#UpSkillWithKalpesh

Day 22

Data Science Unlocked

From Zero to Data Hero

Precision, Recall and its Tradeoffs

Kalpesh Pathade
@DataSimplified

Precision and Recall & It’s Tradeoffs

Type	@DataSimplified

Precision and Recall: A Comprehensive Guide

In machine learning, especially in classification problems, **precision** and **recall** are fundamental evaluation metrics used to measure the effectiveness of a model.

While **accuracy** is a good metric when classes are balanced, it is not always reliable for imbalanced datasets. For example, in fraud detection, spam classification, or medical diagnosis, the number of positive cases is much smaller than negative cases. Precision and recall help in assessing model performance in such cases.

2. Understanding Precision and Recall

Precision and recall are derived from the **confusion matrix**, which consists of:

- **True Positives (TP):** Correctly predicted positive cases.
- **False Positives (FP):** Incorrectly predicted positive cases (actually negative).
- **True Negatives (TN):** Correctly predicted negative cases.
- **False Negatives (FN):** Incorrectly predicted negative cases (actually positive).

The formulas for precision and recall are:

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

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

Example Calculation

Consider a binary classification model that classifies emails as spam or not spam. The confusion matrix is:

	Predicted Positive (Spam)	Predicted Negative (Not Spam)
Actual Positive (Spam)	40 (TP)	10 (FN)
Actual Negative (Not Spam)	5 (FP)	1000 (TN)

- **Precision**:

$\frac{40}{40+5} = \frac{40}{45} \approx 0.89$

- **Recall**:

$\frac{40}{40+10} = \frac{40}{50} = 0.8$

Interpretation

- **High Precision:** Few false positives, meaning that when the model predicts a positive class, it is usually correct.
- **High Recall:** Few false negatives, meaning the model correctly identifies most positive cases.

3. Precision-Recall Trade-off

There is a trade-off between precision and recall. Increasing one often decreases the other:

- **If the model becomes more strict (higher precision):** It reduces false positives but may also miss true positives, leading to lower recall.
- **If the model becomes more lenient (higher recall):** It captures more true positives but also increases false positives, lowering precision.

Balancing Precision and Recall

A balance can be achieved by using the **F1-score**, which is the harmonic mean of precision and recall:

$F1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$

This metric provides a single score to evaluate models, especially for imbalanced datasets.

4. Precision-Recall Curve (PR Curve)

A **Precision-Recall (PR) Curve** is a graphical representation of the trade-off between precision and recall for different threshold values.

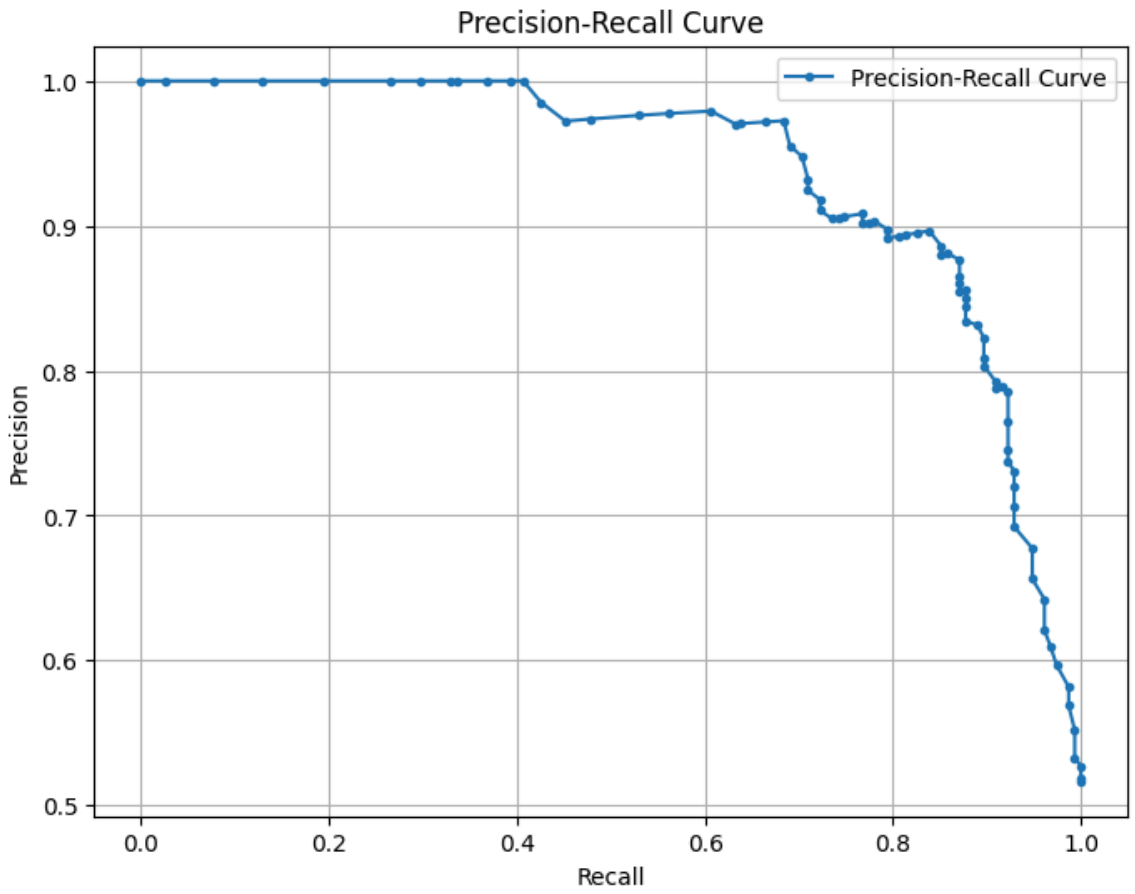
How It Works

1. The classifier outputs probabilities instead of fixed class labels.
2. A threshold determines whether an instance is classified as positive or negative.
3. Lowering the threshold increases recall but reduces precision, and vice versa.
4. The curve is plotted with recall on the x-axis and precision on the y-axis.

Python Implementation

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.metrics import precision_recall_curve
from sklearn.ensemble import RandomForestClassifier
from sklearn.datasets import make_classification
from sklearn.model_selection import train_test_split

# Generate synthetic data
X, y = make_classification(n_samples=1000, n_features=20, random_state=4
2)

# Split into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_stat
e=42)

# Train model
clf = RandomForestClassifier()
clf.fit(X_train, y_train)

# Predict probabilities
y_scores = clf.predict_proba(X_test)[:, 1]
```

```
# Compute precision and recall
precisions, recalls, _ = precision_recall_curve(y_test, y_scores)

# Plot Precision-Recall Curve
plt.figure(figsize=(8, 6))
plt.plot(recalls, precisions, marker='.', label="Precision-Recall Curve")
plt.xlabel("Recall")
plt.ylabel("Precision")
plt.title("Precision-Recall Curve")
plt.legend()
plt.grid()
plt.show()
```



Interpretation of PR Curve

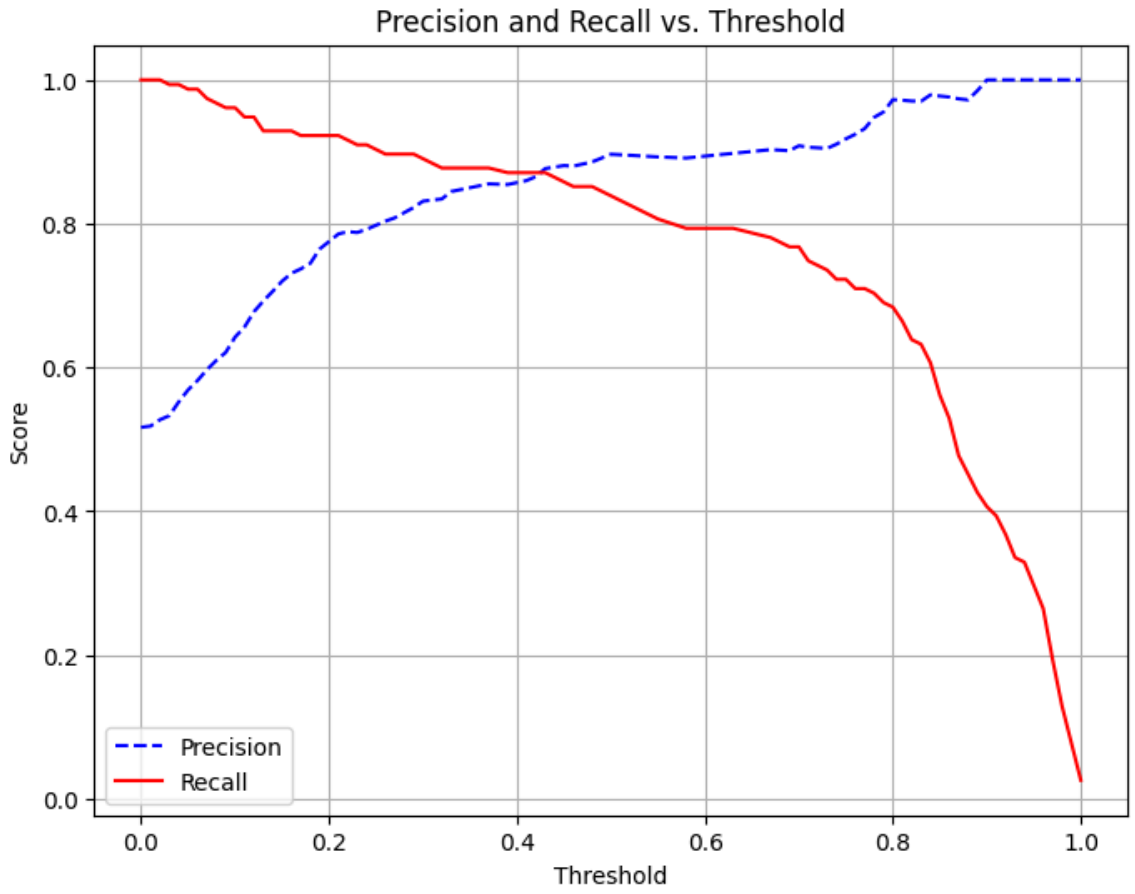
- A model with **high precision and high recall** has a curve close to the **top-right corner**.
- A model with **low precision or recall** has a lower curve.
- A random classifier would result in a horizontal line at the proportion of positive instances in the dataset.

5. Precision and Recall vs. Threshold Curve

A **Precision vs. Recall** curve is a direct plot of precision against recall, without the threshold component.

Python Implementation

```
# Compute precision, recall, and thresholds
precisions, recalls, thresholds = precision_recall_curve(y_test, y_scores)

# Plot Precision and Recall vs Threshold
plt.figure(figsize=(8, 6))
plt.plot(thresholds, precisions[:-1], label="Precision", linestyle="--", color='blu
e')
plt.plot(thresholds, recalls[:-1], label="Recall", linestyle="-", color='red')
plt.xlabel("Threshold")
plt.ylabel("Score")
plt.title("Precision and Recall vs. Threshold")
plt.legend()
plt.grid()
plt.show()
```



Interpretation

- As the **threshold increases**, precision increases (fewer false positives) but recall decreases (more false negatives).
- The optimal threshold depends on the problem domain:
  - **High precision is needed** when false positives are costly (e.g., spam filtering, fraud detection).
  - **High recall is needed** when false negatives are costly (e.g., medical diagnosis).

6. Conclusion

- **Precision and recall** are critical evaluation metrics, especially for imbalanced datasets.
- **Precision-recall trade-off** must be considered depending on the application.

- **Precision-recall curves** provide a visual representation of model performance.
- **Choosing the right threshold** depends on the specific use case.