Essential Guide to Model Evaluation Metrics

Measure what matters: master evaluation metrics to make better clinical predictions.

A detailed and practical guide to model evaluation metrics, with a focus on healthcare and the real-world impact of prediction errors.

rubió
Metabolomics

Ibon Martínez-Arranz | imartinez@labrubiocom
Data Science Manager at Rubió Metabolomics
www.rubiometabolomics.com

Ibon Martínez-Arranz, PhD in Mathematics and Statistics, holds MSc degrees in Applied Statistical Techniques and Mathematical Modeling. He has extensive expertise in statistical modeling and advanced data analysis techniques. Since 2017, he has led the Data Science area at Rubió Metabolomics, driving predictive model development and statistical computing management for metabolomics, data handling, and R&D projects. His doctoral research in Mathematics investigated and adapted genetic algorithms to improve the classification of NAFLD subtypes.

Itziar Mincholé Canals | iminchole@labrubio.com
Data Specialist at Rubió Metabolomics
www.rubiometabolomics.com

Itziar Mincholé Canals has a degree in physical sciences (1999) from the University of Zaragoza (Spain). In 1999 she began developing her professional career as a software analyst and developer, mainly working on web development projects and database programming in different sectors. In 2009 she joined OWL Metabolomics as a bioinformatician, where she has participated in several R&D projects and has developed several software tools for metabolomics. In 2016 she completed the master's degree in applied statistics with R software. After joining the Data Science department, she also supports data analysis.

Essential Guide to Model Evaluation Metrics

Ibon Martínez-Arranz

Life
Feels
Good

rubió
Metabolomics

Contents

Essential Guide to Model Evaluation Metrics

Introduction to Model Evaluation

Why Measuring Model Performance Matters

In data science, building a predictive model is only half the battle — the other half lies in measuring how well that model performs. A model is only as useful as the reliability of the decisions it supports. Especially in critical applications like medicine, where models may inform diagnoses or treatments, understanding evaluation metrics is not just technical diligence; it's a moral responsibility.

A predictive model that classifies whether patients are sick or healthy must do more than just "get it right" most of the time. It must minimize harmful mistakes: **false positives**, where healthy patients are mistakenly labeled as sick, and **false negatives**, where sick patients are declared healthy — a potentially life-threatening error. Hence, evaluating models with appropriate metrics is essential to ensure their real-world utility and safety.

Before diving into specific metrics, we need to understand how we represent a model's predictions against reality. This is where the **confusion matrix** becomes an indispensable tool.

A Motivating Example: Diagnosing a Disease

Let's consider a simplified yet realistic example. Imagine we have developed a machine learning model to diagnose a rare disease based on blood test results.

The dataset contains 1,000 patient records. Of these, 50 patients actually have the disease (positive cases), and 950 do not (negative cases).

Now suppose our model predicts **everyone is healthy** — it simply says "no disease" for every patient. In this case, it would be **95% accurate**, since it correctly labeled 950 out of 1,000 patients. But it failed to detect **any** of the 50 truly sick patients. That's not just a poor model — it's a dangerous one.

Accuracy alone, as we'll see, is often misleading in imbalanced datasets. That's why we need other metrics to evaluate model performance more carefully. And our journey begins with the confusion matrix.

Understanding the Confusion Matrix

The **confusion matrix** is a compact yet powerful tool that summarizes a classifier's predictions. It tells us not only how many predictions were correct, but what kinds of mistakes the model made. Here's a standard 2×2 confusion matrix for binary classification:

	Predicted Positive	**Predicted Negative**
Actual Positive	True Positive (TP)	False Negative (FN)
Actual Negative	False Positive (FP)	True Negative (TN)

- **True Positives (TP):** The model correctly predicts a patient has the disease.
- **False Positives (FP):** The model incorrectly predicts disease in a healthy patient.
- **False Negatives (FN):** The model fails to identify a patient who actually has the disease.
- **True Negatives (TN):** The model correctly predicts a patient is healthy.

This structure is the foundation for most classification metrics. It allows us to compute quantities like **precision**, **recall**, **specificity**, and **F1 score**, all of which we'll explore in later chapters.

Implementing a Confusion Matrix in Python

Python's `scikit-learn` makes it easy to compute and visualize confusion matrices:

```python
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
import matplotlib.pyplot as plt

y_true = [1, 0, 1, 1, 0, 1, 0, 0, 1, 0]  # actual labels
y_pred = [1, 0, 1, 0, 0, 1, 0, 1, 1, 0]  # predicted labels

cm = confusion_matrix(y_true, y_pred)
disp = ConfusionMatrixDisplay(confusion_matrix=cm)
disp.plot()
plt.title("Confusion Matrix Example")
plt.show()
```

```python
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
import matplotlib.pyplot as plt

y_true = [1, 0, 1, 1, 0, 1, 0, 0, 1, 0]  # actual labels
y_pred = [1, 0, 1, 0, 0, 1, 0, 1, 1, 0]  # predicted labels

cm = confusion_matrix(y_true, y_pred)
disp = ConfusionMatrixDisplay(confusion_matrix=cm)
disp.plot()
plt.title("Confusion Matrix Example")
plt.show()
```

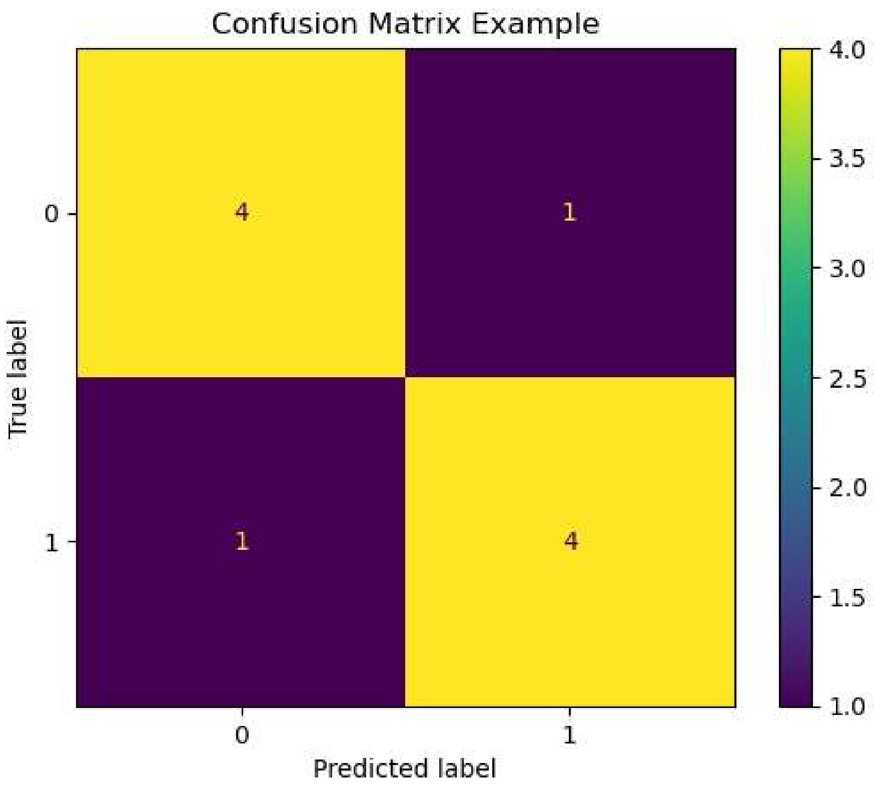

Figure 1: Image generated by the provided code.

This snippet computes and plots the confusion matrix from real vs. predicted values. In medical scenarios, where decision impact is critical, visual tools like this can help data scientists communicate performance more effectively to clinicians and stakeholders.

Transition to the Next Chapter

In this chapter, we've introduced why evaluation matters, explored a medical example, and unpacked the logic and structure of the confusion matrix. But the matrix is only the beginning — the numbers it contains can be transformed into various metrics that reveal different aspects of model behavior.

In the next chapter, we'll dive into these **basic classification metrics** — starting with **accuracy**, but quickly moving beyond it to uncover why precision, recall, and F1 score are more informative in medical settings. We'll connect each metric to its clinical relevance, and show how you can compute them easily in Python.

Basic Classification Metrics

Introduction

Now that we understand the confusion matrix, we can derive from it a set of metrics that quantify a model's performance from different perspectives. Each metric answers a slightly different question and is more or less relevant depending on the clinical context.

In healthcare, we often care more about **detecting sick patients (sensitivity)** or **avoiding unnecessary treatments (specificity)** than we do about overall correctness (**accuracy**). In this chapter, we'll explore:

- What each metric means
- How to calculate it
- When and why it's important
- How to compute it in Python
- How to visualize it geometrically

Accuracy

Definition: The proportion of correct predictions (both positive and negative) over all predictions.

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

Usefulness: Accuracy can be misleading in imbalanced datasets. If 95% of patients are healthy, a model predicting "healthy" for everyone will achieve 95% accuracy — but fail to detect any sick patients.

Python:

```python
from sklearn.metrics import accuracy_score
accuracy_score(y_true, y_pred)
```

Sensitivity (Recall)

Definition: The proportion of actual positives (sick patients) that were correctly identified.

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

Also known as **recall**, this metric answers: *"Of all the sick patients, how many did the model catch?"*

Clinical relevance: Missing a diagnosis (a false negative) can be dangerous. High sensitivity is crucial when the cost of missing a disease is high.

Python:

```python
from sklearn.metrics import recall_score
recall_score(y_true, y_pred)
```

Specificity

Definition: The proportion of actual negatives (healthy patients) that were correctly identified.

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

This metric answers: *"Of all the healthy patients, how many were correctly identified as healthy?"*

Clinical relevance: A low specificity means many healthy people are falsely diagnosed, possibly undergoing unnecessary stress or treatments.

> *Note*: `scikit-learn` does not include specificity by default, but you can calculate it manually.

```python
cm = confusion_matrix(y_true, y_pred)
TN, FP, FN, TP = cm.ravel()
specificity = TN / (TN + FP)
```

Precision

Definition: The proportion of predicted positives that were actually positive.

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

Usefulness: Precision tells us *how much we can trust positive predictions*. In medical contexts where treatment is expensive or risky, precision matters.

Python:

```python
from sklearn.metrics import precision_score
precision_score(y_true, y_pred)
```

F1 Score

Definition: The harmonic mean of precision and recall. It balances the trade-off between both metrics.

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

Usefulness: The F1 score is especially useful in imbalanced scenarios where both false positives and false negatives are costly.

Python:

```python
from sklearn.metrics import f1_score
f1_score(y_true, y_pred)
```

Summary Table

Metric	Formula	Clinical Meaning
Accuracy	$\frac{TP+TN}{TP+TN+FP+FN+kk}$	Overall correctness
Sensitivity	$\frac{TP}{TP+FN}$	Ability to detect sick patients
Specificity	$\frac{TN}{TN+FP}$	Ability to avoid false alarms in healthy patients
Precision	$\frac{TP}{TP+FP}$	Trustworthiness of a positive prediction
F1 Score	Harmonic mean of precision and recall	Balances sensitivity and precision

Looking Ahead

Now that we have the fundamental metrics covered, we are ready to dive deeper into **evaluation through curves and thresholds**. In the next chapter, we will explore **ROC curves**, **AUC**, and **Precision-Recall curves** — tools that show how performance changes as the decision threshold varies, which is particularly important in clinical risk prediction.

Curves and Area Under the Curve (AUC)

Introduction

So far, we've evaluated models with fixed thresholds — classifying predictions as positive or negative based on a decision boundary (e.g., 0.5). However, many models output probabilities rather than hard labels. In these cases, we can adjust the threshold to explore different trade-offs between sensitivity and specificity or between precision and recall.

This is where **ROC curves** and **Precision-Recall curves** become powerful tools. They allow us to visualize a model's performance across all thresholds, helping us make better decisions depending on clinical priorities.

ROC Curve and AUC

What is a ROC Curve?

The **Receiver Operating Characteristic (ROC) curve** plots the **true positive rate (sensitivity)** against the **false positive rate** at various threshold settings.

- **X-axis:** False Positive Rate = FP / (FP + TN)
- **Y-axis:** True Positive Rate = TP / (TP + FN)

Each point on the ROC curve corresponds to a different threshold. The closer the curve is to the top-left corner, the better the model is at distinguishing between classes.

Area Under the ROC Curve (AUC-ROC)

The **Area Under the Curve (AUC)** provides a single value summarizing the entire ROC curve. It represents the probability that the model ranks a randomly chosen positive instance higher than a randomly chosen negative one.

AUC Value	Interpretation	Explanation
1.0	Perfect model	Indicates a model that can classify all samples perfectly without any errors.
0.9	Excellent	Represents a model with high discriminatory power, very close to perfect classification.
0.8	Very good	Signifies a model with strong discrimination ability, though not as high as 0.9.
0.7	Reasonable	Denotes a model with some discriminatory capability, room for improvement may exist.
0.6	Relatively poor	Indicates a model struggling to distinguish between positive and negative samples.
0.5	Random guessing	Represents a model that performs no better than random chance in classification tasks.

Clinical Interpretation

In a medical setting, AUC helps answer: *“How well can the model rank sick patients above healthy ones?”*

However, AUC-ROC can be misleading in **imbalanced datasets**. For example, in rare diseases where negatives dominate, a high AUC may hide poor performance on positives.

Python Example

```python
from sklearn.metrics import roc_curve, auc
import matplotlib.pyplot as plt

y_scores = model.predict_proba(X_test)[:, 1]  # probabilities for positive class
fpr, tpr, thresholds = roc_curve(y_true, y_scores)
roc_auc = auc(fpr, tpr)

plt.plot(fpr, tpr, label=f"ROC curve (AUC = {roc_auc:.2f})")
plt.plot([0, 1], [0, 1], 'k--')
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("ROC Curve")
plt.legend()
plt.show()
```

Precision-Recall Curve and AUC PR

What is a Precision-Recall Curve?

The **Precision-Recall (PR) curve** plots **precision** against **recall** (sensitivity) at different thresholds.

- **X-axis:** Recall
- **Y-axis:** Precision

It is especially informative for **imbalanced datasets**, where ROC curves might present an overly optimistic view.

AUC-PR

The **area under the PR curve (AUC-PR)** provides a summary of the model's ability to maintain precision as recall increases. It's typically lower than AUC-ROC but more reflective of real performance in rare-event settings like disease detection.

Clinical Interpretation

A high precision at high recall means the model can detect most sick patients *without* flooding the clinic with false alarms. This is a crucial balance when deploying screening tools.

Python Example

```python
from sklearn.metrics import precision_recall_curve,
    average_precision_score

precision, recall, thresholds = precision_recall_curve(
    y_true, y_scores)
ap = average_precision_score(y_true, y_scores)

plt.plot(recall, precision, label=f"PR curve (AP = {ap:.2
    f})")
plt.xlabel("Recall")
plt.ylabel("Precision")
plt.title("Precision-Recall Curve")
plt.legend()
plt.show()
```

Trade-offs and Threshold Tuning

The choice of threshold impacts:

- **Sensitivity:** Higher threshold -> fewer false positives, but also more false negatives.
- **Precision:** Lower threshold -> more positives predicted, but potentially lower precision.

In medicine, this trade-off must align with clinical goals:

- **Early screening:** Prefer high sensitivity (low threshold).
- **Confirmatory diagnosis:** Prefer high precision (high threshold).

Suggested Interactive Visualization

The scikit-learn documentation has interactive notebooks that show how metrics change with thresholds. You can also use `plotly` or `ipywidgets` to build threshold sliders.

```python
# Optional: Interactive threshold tuning
from ipywidgets import interact

def update(threshold=0.5):
    y_pred_thresholded = (y_scores >= threshold).astype(
        int)
    acc = accuracy_score(y_true, y_pred_thresholded)
    recall = recall_score(y_true, y_pred_thresholded)
    precision = precision_score(y_true,
        y_pred_thresholded)
    print(f"Threshold: {threshold:.2f} - Accuracy: {acc
        :.2f}, Recall: {recall:.2f}, Precision: {
        precision:.2f}")
```

```python

interact(update, threshold=(0.0, 1.0, 0.05));
```

Summary Table

The following Table provides a concise overview of two commonly used evaluation metrics in machine learning: ROC (Receiver Operating Characteristic) Curve and Precision-Recall Curve. These curves offer insights into the performance of classification models by examining different aspects of their predictions.

Curve Type	Axes	Best For	AUC Meaning
ROC Curve	TPR vs. FPR	General model discrimination	Probability a positive is ranked above a negative
Precision-Recall	Precision vs. Recall	Imbalanced classes, rare disease detection	Average precision across thresholds

---

Conclusion and What's Next

ROC and PR curves provide a **dynamic** view of model performance across thresholds. They help clinicians and data scientists choose models and thresholds that align with real-world needs.

In the next chapter, we'll go beyond standard metrics to explore **less common but powerful ones** — such as **Balanced Accuracy**, **Matthews Correlation Coefficient**, and **Cohen's Kappa** — particularly useful in cases where traditional metrics fall short.

Advanced Classification Metrics

Introduction

Basic metrics like accuracy and F1 score are often sufficient to evaluate models — but not always. In situations with **class imbalance**, **multiple raters**, or a need for **more nuanced comparisons**, advanced metrics provide deeper insights. In healthcare, where subtle mistakes can have serious consequences, these metrics help us evaluate models more rigorously and fairly.

In this chapter, we will cover:

- **Balanced Accuracy** – adjusts for class imbalance.
- **Matthews Correlation Coefficient (MCC)** – a balanced metric even for skewed classes.
- **Cohen's Kappa** – agreement beyond chance.
- **Youden's Index** – useful for threshold optimization in diagnostics.
- **Lift and Gain Charts** – for evaluating probabilistic models in decision-making pipelines.

Balanced Accuracy

Definition

Balanced Accuracy is the **average of sensitivity and specificity**:

$$\text{Balanced Accuracy} = \frac{1}{2}\left(\frac{TP}{TP+FN} + \frac{TN}{TN+FP}\right)$$

It is especially useful for **imbalanced datasets**, as it gives equal weight to the performance on each class.

Clinical Relevance

Imagine a rare disease affecting only 5% of patients. A model could achieve high accuracy by simply predicting everyone as healthy — but balanced accuracy penalizes such behavior, revealing poor sensitivity.

Python Example

```python
from sklearn.metrics import balanced_accuracy_score

balanced_accuracy_score(y_true, y_pred)
```

Matthews Correlation Coefficient (MCC)

Definition

MCC is a correlation coefficient between observed and predicted classifications:

$$\text{MCC} = \frac{(TP \cdot TN) - (FP \cdot FN)}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}}$$

It returns a value between -1 (total disagreement) and +1 (perfect prediction), with 0 indicating random prediction.

Clinical Relevance

Unlike F1 or accuracy, MCC is **invariant to class distribution**. It is ideal when **class imbalance is severe**, and **false positives and negatives** are equally important.

Python Example

```python
from sklearn.metrics import matthews_corrcoef

matthews_corrcoef(y_true, y_pred)
```

Cohen's Kappa

Definition

Cohen's Kappa measures **inter-rater agreement**, adjusting for the agreement expected by chance:

$$\kappa = \frac{p_o - p_e}{1 - p_e}$$

Where: - $ p_o $ is the observed agreement. - $ p_e $ is the expected agreement by chance.

Clinical Relevance

Kappa is useful when comparing predictions from two sources (e.g., model vs. doctor), highlighting whether agreement is meaningful or just random.

Python Example

```python
from sklearn.metrics import cohen_kappa_score

cohen_kappa_score(y_true, y_pred)
```

Youden's Index

Definition

Youden's Index summarizes the performance of a diagnostic test:

$$J = \text{Sensitivity} + \text{Specificity} - 1$$

This index ranges from 0 (worthless test) to 1 (perfect test). It's often used to determine the **optimal threshold** for binary classifiers.

Clinical Relevance

In medicine, choosing the threshold that maximizes Youden's Index ensures the best trade-off between detecting disease and avoiding false alarms.

Python Example (manual):

```python
import numpy as npy
def youden_index(y_true, y_scores):
    from sklearn.metrics import roc_curve
    fpr, tpr, thresholds = roc_curve(y_true, y_scores)
    J = tpr - fpr
    idx = npy.argmax(J)
    return thresholds[idx], J[idx]

optimal_thresh, index_val = youden_index(y_true, y_scores
    )
```

Lift and Gain Charts

What Are They?

Lift and gain charts show how much **better a model performs compared to random selection** when we rank patients by predicted probability and then select the top X%.

- **Gain chart:** Shows the cumulative % of true positives captured as we increase the % of population screened.
- **Lift chart:** Shows the ratio of detected positives to what would be expected by random selection.

Clinical Relevance

These charts are useful when deploying models to **prioritize screenings** or **allocate limited diagnostic resources**. For example, testing the top 10% of patients by risk might identify 60% of true positives — a lift of 6x over random screening.

Python Example with `scikit-plot`

```python
import scikitplot as skplt
import matplotlib.pyplot as plt

skplt.metrics.plot_cumulative_gain(y_true, model.
    predict_proba(X_test))
```

```python
plt.title("Cumulative Gain Chart")
plt.show()

skplt.metrics.plot_lift_curve(y_true, model.predict_proba
    (X_test))
plt.title("Lift Curve")
plt.show()
```

Summary Table

Metric	What it Measures	Best Use Case
Balanced Accuracy	Mean of sensitivity and specificity	Imbalanced datasets
MCC	Overall classification quality	All cases, especially with imbalance
Cohen's Kappa	Agreement beyond chance	Comparing model vs expert predictions
Youden's Index	Optimal diagnostic threshold	Choosing clinical thresholds
Lift / Gain Charts	Screening performance at ranked thresholds	Prioritizing patient testing or interventions

---

Conclusion and What's Next

Advanced metrics give us more power to evaluate models **fairly** and **usefully**, especially in challenging clinical conditions. Some are mathematically complex, but they offer richer understanding when accuracy and F1 are not enough.

Next, we will explore **how metrics change in real-world conditions**: what happens when data is unbalanced, noisy, or when predictions are probabilistic instead of binary. We'll also discuss **calibration** — an often overlooked but essential concept in healthcare risk prediction.

Clinical Interpretation of Evaluation Metrics

Introduction

So far, we've focused on defining and calculating performance metrics. But understanding their **clinical implications** is just as important — and often more challenging. A great metric in one context can be dangerously misleading in another.

In this chapter, we explore: - The real-world meaning of **false positives and false negatives** - How to quantify the **cost of errors** - How to **choose the optimal threshold** based on clinical needs

What Do False Positives and False Negatives Really Mean?

Let's recall our confusion matrix:

	Predicted Positive	**Predicted Negative**
Actual Positive	True Positive (TP)	False Negative (FN)

	Predicted Positive	**Predicted Negative**
Actual Negative	False Positive (FP)	True Negative (TN)

False Negative (FN):

A sick patient is classified as healthy.

Clinical implications:

- Disease goes undetected.
- No treatment is given.
- Disease may progress unnoticed.
- Higher risk of complications or death.

> Example: A patient with early-stage cancer is classified as healthy -> no follow-up scan or biopsy -> late-stage detection.

False Positive (FP):

A healthy patient is classified as sick.

Clinical implications:

- Patient may undergo unnecessary testing or treatment.
- Psychological stress and anxiety.
- Possible side effects from unnecessary interventions.
- Resource waste in health systems.

> Example: A patient is falsely diagnosed with heart disease -> sent for expensive and invasive tests -> mental burden and costs.

Clinical Cost of Errors: Why One Size Doesn’t Fit All

The **impact of errors** varies depending on the disease, the healthcare system, and the availability of treatments. For some conditions, missing a diagnosis (FN) is much worse than overdiagnosis (FP). For others, it’s the reverse.

Disease Context	FP Cost	FN Cost
Cancer screening	Psychological stress, testing	Missed early treatment opportunity
Infectious diseases	Quarantine, stigma	Further spread of infection
Genetic disorders	Counseling, lifestyle changes	Loss of preventive measures
COVID triage (2020)	ICU bed occupied needlessly	Missed emergency support

Choosing the Optimal Threshold

Most models output **probabilities**. By default, we often use a 0.5 threshold to classify patients — but this is arbitrary. The best threshold depends on the **clinical context** and the **relative cost of mistakes**.

Example: Adjusting the Threshold

```python
import numpy as npy
from sklearn.metrics import precision_score, recall_score

thresholds = npy.arange(0.0, 1.01, 0.05)
for t in thresholds:
    y_pred = (y_scores >= t).astype(int)
    p = precision_score(y_true, y_pred)
    r = recall_score(y_true, y_pred)
    print(f"Threshold: {t:.2f} | Precision: {p:.2f},
        Recall: {r:.2f}")
```

This helps us **visualize the trade-off**: as threshold increases, **precision rises**, but **recall falls** — and vice versa.

Tools for Threshold Selection

Youden's Index

Choose the threshold that maximizes:

$$\text{Sensitivity} + \text{Specificity} - 1$$

Cost-sensitive Analysis

Use weighted losses for false positives vs. false negatives. For example:

```
import numpy as npy
def cost_sensitive_score(y_true, y_scores, fp_cost=1,
    fn_cost=5):
    thresholds = npy.arange(0.0, 1.01, 0.01)
    best_thresh = 0
    min_cost = float("inf")
    for t in thresholds:
        y_pred = (y_scores >= t).astype(int)
        cm = confusion_matrix(y_true, y_pred)
        TN, FP, FN, TP = cm.ravel()
        cost = FP * fp_cost + FN * fn_cost
        if cost < min_cost:
            min_cost = cost
```

```
                best_thresh = t
        return best_thresh, min_cost
```

Expected Utility or Net Benefit

A formal approach from decision theory (often used in diagnostic test evaluation).

Clinical Example: Diabetes Risk Prediction

Suppose you have a model that predicts the risk of type 2 diabetes. You can tune your threshold based on:

- If your goal is **early intervention**, prioritize **high sensitivity** (lower threshold).
- If you have **limited capacity** for follow-up testing, prioritize **high precision** (higher threshold).

> **Figure suggestion**: A slider-based plot that updates the confusion matrix and cost scores as the threshold is changed (see previous chapter's interactive widget).

Summary Table

Factor	Low Threshold	High Threshold
Sensitivity	High	Low
Precision	Low	High
False Negatives	Fewer	More

Factor	Low Threshold	High Threshold
False Positives	More	Fewer
Best For	Screening, early detection	Confirmatory diagnosis

Conclusion and What's Next

Metrics alone are meaningless without context. In medicine, we must always ask: *What happens to a real person if the model is wrong?*

Choosing the right threshold is an ethical and practical decision. In the next chapter, we'll explore how to deal with **imbalanced datasets**, where traditional metrics fail and advanced techniques are required to train and evaluate reliable models.

Evaluation under Class Imbalance

Introduction

In medical applications, we often encounter **class imbalance**: diseases are typically **rare**, meaning that the majority of patients in our dataset are healthy. This creates a serious challenge for model evaluation.

A model that simply predicts “healthy” for everyone can appear deceptively good if evaluated with the wrong metric. In this chapter, we will explore:

- Why **accuracy** can be misleading
- How to handle imbalance through **resampling techniques**
- Which **metrics remain robust** under imbalance

Why Accuracy Can Be Misleading

Let’s say only **5% of patients** in a dataset have a disease. That means 95% are healthy.

If a model **always predicts “healthy”**, it will be:

$$\text{Accuracy} = \frac{950}{1000} = 95\%$$

That sounds great — but it’s **useless**. The model **misses every sick patient**. In this context, **accuracy rewards the majority class**, regardless of clinical utility.

Metrics That Survive Imbalance

Here are metrics that provide meaningful insights even when one class is rare:

- **Sensitivity (Recall):** Measures the ability to find sick patients.
- **Precision:** Measures how many predicted positives are actually correct.
- **F1 Score:** Harmonic mean of precision and recall.
- **Balanced Accuracy:** Averages performance on both classes.
- **MCC (Matthews Correlation Coefficient):** Invariant to class size.
- **AUC-PR:** More informative than AUC-ROC in skewed datasets.

> *In rare-disease prediction, always report metrics beyond accuracy.*

Practical Example in Python

```python
from sklearn.metrics import classification_report
from sklearn.datasets import make_classification
from sklearn.linear_model import LogisticRegression

X, y = make_classification(n_samples=1000, n_classes=2,
    weights=[0.95, 0.05], flip_y=0,
                           n_features=20, n_informative
                               =3, random_state=42)

model = LogisticRegression()
model.fit(X, y)
y_pred = model.predict(X)

print(classification_report(y, y_pred, digits=3))
```

Resampling Techniques

To combat imbalance, we can **adjust the data** or **weight the algorithm**.

Undersampling

Randomly remove examples from the majority class to balance the dataset.

```python
from imblearn.under_sampling import RandomUnderSampler

rus = RandomUnderSampler()
X_res, y_res = rus.fit_resample(X, y)
```

Pros: - Simple and fast - Preserves the minority class

Cons: - May discard useful data

Oversampling

Duplicate or synthetically generate minority class examples.

```python
from imblearn.over_sampling import SMOTE

smote = SMOTE()
X_res, y_res = smote.fit_resample(X, y)
```

Pros: - No data loss - Helps the model learn patterns in rare classes

Cons: - Risk of overfitting if oversampled naively

Class Weighting

Many models (like logistic regression, SVMs, and decision trees) allow you to **penalize errors** on minority class more heavily.

```python
model = LogisticRegression(class_weight='balanced')
model.fit(X, y)
```

This approach does **not change the dataset**, only how the model interprets errors.

Visualizing the Impact

You can visualize how different strategies affect model performance:

```python
from sklearn.metrics import roc_auc_score

model.fit(X, y)
print("Baseline AUC-ROC:", roc_auc_score(y, model.
    predict_proba(X)[:, 1]))

model_resampled = LogisticRegression()
model_resampled.fit(X_res, y_res)
print("Resampled AUC-ROC:", roc_auc_score(y,
    model_resampled.predict_proba(X)[:, 1]))
```

Also useful: **precision-recall curves**, which respond more clearly to rare-class improvement.

Summary Table

Method	Goal	Risks
Undersampling	Balance by reducing majority	Loss of information
Oversampling / SMOTE	Boost minority class	Overfitting, synthetic bias
Class weighting	Penalize minority errors	May still overfit noisy cases
Balanced Accuracy	Adjust for imbalance	Less interpretable alone
AUC-PR	Highlight rare class quality	Requires probabilistic output

Method	Goal	Risks
MCC	Robust to imbalance	Harder to explain to clinicians

Clinical Case Study: Rare Genetic Disorder

Imagine you're building a model to detect a rare genetic disorder with a **prevalence of 1%**.

- A naive model will simply predict "healthy" for everyone.
- A well-designed model will **leverage SMOTE** or **adjust class weights** to detect the minority.
- Metrics like **F1 score** and **AUC-PR** will expose the difference in quality.

Suggested Visual

You can adapt a **precision-recall curve** on imbalanced vs. resampled datasets to show how oversampling improves recall at low thresholds. Try using:

- `scikit-plot` for quick plots.
- `plotly` for interactive curves.

Conclusion and What's Next

In the real world, data is rarely clean or balanced. Understanding how imbalance distorts evaluation — and what to do about it — is key to deploying reliable models, especially in healthcare.

In the next chapter, we'll look at **probabilistic predictions** and **model calibration** — making sure that predicted probabilities actually reflect real-world risk, an essential step for clinical decision-making.

Metrics for Probabilistic Models

Introduction

Most modern classifiers — logistic regression, random forests, gradient boosting, neural networks — don't just give a binary prediction. They produce **probabilities**. This allows for more nuanced decision-making: in clinical settings, it helps doctors assess **risk**, **prioritize cases**, and **allocate resources**.

But this introduces a new challenge: how do we **evaluate the quality of probabilities**?

In this chapter, we explore: - **Brier Score**: for measuring the accuracy of probabilistic forecasts - **Log Loss**: penalizing confidence in incorrect predictions - **Calibration**: ensuring that predicted probabilities match observed frequencies - **Reliability curves**: to visualize calibration

Brier Score

Definition

The **Brier Score** is the mean squared error between the predicted probability and the true class label:

$$\text{Brier Score} = \frac{1}{N}\sum_{i=1}^{N}(p_i - y_i)^2$$

Where:

- p_i is the predicted probability of the positive class
- $y_i \in \{0, 1\}$ is the actual label

Values range from 0 (perfect prediction) to 1 (worst).

Clinical Relevance

It penalizes **miscalibration** and **incorrect confidence**. A model that is correct but overly confident (predicting 0.99 when the outcome is 1) will be penalized more than a cautious model predicting 0.7.

Python Example

```python
from sklearn.metrics import brier_score_loss

brier_score_loss(y_true, y_prob)
```

Log Loss (Cross-Entropy Loss)

Definition

Log loss (a.k.a. **binary cross-entropy**) measures the negative log-likelihood of the true label given the predicted probability:

$$\text{Log Loss} = -\frac{1}{N}\sum_{i=1}^{N}[y_i \log(p_i) + (1 - y_i)\log(1 - p_i)]$$

This metric **strongly penalizes incorrect high-confidence predictions**.

Clinical Relevance

Log loss is more sensitive than the Brier Score. It rewards **well-calibrated, confident** predictions and severely punishes **wrong, overconfident ones**.

Python Example

```python
from sklearn.metrics import log_loss

log_loss(y_true, y_prob)
```

> Always ensure `y_prob` are probabilities, not class labels.

Calibration: Do Probabilities Reflect Reality?

Calibration refers to how well the predicted probabilities match actual outcomes.

- A **well-calibrated** model that outputs `0.8` for 100 patients should be correct about 80 times.
- A **poorly calibrated** model may output extreme probabilities (0.99 or 0.01) without justification.

Poor calibration can lead to **over-treatment or under-treatment** in medical decision-making.

Reliability Diagrams (Calibration Curves)

These curves plot **predicted probability** vs **observed frequency** of positives. A perfectly calibrated model lies on the **diagonal**.

Python Example

```python
from sklearn.calibration import calibration_curve
import matplotlib.pyplot as plt

prob_true, prob_pred = calibration_curve(y_true, y_prob,
    n_bins=10)

plt.plot(prob_pred, prob_true, marker='o', label='Model')
plt.plot([0, 1], [0, 1], linestyle='--', label='Perfectly
     calibrated')
plt.xlabel('Mean predicted probability')
plt.ylabel('Fraction of positives')
plt.title('Calibration Curve (Reliability Diagram)')
plt.legend()
plt.show()
```

How to Improve Calibration

Some models (like **logistic regression**) are naturally well-calibrated. Others (e.g., **random forests**, **gradient boosting**) may need post-processing:

Platt Scaling

Fits a logistic regression on the model outputs.

Isotonic Regression

Fits a non-parametric monotonic function for flexible calibration.

Python Example with `CalibratedClassifierCV`

```python
from sklearn.calibration import CalibratedClassifierCV

cal_model = CalibratedClassifierCV(base_estimator=model,
    method='isotonic', cv=5)
cal_model.fit(X_train, y_train)
```

When Does Calibration Matter?

Situation	Importance of Calibration
Binary decision only	Low
Triage / prioritization	High
Communicating risk to clinicians	Very high
Integrating model into EHR systems	Extremely high

Summary Table

Metric	Measures	Best For
Brier Score	Squared error of probability	General probabilistic accuracy

Metric	Measures	Best For
Log Loss	Likelihood penalization	Model optimization, high sensitivity
Calibration Curve	Visual check of calibration	Clinical risk models
Calibrated Models	Probability alignment	Safety-critical applications

Conclusion and What's Next

In medicine, probability isn't just a number — it's a signal to act. Miscalibrated probabilities can lead to **overconfidence** or **misdiagnosis**.

In the final chapter, we'll cover **explainability, fairness, and ethical use** of metrics and models. A model that performs well but hides its logic or embeds bias can do more harm than good — even with perfect AUC.

Ethics, Fairness, and Interpretability

Introduction

A model can be **accurate**, **well-calibrated**, and **statistically robust**, and still be **unethical** or **harmful**.

In healthcare, machine learning is not just a technical exercise — it's a social one. Predictive models can **amplify health disparities**, **exclude vulnerable groups**, or **make decisions no human can understand**.

In this final chapter, we examine:

- What it means for a model to be **fair**
- How **bias** can emerge — even from well-intentioned data
- How to evaluate models ethically and transparently
- Tools and techniques to improve **interpretability**

Bias in Medical Models

Bias can appear in several forms:

Type of Bias	Example
Representation	Minorities underrepresented in training data

Type of Bias	Example
Measurement	Data collection varies across demographics
Historical	Models trained on data that reflect systemic inequities
Labeling	Labels reflect subjective or biased clinician decisions

Case Study: Pulse Oximetry

Recent studies revealed that **pulse oximeters** may be less accurate in individuals with darker skin tones — a form of **measurement bias** that, if replicated in predictive models, could exacerbate inequalities in triage and treatment.

Fairness Metrics

Several metrics help assess fairness:

Demographic Parity

The model's output is independent of a protected attribute (e.g., race, sex):

$$P(\hat{Y} = 1 \mid A = 0) = P(\hat{Y} = 1 \mid A = 1)$$

Equal Opportunity

Equal true positive rates across groups:

$$P(\hat{Y} = 1 \mid Y = 1, A = 0) = P(\hat{Y} = 1 \mid Y = 1, A = 1)$$

Equalized Odds

Equal false positive and true positive rates:

$$P(\hat{Y} = 1 \mid Y, A) \text{ is the same across groups}$$

Interpretability: Making Models Understandable

In healthcare, a model's decision **must be explainable**. Clinicians need to trust — and sometimes **justify** — model outputs.

Feature Importance

For tree-based models:

```python
import matplotlib.pyplot as plt

importances = model.feature_importances_
plt.barh(feature_names, importances)
```

SHAP Values

SHAP provides local interpretability by attributing contributions to each feature.

```python
import shap

explainer = shap.TreeExplainer(model)
shap_values = explainer.shap_values(X_test)
shap.summary_plot(shap_values, X_test)
```

LIME

LIME builds local surrogate models for explanation.

Clinical Implications

Challenge	Ethical Risk	Example
Black-box models	Lack of transparency	Clinicians can't verify or trust decisions
Biased predictions	Exacerbating inequality	Systematic under-diagnosis of specific groups
Unclear thresholds	Unjust triage decisions	Denial of treatment based on arbitrary cutoffs
Overreliance on models	Deskilling of clinicians	Ignoring human judgment in edge cases

Summary Table

Concept	Tool / Method	Application
Bias detection	AIF360, Fairlearn	Auditing predictions
Interpretability	SHAP, LIME, Feature Importance	Explaining model decisions
Fair evaluation	Equal opportunity, parity	Ethical comparison of models

Concept	Tool / Method	Application
Clinical transparency	Explainable AI (XAI)	Building trust in model-based decisions

Final Thoughts

Metrics are not enough. The best evaluation frameworks combine **mathematical rigor** with **ethical awareness**. In healthcare, our models must not only be **accurate** — they must be **fair**, **understandable**, and **accountable**.

> A good model respects the truth.
> A great model respects the people it's meant to help.

What's Next

This is the end of the technical content — but the start of the real journey. Deploying models in medicine requires **ongoing evaluation**, **human oversight**, and **humility**.

As a final section of the book, we recommend:

- A curated list of **further readings**
- Links to datasets and open-source tools
- Code notebooks to explore real-world use cases

rubió
Metabolomics