Calculate Matthews Correlation Coefficient in Python

Authored by

Mohammed looti

November 2, 2025

The Matthews correlation coefficient (MCC) (1/5) is an essential performance metric used to evaluate the quality of a classification model (1/5). Unlike simpler metrics like accuracy or F1 score, MCC is considered one of the most reliable measures for binary classification tasks, especially when dealing with skewed class distributions.

Understanding the Matthews Correlation Coefficient (MCC)

The primary purpose of the Matthews correlation coefficient (MCC) (2/5) is to provide a balanced assessment of a model's predictive capability. It effectively summarizes the entire Confusion Matrix (1/5) into a single value, accounting for the ratio of all four categories: true positives, true negatives, false positives, and false negatives. This comprehensive approach ensures that the metric is robust against class imbalance.

A high MCC value indicates that the classifier performs well across all categories, not just the majority class. This makes it particularly valuable in fields where misclassification of the minority class carries significant weight, such as medical diagnostics or fraud detection. By maximizing MCC, we ensure the model exhibits good correlation between the predicted labels and the ground truth labels.

Why MCC is Superior for Imbalanced Datasets

In practical machine learning applications, we often encounter imbalanced datasets (1/5), where one class heavily outweighs the other. In such scenarios, simple accuracy can be highly misleading. For instance, if 95% of data points belong to Class A, a model that simply predicts Class A every time will achieve 95% accuracy, yet it is utterly useless.

The Matthews correlation coefficient (MCC) (3/5) overcomes this limitation because it is fundamentally a correlation measure between the actual and predicted binary classifications. It is calculated such that it yields a high score only if the model performs well on both the positive and negative classes, regardless of their relative size. This intrinsic characteristic makes MCC the preferred metric when working with imbalanced datasets (2/5) or when the costs of false positives and false negatives are significantly different.

Deconstructing the MCC Formula and Components

The mathematical definition of MCC ensures its balance and reliability. It is calculated as follows:

MCC = (TP*TN - FP*FN) / √(TP+FP)(TP+FN)(TN+FP)(TN+FN)

The numerator (TP*TN - FP*FN) captures the difference between correct predictions and erroneous predictions. The denominator serves as a normalization factor, ensuring the resulting

value falls within a fixed range.

The components used in this formula are derived directly from the Confusion Matrix (2/5):

TP: Number of **True Positives** (correctly predicted positive instances).
TN: Number of **True Negatives** (correctly predicted negative instances).
FP: Number of **False Positives** (negative instances incorrectly classified as positive).
FN: Number of **False Negatives** (positive instances incorrectly classified as negative).

The resulting value for MCC ranges from -1 to 1, providing clear interpretability:

-1 indicates total disagreement or inverse prediction--the model consistently predicts the opposite of the actual class.
0 is synonymous with completely random guessing--the model performs no better than chance.
1 indicates total agreement--the model achieves perfect prediction between predicted classes and actual classes.

Practical Example: Predicting NBA Draft Outcomes

To illustrate the calculation of MCC, consider a scenario where a sports analyst uses a classification model (2/5) to predict whether 400 different college basketball players will be drafted into the NBA. In this dataset, being drafted (Positive class) is a relatively rare event, making this a classic example of an imbalanced classification problem.

The analyst runs the model, and the predictions are summarized in the following Confusion Matrix (3/5):

		Predicted	
		Drafted = Yes	**Drafted = No**
Actual	**Drafted = Yes**	**15** (True Positive)	**5** (False Negative)
	Drafted = No	**5** (False positive)	**375** (True Negative)

From the matrix, we extract the four essential values needed for the MCC calculation:

True Positives (TP): 15 (Correctly predicted drafted)
True Negatives (TN): 375 (Correctly predicted undrafted)
False Positives (FP): 5 (Incorrectly predicted drafted)
False Negatives (FN): 5 (Incorrectly predicted undrafted)

Applying these values to the Matthews correlation coefficient (MCC) (4/5) formula allows us to quantify the model's performance:

First, calculate the numerator and the terms in the denominator:

Numerator (TP*TN - FP*FN) = (15 * 375) - (5 * 5) = 5625 - 25 = 5600
Denominator Terms:
TP+FP = 15 + 5 = 20
TP+FN = 15 + 5 = 20
TN+FP = 375 + 5 = 380
TN+FN = 375 + 5 = 380

Next, substitute these values back into the MCC equation:

MCC = 5600 / √(20)(20)(380)(380)
MCC = 5600 / √14440000
MCC = 5600 / 3800
MCC ≈ 0.7368

The resulting Matthews correlation coefficient is **0.7368**. Since this value is significantly closer to 1 than to 0, it strongly suggests that the model exhibits a decent and reliable performance in predicting whether or not players will be drafted, effectively handling the inherent class imbalance of the scenario.

Implementing MCC in Python using Scikit-learn

While manual calculation is helpful for understanding the underlying mathematics, leveraging powerful data science libraries is standard practice in Python. The scikit-learn (1/5) library provides an efficient, optimized function specifically designed for this metric. The function, appropriately named **matthews_corrcoef()**, resides within the sklearn (2/5).metrics module.

The following example demonstrates how to define the array of predicted classes and the array of actual classes corresponding to our NBA draft scenario, and then calculate the Matthews correlation coefficient using the built-in function:

import numpy as np
from sklearn.metrics import matthews_corrcoef

# Define array of actual classes (20 drafted , 380 undrafted)
actual = np.repeat(, repeats=)

Define array of predicted classes based on Confusion Matrix results:

```
# TP=15, FN=5 (Drafted, but 5 predicted Undrafted)
# FP=5, TN=375 (Undrafted, but 5 predicted Drafted)
pred = np.repeat(, repeats=)

# Calculate Matthews correlation coefficient
matthews_corrcoef(actual, pred)
```

```
0.7368421052631579
```

As expected, the computed MCC value of **0.7368** precisely matches the result derived through the manual calculation, confirming the accuracy and utility of the **matthews_corrcoef()** function provided by scikit-learn (3/5).

Interpreting the Result and Best Practices

A score of 0.7368 suggests a strong positive correlation between the model's predictions and the true outcomes. This is a robust indicator that the model is making accurate predictions across both the minority class (drafted players) and the majority class (undrafted players). If the model had merely prioritized high accuracy by predicting 'Undrafted' for most players, the MCC score would have been significantly lower, close to zero.

In practice, the Matthews correlation coefficient (MCC) (5/5) should be prioritized as the primary performance metric whenever you are building a classification model (3/5) where class distributions are uneven or where misclassification costs are not symmetrical. Relying solely on metrics like precision, recall, or F1 score can sometimes obscure underlying issues in how the model handles true negatives, which MCC incorporates seamlessly.

Note: You can find the complete documentation for the **matthews_corrcoef()** function here.

Additional Resources for Classification Metrics

Understanding MCC is just one step in mastering model evaluation. The following tutorials explain how to calculate other common metrics for classification models in Python, which often complement the insights gained from analyzing the Confusion Matrix (4/5):