# Precision-Recall Curves and Evaluation Metrics

Nipun Batra

IIT Gandhinagar

October 14, 2025

# Table of Contents

# Motivation: Real-World Application

# Brick Kiln Detection from Satellite Imagery

**Problem:** Identify illegal brick kilns using satellite imagery

**Key Points: Why This Matters**

- Environmental monitoring and air quality
- Thousands of square kilometers to survey
- Manual inspection is infeasible

# The Challenge: Scale of the Problem

## Dataset Scale

- **Images to scan:** 10,000 satellite images
- **Manual inspection time:** 30 seconds per image
- **Total manual effort:** $10{,}000 \times 30s$
- **That's 83 hours of continuous work!**

Can we automate this with machine learning?

# Why Not Just Use Accuracy?

## Three Models to Choose From

- Model A: 95% accuracy
- Model B: 92% accuracy
- Model C: 89% accuracy

# Why Not Just Use Accuracy?

## Three Models to Choose From

- Model A: 95% accuracy
- Model B: 92% accuracy
- Model C: 89% accuracy

**Key Points: The Problem**

Accuracy doesn't tell us about the **types of errors**!

# Types of Errors Matter

**Example: False Positive (Type I Error)**

Model says "brick kiln detected" but there isn't one

- Wastes inspector's time
- Reduces trust in the system

**Example: False Negative (Type II Error)**

Model misses an actual brick kiln

- Environmental violation goes undetected
- Defeats the purpose of monitoring

# Scenario 1: High Precision Model

**Example: Conservative Classifier**

**Model behavior:** Only flags when very confident

## Results

- Flags 100 images as "has brick kiln"
- Inspector time: $100 \times 30s = 50$ minutes

**Key Points: Trade-offs**

✓ Few false alarms

✓ Inspector time well-spent

× Might miss many kilns

# Scenario 2: High Recall Model

**Example: Aggressive Classifier**

**Model behavior:** Flags anything suspicious

## Results

- Flags 2,000 images as “has brick kiln”
- Inspector time: $2{,}000 \times 30s = 16.7$ hours

**Key Points: Trade-offs**

✓ Catches almost all kilns

× Many false alarms

× Wastes inspector time

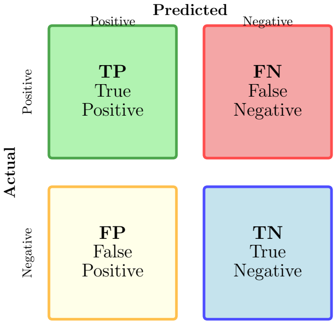
# Classification Metrics Fundamentals

# The Confusion Matrix

**Definition: Confusion Matrix**

| | | Predicted | |
|---|---|---|---|
| | | Pos | Neg |
| Actual | Pos | TP | FN |
| | Neg | FP | TN |

- **TP**: Correct positive
- **FP**: Type I error
- **TN**: Correct negative
- **FN**: Type II error



# Precision: Reliability of Positive Predictions

**Definition: Precision**

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

**Question it answers:**
Of all instances we predicted as positive,
what fraction was actually positive?

# Precision: Example

**Example: Brick Kiln Detection**

- Model flags 100 images as having brick kilns
- 80 actually have brick kilns (TP)
- 20 are false alarms (FP)

$$\text{Precision} = \frac{80}{100} = 0.80 \text{ or } 80\%$$

**Key Points: Interpretation**

When the model says “brick kiln detected,” it’s correct 80% of the time

# Recall: Completeness of Detection

**Definition: Recall (Sensitivity, TPR)**

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

**Question it answers:**
Of all actual positive instances,
what fraction did we correctly identify?

# Recall: Example

### Example: Brick Kiln Detection

- 150 images actually contain brick kilns
- Model correctly identifies 80 (TP)
- Model misses 70 of them (FN)

$$\text{Recall} = \frac{80}{150} = 0.533 \text{ or } 53.3\%$$

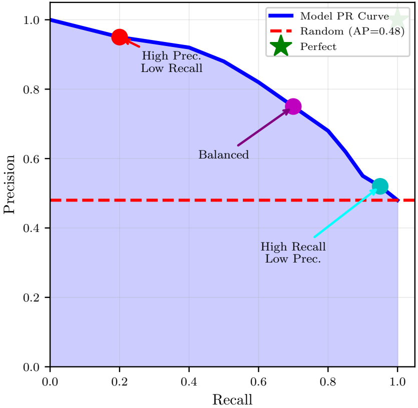
### Key Points: Interpretation

The model finds only about half of all brick kilns

# The Precision-Recall Trade-off

**Key Points: Fundamental Tension**

Improving one metric often hurts the other!



# Trade-off: Model Behavior

## Conservative Model

- High threshold
- Few predictions
- High precision
- Low recall

## Aggressive Model

- Low threshold
- Many predictions
- Low precision
- High recall

# Classification Thresholds

# From Probabilities to Predictions

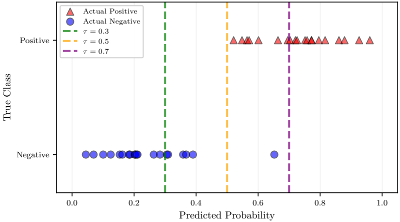
**Definition: How Classifiers Work**

Most classifiers output **probabilities**, not direct predictions

Classification threshold $\tau$ converts probabilities to classes:

$$\hat{y} = \begin{cases} 1 & \text{if } P(y = 1|x) \geq \tau \\ 0 & \text{if } P(y = 1|x) < \tau \end{cases}$$

Default: $\tau = 0.5$



# Threshold Example

**Example: Three Images, Different Thresholds**

| Image | $P(\text{kiln})$ | $\tau = 0.5$ | $\tau = 0.7$ |
|---|---|---|---|
| A | 0.85 | Positive | Positive |
| B | 0.62 | Positive | Negative |
| C | 0.38 | Negative | Negative |

**Key Points: Key Insight**

Same model, different thresholds = different predictions!

# Low Threshold Effects

**Threshold $\tau = 0.3$**

Classify as positive if $P(y = 1|x) \geq 0.3$

- More instances classified as positive
- Higher recall (catch more positives)
- Lower precision (more false positives)
- More false alarms

**Use when:** Missing positives is costly

# High Threshold Effects

**Threshold $\tau = 0.7$**

Classify as positive if $P(y = 1|x) \geq 0.7$

- Fewer instances classified as positive
- Lower recall (miss more positives)
- Higher precision (fewer false positives)
- Fewer false alarms

**Use when:** False alarms are costly

# Precision-Recall Curves

# What is a PR Curve?

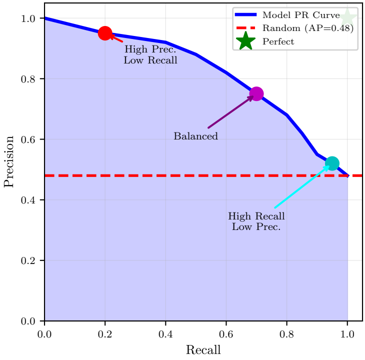
**Definition: Precision-Recall Curve**

A plot showing precision vs. recall for all possible threshold values

- **X-axis:** Recall
- **Y-axis:** Precision
- Each point = one threshold value

**Key Points: What It Shows**

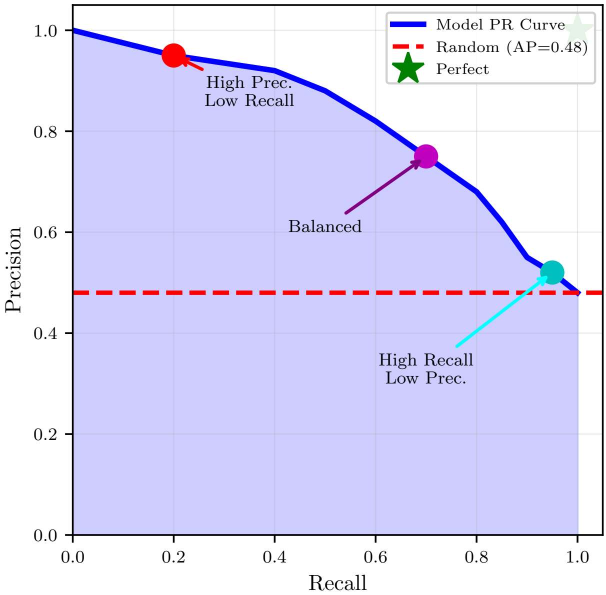
The complete trade-off space between precision and recall



# Building a PR Curve: Steps

1. Train classifier (e.g., Logistic Regression)
2. Get predicted probabilities for test set
3. For each threshold $\tau \in [0, 1]$:
    - Apply threshold to get predictions
    - Compute confusion matrix
    - Calculate precision and recall
    - Plot (recall, precision) point

# Building a PR Curve: Visualization



# Implementation in Scikit-learn

**Python Code**

```
from sklearn.metrics import precision_recall_curve

# Get predicted probabilities
y_scores = model.predict_proba(X_test)[:, 1]

# Compute PR curve
precision, recall, thresholds = \
    precision_recall_curve(y_test, y_scores)
```

# Example: Synthetic Dataset

## Example: Dataset from Notebook

- Created using `make_blobs()`
- 100 samples, 2 features, 2 classes
- Training: 40 samples
- Test: 60 samples
- Cluster standard deviation: 8.0
- Classifier: Logistic Regression

# Threshold Analysis: Low Values

**Example: From Notebook: Threshold = 0.00**

- **Precision:** 0.48
- **Recall:** 1.00

**Interpretation:**

- Classifies almost everything as positive
- Catches all positive cases (perfect recall)
- But only 48% are actually positive

# Threshold Analysis: Medium Values

**Example: From Notebook: Threshold = 0.50**

- **Precision:** 0.74
- **Recall:** 0.69

**Interpretation:**

- Balanced operating point
- Good precision: 74% of predictions correct
- Good recall: finds 69% of positives
- This is the default threshold

# Threshold Analysis: High Values

**Example: From Notebook: Threshold = 0.90**

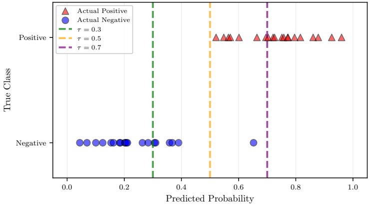
- **Precision:** 1.00
- **Recall:** 0.24

**Interpretation:**

- Very conservative classification
- Perfect precision: all predictions correct!
- But misses 76% of positive cases
- Only confident predictions are made

# Complete Threshold Table

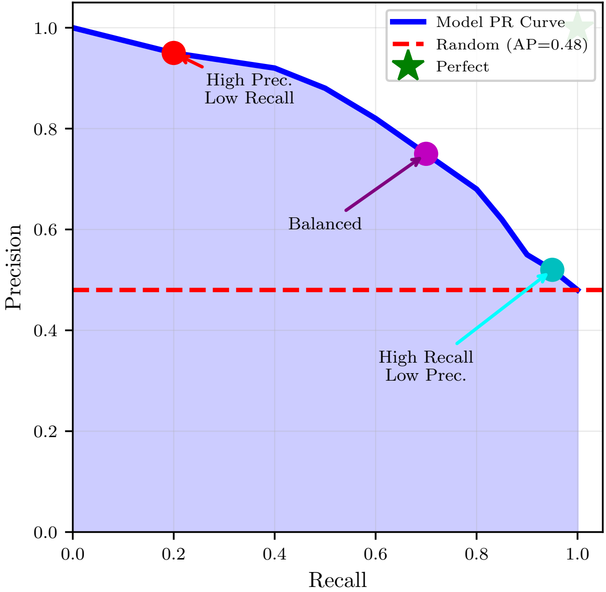
| Threshold | Precision | Recall |
|---|---|---|
| 0.00 | 0.48 | 1.00 |
| 0.10 | 0.55 | 0.98 |
| 0.30 | 0.65 | 0.85 |
| 0.50 | 0.74 | 0.69 |
| 0.70 | 0.85 | 0.45 |
| 0.90 | 1.00 | 0.24 |




**Key Points: Observation**

As threshold increases: Precision $\uparrow$, Recall $\downarrow$

# Interpreting PR Curves



# Interpreting PR Curves

**Key Points: What Makes a Good Curve?**

- Curve closer to top-right is better
- Top-right $=$ high precision AND high recall
- Perfect classifier: stays at $(1, 1)$

# Interpreting PR Curves: Baseline

## Baseline: Random Classifier

Horizontal line at $y = \frac{\text{\# positives}}{\text{total}}$

For balanced classes: $y = 0.5$

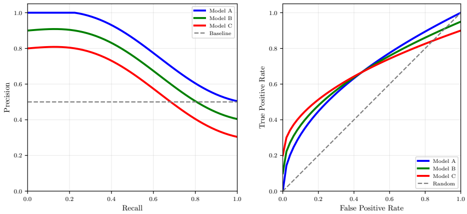
### Example: Example

If 48% of data is positive class:
Random classifier has precision $\approx 0.48$ at all recall levels

# Comparing Models with PR Curves

## Model Comparison Rules

1. If one curve dominates (always above), that model is better
2. If curves cross, choice depends on your needs:
   - Need high precision? Use left side of curve
   - Need high recall? Use right side of curve



# Application-Specific Decisions

# When to Prioritize Precision

**Example: High Precision Scenarios**

False positives are costly:

- **Spam detection**
  Don't want legitimate emails in spam folder
- **Medical diagnosis**
  Before expensive/risky treatment
- **Fraud detection**
  Don't block legitimate transactions

**Strategy:** Choose high threshold

# When to Prioritize Recall

**Example: High Recall Scenarios**

False negatives are costly:

- **Cancer screening**
  Can't afford to miss cases
- **Security threats**
  Missing a threat is catastrophic
- **Environmental compliance**
  Must catch all violations

**Strategy:** Choose low threshold

# Decision Analysis: Option A

## High Precision Choice: $\tau = 0.7$

**Metrics:**

- Precision: 0.85
- Recall: 0.55

### Example: Implications

- Flags 200 images
- 170 true positives, 30 false positives
- Inspection time: 1.7 hours
- Misses 45% of kilns

# Decision Analysis: Option B

## High Recall Choice: $\tau = 0.4$

**Metrics:**

- Precision: 0.65
- Recall: 0.85

### Example: Implications

- Flags 500 images
- 325 true positives, 175 false positives
- Inspection time: 4.2 hours
- Only misses 15% of kilns

# Which Option to Choose?

## Decision Factors

- **Budget:** How much inspector time available?
- **Legal:** Required detection rate?
- **Environmental urgency:** Cost of missed kilns?

**Key Points: Typical Choice**

For environmental compliance:
Option B (high recall) is usually preferred

Missing violations is worse than
spending extra inspection time

# Related Metrics

# F1 Score: Balancing Both Metrics

**Definition: F1 Score**

Harmonic mean of precision and recall:

$$F_1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

Alternative form:

$$F_1 = \frac{2 \cdot \text{TP}}{2 \cdot \text{TP} + \text{FP} + \text{FN}}$$

# Why Harmonic Mean?

**Key Points: Properties of F1**

- Range: $[0, 1]$, higher is better
- Heavily penalizes imbalanced metrics
- Both precision and recall must be good

**Example: Example Comparison**

- $P = 0.80, R = 0.60 \Rightarrow F_1 = 0.686$
- $P = 0.70, R = 0.70 \Rightarrow F_1 = 0.700$

Balanced metrics give better F1!

# $F_\beta$ Score: Weighted Version

**Definition: $F_\beta$ Score**

$$F_\beta = (1 + \beta^2) \cdot \frac{\text{Precision} \cdot \text{Recall}}{\beta^2 \cdot \text{Precision} + \text{Recall}}$$

**Parameter $\beta$:**

- $\beta = 1$: Equal weight ($F_1$ score)
- $\beta < 1$: Favor precision (e.g., $F_{0.5}$)
- $\beta > 1$: Favor recall (e.g., $F_2$)

# $F_\beta$ Applications: High Recall

**Example: $F_2$ Score**

**Use when:** Recall is $2\times$ more important than precision

**Applications:**

- **Cancer screening**
  Missing a cancer case is catastrophic
- **Security threat detection**
  Can't afford to miss threats
- **Environmental compliance**
  Our brick kiln detection example

Higher $\beta$ = More weight on recall

# $F_\beta$ Applications: High Precision

**Example: $F_{0.5}$ Score**

**Use when:** Precision is $2\times$ more important than recall

**Applications:**

- **Search engines**
  Show most relevant results first
- **Spam detection**
  Avoid false positives (legitimate emails in spam)
- **Medical diagnoses**
  Before expensive/invasive treatments

Lower $\beta$ = More weight on precision

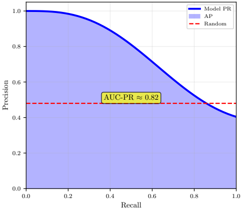
# Average Precision (AP)

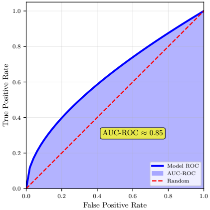
**Definition: Average Precision**

Area under the precision-recall curve:

$$\text{AP} = \sum_{n=1}^{N} (R_n - R_{n-1}) \cdot P_n$$

where $P_n$ and $R_n$ are precision and recall at the $n$-th threshold



# Average Precision: Properties

**Key Points: Key Properties**

- Range: $[0, 1]$, higher is better
- Single number summarizing entire curve
- Perfect classifier: AP $= 1.0$
- Weighted by recall changes

# When to Use Average Precision

**Key Points: Use Cases**

- Comparing models across all thresholds
- When you can't choose single operating point
- Benchmark competitions

**Example: Object Detection**

**mAP** (mean Average Precision):

Average of AP across all object classes

Standard metric in COCO, Pascal VOC

# Specificity (True Negative Rate)

**Definition: Specificity**

$$\text{Specificity} = \frac{\text{TN}}{\text{TN} + \text{FP}}$$

Fraction of negatives correctly identified

**Example: Example**

Out of 100 non-kiln images, if we correctly identify 90:
$\text{Specificity} = 90/100 = 0.90$

# False Positive Rate (FPR)

**Definition: FPR**

$$\text{FPR} = \frac{\text{FP}}{\text{FP} + \text{TN}} = 1 - \text{Specificity}$$

Fraction of negatives wrongly classified

**Key Points: Relationship**

FPR and Specificity are complements:
$\text{FPR} + \text{Specificity} = 1$

# ROC Curves

# What is ROC?

**Definition: ROC: Receiver Operating Characteristic**

Developed during World War II for analyzing radar signals

**Breaking down the name:**

- **R**eceiver: The detector/classifier receiving signals
- **O**perating: Different operating points (thresholds)
- **C**haracteristic: Performance at each threshold

**Key Points: Historical Context**

Originally used to analyze radar operators' ability
to correctly detect enemy aircraft from radar signals

# ROC Curve Definition

**Definition: What ROC Plots**

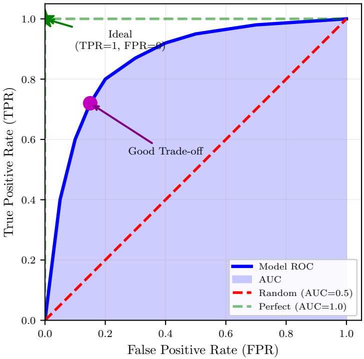
ROC curve plots TPR vs FPR at all thresholds

- **X-axis:** False Positive Rate (FPR)

$$\text{FPR} = \frac{\text{FP}}{\text{FP} + \text{TN}}$$

- **Y-axis:** True Positive Rate (TPR) = Recall

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



# Intuitive Understanding: TPR

**Example: True Positive Rate (TPR)**

$$\text{TPR} = \frac{\text{TP}}{\text{TP} + \text{FN}} = \frac{\text{TP}}{\text{All Actual Positives}}$$

**Question it answers:** Of all actual brick kilns, what fraction did we detect?

- Same as Recall!
- Measures: Sensitivity of the detector
- High TPR = Catches most positives
- Low TPR = Misses many positives

# Intuitive Understanding: FPR

**Example: False Positive Rate (FPR)**

$$\text{FPR} = \frac{\text{FP}}{\text{FP} + \text{TN}} = \frac{\text{FP}}{\text{All Actual Negatives}}$$

**Question it answers:** Of all non-kiln images, what fraction did we
incorrectly flag as having kilns?

- Measures: False alarm rate
- High FPR = Many false alarms
- Low FPR = Few false alarms
- FPR $= 1-$ Specificity

# The ROC Trade-off

**Key Points: Fundamental Trade-off**

As we vary the threshold:

- Lower threshold $\rightarrow$ Higher TPR, Higher FPR
- Higher threshold $\rightarrow$ Lower TPR, Lower FPR

## Low Threshold

- Catch more positives
- But more false alarms
- Top-right of ROC

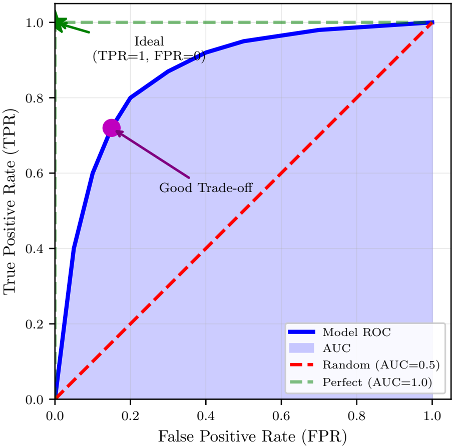
## High Threshold

- Fewer false alarms
- But miss more positives
- Bottom-left of ROC

# Building a ROC Curve: Steps

1. Train classifier, get predicted probabilities
2. For each threshold $\tau \in [0, 1]$:
   - Apply threshold to get predictions
   - Compute confusion matrix
   - Calculate TPR and FPR
   - Plot point (FPR, TPR)
3. Connect points to form curve

# Building a ROC Curve: Interpretation

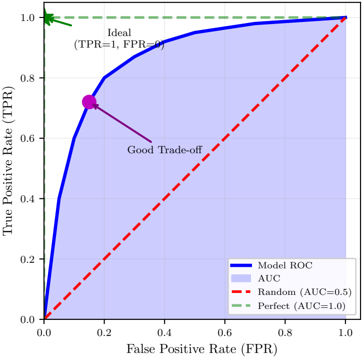



- **Perfect classifier:** Curve hugs top-left corner
- **Random classifier:** Diagonal line

# Interpreting ROC Curves

**Key Points: Good ROC Curve**

- Closer to top-left
- Top-left = perfect!
- TPR=1, FPR=0
- High TPR, low FPR




## Baselines

- **Perfect:** Top-left
- **Random:** Diagonal
- **Bad:** Below diagonal

# Example: Same Dataset

**Example: From Notebook**

Using our Logistic Regression model

| Threshold | TPR (Recall) | FPR |
|---|---|---|
| 0.00 | 1.00 | 1.00 |
| 0.30 | 0.83 | 0.35 |
| 0.50 | 0.69 | 0.23 |
| 0.70 | 0.52 | 0.10 |
| 0.90 | 0.24 | 0.00 |

**Key Points: Observation**

As threshold increases: TPR $\downarrow$, FPR $\downarrow$

# AUC-ROC: Area Under ROC Curve

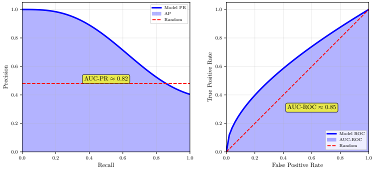
## Definition: AUC-ROC

Single number summarizing entire ROC curve

$$\text{AUC-ROC} = \int_0^1 \text{TPR}(\text{FPR})\, d(\text{FPR})$$

**Interpretation:**

- Range: $[0, 1]$
- Perfect: AUC $= 1.0$
- Random: AUC $= 0.5$
- Higher is better



# AUC-ROC Intuition

**Key Points: Probabilistic Interpretation**

AUC-ROC = Probability that the model ranks
a random positive example higher than
a random negative example

**Example: Example**

- AUC = 0.95: 95% chance model scores a true kiln higher than a non-kiln
- AUC = 0.50: Model is guessing randomly
- AUC = 0.85: Good discrimination ability

# ROC Implementation

## Scikit-learn Implementation

```
from sklearn.metrics import (
    roc_curve, roc_auc_score,
    RocCurveDisplay
)

# Get predicted probabilities
y_scores = model.predict_proba(X_test)[:, 1]

# Compute ROC curve
fpr, tpr, thresholds = roc_curve(y_test, y_scores)
auc_roc = roc_auc_score(y_test, y_scores)

# Visualize
display = RocCurveDisplay(fpr=fpr, tpr=tpr,
                          roc_auc=auc_roc)
display.plot()
```

# Comparing Multiple Models

**Example: From Notebook: 3 Classifiers**

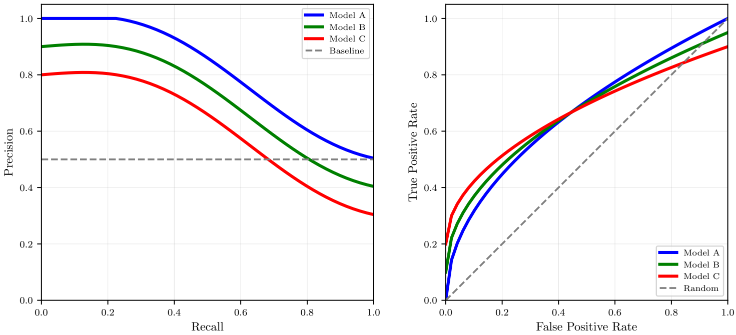
- Logistic Regression (linear boundary)
- Random Forest (non-linear, ensemble)
- SVM with RBF kernel (non-linear)

| **Model** | **AUC-ROC** | **AUC-PR** |
|---|---|---|
| Random Forest | 0.92 | 0.90 |
| SVM (RBF) | 0.89 | 0.87 |
| Logistic Regression | 0.86 | 0.83 |

(Values approximate from notebook example)

# PR vs ROC: When to Use Each

# Comparing PR and ROC Curves




## PR Curve

**Plots:** Precision vs Recall
**Focus:** Positive class
**Sensitive to:** Imbalance

## ROC Curve

**Plots:** TPR vs FPR
**Focus:** Both classes
**Robust to:** Imbalance

# Key Difference: Class Imbalance

## Critical Insight

ROC curves can be overly optimistic
on highly imbalanced datasets!

### Example: Why?

FPR uses TN in denominator:

$$\mathrm{FPR} = \frac{\mathrm{FP}}{\mathrm{FP} + \textcolor{red}{\mathrm{TN}}}$$

With many negatives, even lots of FPs
can give a low FPR

# Example: Imbalanced Data Setup

**Example: Scenario: Highly Imbalanced Dataset**

- **Total images:** 1,000
- **Positive class** (has brick kilns): 50 (5%)
- **Negative class** (no kilns): 950 (95%)

This is a realistic scenario!
Many real-world problems have imbalanced classes

# Example: Imbalanced Data Analysis

## Model with 100 False Positives

Suppose our model produces 100 false alarms:

**Precision impact:**

- Many false alarms per true positive
- Precision will be **low** (obvious problem!)

**FPR appears good:**

$$\text{FPR} = \frac{100}{100 + 850} = \frac{100}{950} = 0.105$$

Even with 100 false positives, FPR is only 10.5%!

**Key Points: Conclusion**

**PR curve:** Shows the problem clearly
**ROC curve:** Can hide issues in imbalanced data

# Practical Considerations

# PR Curves vs ROC Curves

**Key Points: Use PR Curves When:**

- Classes are highly imbalanced
- You care primarily about positive class
- False positives and negatives differ in cost

**Examples:** Rare disease, fraud, information retrieval

**Use ROC Curves When:**

- Classes are relatively balanced
- Both classes equally important

# Why PR for Imbalanced Data?

**Example: Brick Kiln Dataset**

- Total: 10,000 images
- Positive (has kiln): 150 (1.5%)
- Negative (no kiln): 9,850 (98.5%)

**Naive Classifier**

Always predict "no kiln":

- Accuracy: 98.5% (looks great!)
- Precision: undefined
- Recall: 0% (useless!)

# The Problem with Accuracy

**Key Points: Why Accuracy Fails**

With extreme imbalance (1.5% positive):

- Accuracy dominated by majority class
- High accuracy doesn't mean good performance
- Need metrics focused on positive class

Use Precision, Recall, and PR curves!

# Visualization with Scikit-learn

## Complete Implementation

```
from sklearn.metrics import (
    precision_recall_curve,
    average_precision_score,
    PrecisionRecallDisplay
)

# Get scores
y_scores = model.predict_proba(X_test)[:, 1]

# Compute metrics
precision, recall, thresholds = \
    precision_recall_curve(y_test, y_scores)
ap = average_precision_score(y_test, y_scores)

# Visualize
display = PrecisionRecallDisplay(
    precision, recall, average_precision=ap)
```

# Pop Quiz #1

**Answer this!**

**A model detects defective products (2% of all products). Your model achieves:**

- **Precision: 0.60**
- **Recall: 0.90**

**Out of 10,000 products, how many will be flagged?**

A) 150
B) 300
C) 600
D) 900

# Pop Quiz #1

**Answer this!**

**A model detects defective products (2% of all products). Your model achieves:**

- **Precision: 0.60**
- **Recall: 0.90**

**Out of 10,000 products, how many will be flagged?**

A) 150
B) 300
C) 600
D) 900

**Answer: B) 300**

# Pop Quiz Answer

**Example: Solution**

**Answer: B) 300**

**Step 1:** Actual defective products

$$10{,}000 \times 0.02 = 200$$

**Step 2:** True Positives (Recall $= 0.90$)

$$\text{TP} = 200 \times 0.90 = 180$$

**Step 3:** Use precision formula

$$0.60 = \frac{180}{\text{Total flagged}}$$

$$\text{Total flagged} = \frac{180}{0.60} = 300$$

# Pop Quiz #2

**Answer this!**

**Which scenario needs model with Precision=0.70, Recall=0.85 over Precision=0.85, Recall=0.70?**

A) Email spam detection
(false positives lose legitimate mail)

B) Airport security screening
(missing threats is catastrophic)

C) Credit card fraud
(false positives block legitimate purchases)

D) All equally

# Pop Quiz #2

**Answer this!**

**Which scenario needs model with Precision=0.70, Recall=0.85 over Precision=0.85, Recall=0.70?**

A) Email spam detection
(false positives lose legitimate mail)

B) Airport security screening
(missing threats is catastrophic)

C) Credit card fraud
(false positives block legitimate purchases)

D) All equally

**Answer:** **B) Airport security screening**

# Pop Quiz Answer

**Example: Solution**

**Answer: B) Airport security screening**

**Reasoning:**

- First model has **higher recall (0.85)**
- Catches more true positives
- Missing a threat = catastrophic
- Better to have false alarms than miss threats

Options (a) and (c): False positives are costly
$\Rightarrow$ Need high precision

# Summary

# Key Takeaways (1/2)

**Key Points: Core Concepts**

1. **Precision:** Reliability of predictions
2. **Recall:** Completeness of detection
3. **Trade-off:** Can't maximize both
4. **Thresholds:** Control the trade-off

# Key Takeaways (2/2)

**Key Points: Practical Insights**

5. **PR curves:** Show all trade-offs
6. **Application:** Determines best point
7. **Imbalanced data:** PR better than accuracy
8. **Summary metrics:** F1, AP

# Workflow Summary

1. Train classifier
2. Generate PR curve on validation set
3. Analyze precision-recall trade-offs
4. Choose threshold based on:
   - Application requirements
   - Cost of errors
   - Available resources
5. Validate on test set
6. Monitor in production
7. Adjust if requirements change

# The Right Model for YOUR Application

The best model makes the right trade-offs
for **your specific application**

Not the highest accuracy,
not the highest F1,
but the one that aligns with your goals!

# Further Resources

- **Notebook:** `pr-curve.html`
  Running example with visualization code
- **Documentation:**
  Scikit-learn Precision-Recall guide
- **Related topics:**
  - ROC curves and AUC
  - Cost-sensitive learning
  - Threshold optimization
  - Multi-class metrics

# Thank you!

Questions?