Classification Metrics

Complete Reference Guide with MNIST Examples

Table of Contents

1. Introduction to Classification Metrics

Classification metrics are essential tools for evaluating the performance of machine learning models. They help us understand how well our model is performing and guide us in model selection and improvement. This guide covers all major classification metrics with practical examples using the MNIST dataset.

Key Concepts

Binary Classification: Distinguishing between two classes (e.g., spam vs. not spam)

Multi-class Classification: Distinguishing between multiple classes (e.g., digits 0-9 in MNIST)

Ground Truth: The actual correct labels

Predictions: What our model predicts

2. Confusion Matrix

The confusion matrix is the foundation of classification evaluation. It's a table that shows the actual vs predicted classifications.

Binary Classification Confusion Matrix

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

MNIST Example: Digit 5 vs. Not-5

Let's say we're building a binary classifier to detect digit "5" in MNIST:

- **TP (True Positive):** Model correctly identifies a "5" as "5"
- **TN (True Negative):** Model correctly identifies a "7" as "not-5"
- **FP (False Positive):** Model incorrectly identifies a "6" as "5"
- **FN (False Negative):** Model incorrectly identifies a "5" as "not-5"

		Predicted "5"	Predicted "Not-5"
Actual	Is "5"	**850 (TP)**	**150 (FN)**
	Not "5"	**200 (FP)**	**8800 (TN)**

Multi-class Confusion Matrix

For multi-class problems like MNIST (10 digits), the confusion matrix becomes larger:

	Pred 0	Pred 1	Pred 2	...	Pred 9
Actual 0	**980**	**0**	**5**	**...**	**2**
Actual 1	**1**	**1130**	**3**	**...**	**1**
Actual 2	**8**	**12**	**1000**	**...**	**5**
...	**...**	**...**	**...**	**...**	**...**
Actual 9	**5**	**8**	**2**	**...**	**995**

3. Accuracy

Accuracy is the most intuitive metric - it measures the proportion of correct predictions.

```
Accuracy = (TP + TN) / (TP + TN + FP + FN)
```

```
Accuracy = Number of Correct Predictions / Total
Number of Predictions
```

MNIST Accuracy Example

If our MNIST classifier correctly identifies 9,500 out of 10,000 test images:

```
Accuracy = 9,500 / 10,000 = 0.95 = 95%
```

Multi-class Accuracy

For multi-class problems, accuracy is calculated the same way - just count all correct predictions across all classes.

How Much Accuracy is Good?

The answer depends on your domain:

- **MNIST:** 95%+ is reasonable, 99%+ is very good
- **Medical diagnosis:** Even 95% might not be acceptable
- **Spam detection:** 90%+ is often sufficient
- **Image recognition:** Varies widely by complexity

The Problem with Accuracy

Accuracy Can Be Misleading!

Consider a dataset where 95% of emails are not spam. A classifier that always predicts "not spam" will have 95% accuracy but is completely useless!

Class Imbalance Problem: When classes are unbalanced, accuracy doesn't tell the full story.

MNIST Imbalanced Example

Imagine we have an imbalanced MNIST subset:

- 9,900 images of digits 0-8
- 100 images of digit 9

A model that never predicts "9" would still have 99% accuracy, but it would never detect the digit 9!

4. Precision

Precision answers: "Of all the positive predictions I made, how many were actually correct?"

```
Precision = TP / (TP + FP)
```

Precision focuses on the quality of positive predictions.

MNIST Precision Example

For digit "5" detection:

- Model predicted "5" for 1,050 images
- 850 were actually "5" (TP = 850)
- 200 were not "5" (FP = 200)

```
Precision = 850 / (850 + 200) = 850 / 1,050 =
0.81 = 81%
```

Interpretation: When the model predicts "5", it's correct 81% of the time.

When is Precision Important?

- **Spam detection:** You don't want to mark important emails as spam
- **Medical diagnosis:** False positives can cause unnecessary worry/treatment
- **Fraud detection:** False positives block legitimate transactions

5. Recall (Sensitivity)

Recall answers: "Of all the actual positive cases, how many did I correctly identify?"

```
Recall = TP / (TP + FN)
```

Recall focuses on finding all positive cases.

MNIST Recall Example

For digit "5" detection:

- There were 1,000 actual "5"s in the test set
- Model correctly identified 850 of them (TP = 850)
- Model missed 150 of them (FN = 150)

```
Recall = 850 / (850 + 150) = 850 / 1,000 = 0.85 = 85%
```

Interpretation: The model found 85% of all actual "5"s.

When is Recall Important?

- **Cancer screening:** Missing a cancer case is very serious
- **Security systems:** Missing a threat could be catastrophic
- **Search engines:** You want to find all relevant documents

6. F1 Score

The F1 score is the harmonic mean of precision and recall. It balances both metrics.

```
F1 = 2 × (Precision × Recall) / (Precision + Recall)
```

```
F1 = 2TP / (2TP + FP + FN)
```

MNIST F1 Score Example

Using our digit "5" example:

- Precision = 81%
- Recall = 85%

```
F1 = 2 × (0.81 × 0.85) / (0.81 + 0.85) = 2 ×
0.6885 / 1.66 = 0.83 = 83%
```

Why Use F1 Score?

- Provides a single metric that balances precision and recall
- Useful when you need both high precision and high recall
- More informative than accuracy for imbalanced datasets
- Harmonic mean penalizes extreme values (if either precision or recall is very low, F1 will be low)

F1 Score Limitations

F1 score gives equal weight to precision and recall. Sometimes you might care more about one than the other. In such cases, consider F-beta score or use precision and recall separately.

7. Type I and Type II Errors

These are classical statistical terms that map directly to our confusion matrix concepts:

Error Types

Type I Error (α): False Positive - Rejecting a true null hypothesis

Type II Error (β): False Negative - Failing to reject a false null hypothesis

In Classification Context:

- **Type I Error = False Positive (FP):** Predicting positive when it's actually negative
- **Type II Error = False Negative (FN):** Predicting negative when it's actually positive

MNIST Type I/II Error Example

For detecting digit "5":

- **Type I Error:** Classifying a "6" as a "5" (False Positive)
- **Type II Error:** Classifying a "5" as something else (False Negative)

Real-world example - Medical testing:

- **Type I Error:** Test says patient has disease when they don't (False Positive)
- **Type II Error:** Test says patient is healthy when they have disease (False Negative)

Trade-offs Between Type I and Type II Errors

There's usually a trade-off between these errors:

- Reducing Type I errors often increases Type II errors
- The cost of each error type varies by application
- Adjust classification threshold based on which error is more costly

8. Multi-class Precision and Recall

For multi-class problems like MNIST, we can calculate precision and recall in several ways:

Per-Class Metrics

Calculate precision and recall for each class individually by treating it as a binary problem (one class vs. all others).

MNIST Per-Class Example

For digit "3" in a 10-class problem:

- **TP:** Images correctly classified as "3"
- **FP:** Images incorrectly classified as "3" (actually other digits)
- **FN:** Images that are "3" but classified as other digits
- **TN:** Images correctly classified as non-"3"

Macro-averaged Metrics

Calculate metrics for each class, then take the average:

$$\text{Macro Precision} = (\text{Precision}_1 + \text{Precision}_2 + \ldots + \text{Precision}_n) / n$$

$$\text{Macro Recall} = (\text{Recall}_1 + \text{Recall}_2 + \ldots + \text{Recall}_n) / n$$

MNIST Macro-averaged Example

If we have precision for each digit:

- Digit 0: 0.98, Digit 1: 0.99, Digit 2: 0.95, ...
- Digit 9: 0.93

```
Macro Precision = (0.98 + 0.99 + 0.95 + ... +
0.93) / 10
```

Micro-averaged Metrics

Pool all true positives, false positives, and false negatives across all classes:

```
Micro Precision = Σ(TP_i) / (Σ(TP_i) + Σ(FP_i))
```

```
Micro Recall = Σ(TP_i) / (Σ(TP_i) + Σ(FN_i))
```

Weighted-averaged Metrics

Similar to macro-average, but weighted by the number of samples in each class:

```
Weighted Precision = Σ(Precision_i × Support_i) /
Total_samples
```

Which Average to Use?

- **Macro:** Treats all classes equally (good for balanced datasets)
- **Micro:** Influenced by class frequency (good for imbalanced datasets)
- **Weighted:** Accounts for class imbalance while preserving interpretability

9. Confusion Matrix for Multi-class

Complete MNIST Confusion Matrix Analysis

A typical MNIST confusion matrix might show:

```
# Sample MNIST confusion matrix (simplified) Predicted
0 1 2 3 4 5 6 7 8 9 0 [980 0 1 0 0 3 0 1 2 1] 1 [ 0
1134 2 1 0 1 4 1 2 0] A 2 [ 3 8 1000 5 2 1 4 10 5 1] c
3 [ 0 0 6 993 0 8 0 6 6 1] t 4 [ 1 0 4 0 957 0 5 1 3
11] u 5 [ 3 0 1 8 1 873 5 1 2 0] a 6 [ 6 3 0 0 3 7 937
0 2 0] l 7 [ 1 8 18 1 0 0 0 999 1 0] 8 [ 3 2 5 8 5 6 1
3 941 0] 9 [ 4 7 0 6 10 4 0 5 2 971]
```

Reading the matrix:

- Diagonal elements = correct predictions
- Off-diagonal elements = misclassifications
- Row sums = actual class counts
- Column sums = predicted class counts

Common Misclassifications in MNIST

- **4 ↔ 9:** Similar shape, especially when handwritten
- **3 ↔ 8:** Can look similar with poor handwriting
- **5 ↔ 6:** The bottom loop can be ambiguous
- **1 ↔ 7:** Depends on writing style

10. When Accuracy is Misleading

The Accuracy Paradox

High accuracy doesn't always mean a good model. Here's why:

Class Imbalance Problem

Extreme Imbalance Example

Imagine detecting a rare disease that affects 0.1% of the population:

		Predicted Healthy	Predicted Sick
Actual	Healthy	**9,990 (TN)**	**0 (FP)**
	Sick	**10 (FN)**	**0 (TP)**

A model that always predicts "healthy" has:

- **Accuracy:** 9,990/10,000 = 99.9% (Excellent!)
- **Precision:** 0/0 = Undefined
- **Recall:** 0/10 = 0% (Terrible!)
- **F1 Score:** 0% (Terrible!)

This model is useless despite 99.9% accuracy!

MNIST Imbalance Scenarios

Biased MNIST Dataset

Suppose we have a biased MNIST subset:

- 8,000 images of digit "1" (easy to recognize)
- 200 images each of digits 0, 2-9 (1,800 total)

A naive model might achieve high accuracy by:

- Always predicting "1" → 8,000/9,800 = 82% accuracy
- But 0% recall for digits 0, 2-9
- Completely fails at the intended task

11. Comprehensive MNIST Example

Real MNIST Classification Results

Scenario: CNN trained on MNIST, evaluated on 10,000 test images

```
Classification Report: precision recall f1-score
support 0 0.99 0.99 0.99 980 1 0.99 0.99 0.99 1135 2
0.98 0.97 0.97 1032 3 0.98 0.98 0.98 1010 4 0.98 0.98
0.98 982 5 0.98 0.98 0.98 892 6 0.99 0.99 0.99 958 7
0.97 0.98 0.98 1028 8 0.98 0.97 0.97 974 9 0.97 0.96
0.97 1009 accuracy 0.98 10000 macro avg 0.98 0.98 0.98
10000 weighted avg 0.98 0.98 0.98 10000
```

Analysis:

- **Overall Accuracy:** 98% - Very good for MNIST
- **Balanced Performance:** All digits have similar F1 scores (96-99%)
- **Macro vs Weighted:** Nearly identical (indicates balanced dataset)