- **Perfect Classifier:** Hugs the top-left corner
 - $TPR = 1, FPR = 0$
 - Achieves 100% sensitivity with 0% false positives
- **Good Classifier:** Curves toward top-left
 - Better than random at all thresholds
 - Allows threshold tuning based on requirements
- **Random Classifier:** Diagonal line
 - $TPR = FPR$ at all thresholds
 - No discriminative ability
- **Poor Classifier:** Below diagonal
 - Worse than random guessing
 - Can be inverted to become useful

Threshold Movement:



- **High Threshold:** Conservative
- **Medium Threshold:** Balanced
- **Low Threshold:** Liberal

AUC: Area Under the ROC Curve

AUC Definition:

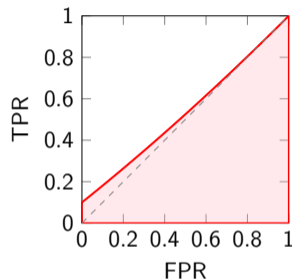
- Area Under the ROC Curve
- Single number summarizing ROC performance
- Range: $[0, 1]$

AUC Interpretation:

- **AUC = 1.0:** Perfect classifier
- **AUC = 0.9-1.0:** Excellent
- **AUC = 0.8-0.9:** Good
- **AUC = 0.7-0.8:** Fair
- **AUC = 0.6-0.7:** Poor
- **AUC = 0.5:** Random classifier
- **AUC \leq 0.5:** Worse than random

Probabilistic Interpretation:

AUC = Probability that model ranks a random positive instance higher than a random negative instance



Advantages of AUC:

- Threshold-independent
- Scale-invariant
- Classification-threshold-invariant
- Good for comparing models

How to Choose the Right Evaluation Metric:

Scenario	Primary Metric	Reasoning
Balanced Dataset	Accuracy	Simple and interpretable when classes are equally represented
Imbalanced Dataset	F1-Score, AUC	Accuracy can be misleading; need metrics that account for class imbalance
Cost of FP \gg FN	Precision	When false positives are very costly (e.g., spam detection)
Cost of FN \gg FP	Recall	When false negatives are very costly (e.g., medical diagnosis)
Need Single Metric	F1-Score	When you need to balance precision and recall
Probability Ranking	AUC	When you care about ranking quality across all thresholds
Multiple Classes	Macro/Micro F1	Extensions of binary metrics to multi-class problems

Real-World Application Examples

Medical Diagnosis:

- **Priority:** High Recall
- **Reasoning:** Missing a disease (FN) is worse than false alarm (FP)
- **Metrics:** Recall, Sensitivity

Email Spam Filter:

- **Priority:** High Precision
- **Reasoning:** Blocking important emails FP is worse than letting spam through FN
- **Metrics:** Precision, Specificity

Credit Card Fraud:

- **Priority:** Balanced F1-Score
- **Reasoning:** Both false alarms and missed fraud are costly
- **Metrics:** F1-Score, AUC

Search Engine:

- **Priority:** Recall with acceptable Precision
- **Reasoning:** Users want comprehensive results
- **Metrics:** Recall@K, MAP

Recommendation System:

- **Priority:** High Precision
- **Reasoning:** Bad recommendations hurt user experience
- **Metrics:** Precision@K, NDCG

Quality Control:

- **Priority:** High Recall
- **Reasoning:** Missing defects can be dangerous
- **Metrics:** Recall, F1-Score

Extending Metrics to Multiclass Problems

Challenge: Binary metrics don't directly apply to multiclass problems

Macro Averaging:

$$\text{Macro-F1} = \frac{1}{C} \sum_{i=1}^C F1_i \quad (13)$$

- Calculate metric for each class
- Average across all classes
- Treats all classes equally
- Good when all classes are important

Example (3 classes):

- Class A: $F1 = 0.8$
- Class B: $F1 = 0.6$
- Class C: $F1 = 0.9$
- $\text{Macro-F1} = (0.8 + 0.6 + 0.9)/3 = 0.77$

Micro Averaging:

$$\text{Micro-F1} = \frac{2 \cdot \sum TP_i}{\sum (2 \cdot TP_i + FP_i + FN_i)} \quad (14)$$

- Aggregate all TP, FP, FN across classes
- Calculate metric on aggregated counts
- Weighted by class frequency
- Good for imbalanced datasets

Weighted Averaging:

$$\text{Weighted-F1} = \sum_{i=1}^C w_i \cdot F1_i \quad (15)$$

Where w_i is the proportion of class i

Multiclass Confusion Matrix Example

3-Class Classification Problem:

	Predicted			
Actual	Cat	Dog	Bird	Total
Cat	80	5	15	100
Dog	10	85	5	100
Bird	20	10	70	100
Total	110	100	90	300

Cat Class:

- Precision: $\frac{80}{110} = 0.727$
- Recall: $\frac{80}{100} = 0.80$
- F1: $\frac{2 \times 0.727 \times 0.80}{0.727 + 0.80} = 0.762$

Dog Class:

- Precision: $\frac{85}{100} = 0.85$
- Recall: $\frac{85}{100} = 0.85$
- F1: $\frac{2 \times 0.85 \times 0.85}{0.85 + 0.85} = 0.85$

Bird Class:

- Precision: $\frac{70}{90} = 0.778$
- Recall: $\frac{70}{100} = 0.70$
- F1: $\frac{2 \times 0.778 \times 0.70}{0.778 + 0.70} = 0.737$

Macro-F1: $\frac{0.762 + 0.85 + 0.737}{3} = 0.783$

Class Imbalance: Strategies and Considerations

Common Issues with Imbalanced Datasets: Problems:

- High accuracy with poor minority class performance
- Models biased toward majority class
- Misleading evaluation metrics

Data-Level Solutions:

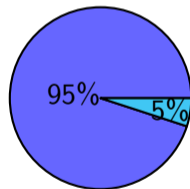
- **Oversampling:** SMOTE, ADASYN
- **Undersampling:** Random, Tomek links
- **Combined:** SMOTETomek

Algorithm-Level Solutions:

- Cost-sensitive learning
- Ensemble methods (BalancedBagging)
- Threshold tuning

Evaluation Strategies:

- Use appropriate metrics (F1, AUC)
- Stratified sampling for train/test split
- Cross-validation with stratification



- Majority Class
- Minority Class

Imbalanced Dataset
5% Minority Class

Cross-Validation for Robust Evaluation

Why Cross-Validation?

- Single train/test split can be misleading
- Better estimate of model performance
- Reduces variance in evaluation
- Uses all data for both training and testing



Fold 3 as Test



Results:

Fold 1: F1 = 0.85
Fold 2: F1 = 0.82
Fold 3: F1 = 0.88
Fold 4: F1 = 0.79
Fold 5: F1 = 0.86

Mean F1: 0.84 ± 0.03

K-Fold Cross-Validation Process:

- 1 Split data into K folds
- 2 Train on K-1 folds, test on 1 fold
- 3 Repeat K times (each fold as test once)
- 4 Average results across all folds

Common Choices:

- $K = 5$ or $K = 10$ (most common)
- Leave-One-Out ($K = n$) for small datasets

Python Implementation Example

Complete Evaluation Pipeline:

```
from sklearn.metrics import (accuracy_score, precision_score,
                             recall_score, f1_score, roc_auc_score,
                             classification_report, confusion_matrix)
from sklearn.model_selection import cross_val_score
import numpy as np
def comprehensive_evaluation(model, X_test, y_test, X_train=None, y_train=None):
    """Comprehensive model evaluation function"""
    y_pred = model.predict(X_test) # Make predictions
    y_proba = model.predict_proba(X_test)[: , 1] # For binary classification
    accuracy = accuracy_score(y_test, y_pred) # Basic metrics
    precision = precision_score(y_test, y_pred)
    recall = recall_score(y_test, y_pred)
    f1 = f1_score(y_test, y_pred)
    auc = roc_auc_score(y_test, y_proba)
    print("==== Model Evaluation Results ====") # Print results
    print(f"Accuracy:--{accuracy:.3f}")
    print(f"Precision:--{precision:.3f}")
    print(f"Recall:----{recall:.3f}")
    print(f"F1-Score:--{f1:.3f}")
    print(f"AUC:-----{auc:.3f}")
    # Cross-validation (if training data provided)
    if X_train is not None and y_train is not None:
        cv_scores = cross_val_score(model, X_train, y_train, cv=5, scoring='f1')
        print(f"CV-F1:-----{cv_scores.mean():.3f}+-{cv_scores.std():.3f}")
    return {'accuracy': accuracy, 'precision': precision, 'recall': recall,
            'f1': f1, 'auc': auc}
```

Visualization Code Example (ROC Curve and Metrics Visualization)

```
import matplotlib.pyplot as plt
from sklearn.metrics import roc_curve, auc
import seaborn as sns
def plot_evaluation_metrics(y_test, y_pred, y_proba):
    """Plot comprehensive evaluation visualizations"""
    fig, ((ax1, ax2), (ax3, ax4)) = plt.subplots(2, 2, figsize=(12, 10))
    cm = confusion_matrix(y_test, y_pred) # 1. Confusion Matrix
    sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', ax=ax1)
    ax1.set_title('Confusion Matrix')
    ax1.set_ylabel('True-Label')
    ax1.set_xlabel('Predicted-Label')
    fpr, tpr, _ = roc_curve(y_test, y_proba) # 2. ROC Curve
    roc_auc = auc(fpr, tpr)
    ax2.plot(fpr, tpr, color='darkorange', lw=2, label=f'ROC-curve-(AUC--{roc_auc:.2f})')
    ax2.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='—')
    ax2.set_xlabel('False-Positive-Rate')
    ax2.set_ylabel('True-Positive-Rate')
    ax2.set_title('ROC-Curve')
    metrics = ['Accuracy', 'Precision', 'Recall', 'F1-Score'] # 3. Metrics Bar Chart
    values = [accuracy_score(y_test, y_pred), precision_score(y_test, y_pred),
              recall_score(y_test, y_pred), f1_score(y_test, y_pred)]
    ax3.bar(metrics, values, color=['blue', 'green', 'red', 'orange'])
    ax3.set_title('Performance-Metrics')
    ax3.set_ylabel('Score')
    ax4.hist(y_proba[y_test==0], alpha=0.5, label='Negative', bins=20) # 4. Prediction Distribution
    ax4.hist(y_proba[y_test==1], alpha=0.5, label='Positive', bins=20)
    ax4.set_xlabel('Predicted-Probability')
    ax4.set_ylabel('Frequency')
    ax4.legend()
    plt.tight_layout()
    plt.show()
```

Best Practices for Model Evaluation

Data Preparation:

- Always use separate test set
- Stratify splits for imbalanced data
- Never use test data for model selection
- Consider temporal splits for time series

Metric Selection:

- Choose metrics aligned with business goals
- Report multiple metrics, not just one
- Use confidence intervals when possible
- Consider comput. cost vs. insight trade-off

Statistical Rigor:

- Use cross-validation for robust estimates
- Report standard deviations
- Test for statistical significance
- Consider multiple random seeds

Interpretation:

- Understand what each metric measures
- Consider class distribution effects
- Analyze confusion matrices
- Look at per-class performance

Documentation:

- Document evaluation methodology
- Record hyperparameter settings
- Track data preprocessing steps
- Version control evaluation scripts

Continuous Monitoring:

- Monitor performance over time
- Check for data drift
- Re-evaluate periodically
- Plan for model updates

Common Pitfalls and How to Avoid Them

Pitfall	Why It's Bad	Solution
Data Leakage	Future information in training data	Careful temporal splits, feature engineering review
Cherry-picking Metrics	Only reporting favorable results	Report comprehensive metrics, pre-define evaluation
Test Set Reuse	Overfitting to test set	Use validation set for model selection, test only once
Ignoring Class Imbalance	Misleading accuracy scores	Use appropriate metrics (F1, AUC), balanced sampling
Single Split Evaluation	Unreliable performance estimates	Use cross-validation, multiple random seeds
Threshold Assumptions	Using default 0.5 threshold	Optimize threshold based on business requirements

Key Takeaways

Essential Concepts:

- **Confusion Matrix:** Foundation for all metrics
- **Accuracy:** Good for balance data, misleading for imbalanced
- **Precision:** Quality of positive predictions
- **Recall:** Coverage of actual positives
- **F1-Score:** Harmonic mean balancing precision/recall
- **ROC/AUC:** Threshold-independent performance

Decision Framework:

- Understand your problem domain
- Consider costs of different error types
- Choose metrics aligned with business goals
- Use multiple complementary metrics

Practical Guidelines:

- Always use proper train/validation/test splits
- Cross-validate for robust estimates
- Be extra careful with imbalanced datasets
- Document your evaluation methodology
- Monitor model performance over time

Remember:

“The best metric is the one that aligns with your business objective”



Questions?

Discussion Topics:

- When have you encountered misleading accuracy?
- How do you handle extreme class imbalance?
- What metrics matter most in your domain?
- Experience with cross-validation strategies?