Learning F1 Score Calculation in Python with Examples

Authored by

Mohammed looti

November 2, 2025

Introduction to F1 Score: A Crucial Classification Metric

In the field of **Machine Learning**, particularly when tackling binary or multi-class classification problems, the choice of evaluation metric is paramount. Simply relying on accuracy can be misleading, especially when dealing with datasets where the class distribution is highly imbalanced. This scenario necessitates the use of more sophisticated metrics that provide a balanced view of model performance. The **F1 Score** is one such metric, universally recognized for its ability to provide a comprehensive assessment of a classifier's effectiveness.

The **F1 Score** is defined as the harmonic mean of two fundamental metrics: **Precision** and **Recall**. By incorporating both of these aspects, the F1 Score yields a single value that represents the model's ability to correctly classify positive instances while minimizing both false positives and false negatives. A high F1 Score indicates that the model has low false positive rates (high **Precision**) and low false negative rates (high **Recall**). This balance is crucial in applications where the costs associated with missing a positive case are similar to the costs of incorrectly flagging a negative case as positive.

Unlike the arithmetic mean, the harmonic mean used in the **F1 Score** calculation is conservative; it heavily penalizes models where there is a large disparity between **Precision** and **Recall**. If a model excels at one metric but fails miserably at the other, the resulting F1 Score will be low, accurately reflecting the model's unreliability. Consequently, mastering the calculation and interpretation of the **F1 Score** is an essential skill for developing robust and reliable predictive systems in data science.

Understanding Precision and Recall: The Foundation of F1 Score

The calculation of the **F1 Score** is entirely dependent on the outcomes reported in the **Confusion Matrix**. This matrix summarizes the four possible outcomes of a classification prediction: True Positives (TP), False Positives (FP), True Negatives (TN), and False Negatives (FN). Based on these counts, we can derive **Precision** and **Recall**, which offer distinct perspectives on model performance.

Precision, sometimes referred to as the Positive Predictive Value, focuses on the accuracy of the positive predictions made by the model. It answers the question: "When the model predicts an instance belongs to the positive class, what is the probability that it is actually correct?" A high **Precision** score is desirable in scenarios where minimizing false alarms is critical, such as spam detection or identifying fraudulent transactions. The mathematical definition is:

Precision = True Positives / (True Positives + False Positives)

Conversely, **Recall**, also known as Sensitivity or True Positive Rate, focuses on the model's

coverage of the actual positive cases. It answers the question: "Of all the instances that truly belong to the positive class, what fraction did the model successfully identify?" High **Recall** is paramount in fields like medical diagnosis, where the failure to identify a positive case (a false negative) can have severe consequences. The formula for **Recall** is:

Recall = True Positives / (True Positives + False Negatives)

The challenge in classification modeling often lies in the inherent trade-off between **Precision** and **Recall**. Adjusting the classification threshold to increase one metric often leads to a decrease in the other. The **F1 Score** provides a compromise by combining these two metrics into a single weighted measure using the harmonic mean. This ensures that a model must perform well on both identification accuracy (Precision) and completeness (Recall) to achieve a high overall score:

F1 Score = 2 * (Precision * Recall) / (Precision + Recall)

Case Study: Calculating F1 Score Using a Classification Example

To solidify our understanding, let us examine a concrete example. Imagine we are utilizing a **Logistic Regression** model tasked with predicting whether 400 college basketball players will be drafted into the NBA. In this context, being drafted is the positive outcome (Class 1).

The resulting predictions from the model are summarized below in the **Confusion Matrix**, which details how the 400 observations were classified:

		Predicted	
		Drafted = Yes	**Drafted = No**
Actual	**Drafted = Yes**	**120** (True Positive)	**40** (False Negative)
	Drafted = No	**70** (False positive)	**170** (True Negative)

Based on the data provided in the matrix, we extract the necessary counts:

True Positives (TP): 120 (Players correctly predicted as drafted)
False Positives (FP): 70 (Players incorrectly predicted as drafted)
False Negatives (FN): 40 (Players who were drafted but missed by the model)
True Negatives (TN): 170 (Players correctly predicted as not drafted)

The total actual positive count is 160 (120 TP + 40 FN), and the total predicted positive count is 190 (120 TP + 70 FP). These figures allow us to calculate the specific metrics for this predictive

model. We begin by calculating **Precision**:

Precision = True Positive / (True Positive + False Positive) = 120 / (120 + 70) = 120 / 190 = **0.63157**

This result shows that roughly 63.16% of the players the model identified as drafted were, in fact, drafted. Next, we calculate **Recall**, which assesses how many of the truly draft-worthy players the model successfully captured:

Recall = True Positive / (True Positive + False Negative) = 120 / (120 + 40) = 120 / 160 = **0.75**

The model identified 75% of all players who were actually drafted. Finally, we calculate the **F1 Score**, integrating both of these results:

F1 Score = 2 * (0.63157 * 0.75) / (0.63157 + 0.75) = 2 * (0.4736775) / (1.38157) = 0.947355 / 1.38157 = **0.6857**

The resulting **F1 Score** of 0.6857 provides a clear, single measure of the model's overall efficacy, highlighting the modest performance resulting from the lower **Precision** value.

Implementing the F1 Score Calculation in Python

In a real-world data science workflow, manual calculations are replaced by efficient library functions. The **scikit-learn** (**sklearn**) package provides a comprehensive set of machine learning tools, including optimized functions for metric calculation. The `f1_score()` function allows data scientists to compute the F1 Score directly from the arrays of actual and predicted values.

We can use the exact parameters from our case study to verify the manual result using Python. We first need to define the `actual` array (160 ones and 240 zeros) and the `pred` array, which reflects the confusion matrix counts (120 TP, 40 FN, 70 FP, 170 TN). NumPy's `repeat` function is ideal for constructing these arrays quickly:

```
import numpy as np
from sklearn.metrics import f1_score

#define array of actual classes
actual = np.repeat(, repeats=)

#define array of predicted classes
pred = np.repeat(, repeats=)

#calculate F1 score
f1_score(actual, pred)
```

0.6857142857142857

The output confirms the result obtained earlier: **0.6857**. Utilizing the `f1_score()` function simplifies the evaluation process considerably and minimizes potential calculation errors. Furthermore, for more complex scenarios involving multi-class classification, the **scikit-learn** function allows for various averaging techniques (e.g., micro, macro, weighted) to adapt the F1 Score calculation to different problem requirements.

Interpreting and Applying the F1 Score for Model Selection

The primary advantage of the **F1 Score** is its use as a decisive metric for model selection. When a data scientist is tasked with optimizing a classification model, they often compare several candidate algorithms or hyperparameter configurations. The model that achieves the highest **F1 Score** is usually the preferred choice, assuming the goal is balanced performance between **Precision** and **Recall**.

Consider the scenario where we test two different models on the NBA draft data. Model A achieved an F1 Score of 0.6857, as calculated above. If Model B, perhaps a Random Forest algorithm, is tested and yields an **F1 Score** of 0.75, then Model B would be definitively selected as the superior classifier. The higher score signifies that Model B is more effective at correctly identifying the positive class while maintaining stricter control over false predictions.

It is crucial, however, to contextualize the F1 Score within the specific domain requirements. While the F1 Score is excellent for balanced evaluation, situations demanding extreme emphasis on one metric over the other may require different optimization strategies. For example, if the cost of a false positive is significantly higher than a false negative (e.g., triggering an unnecessary emergency response), a model might be selected specifically for its higher **Precision**, even if its **F1 Score** is slightly lower than a competitor's. Nevertheless, the F1 Score remains the standard benchmark when the costs of Type I and Type II errors are relatively equivalent, ensuring the deployed model is robust on both fronts.

Additional Resources for Classification Metrics

For those interested in delving deeper into evaluation metrics and classification modeling techniques within the Python ecosystem, the following resources are highly recommended:

How to Calculate Balanced Accuracy in Python
A Comprehensive Guide to Classification Report Metrics
Exploring the Trade-offs: ROC Curves and AUC for Imbalanced Data