Understanding and Calculating Symmetric Mean Absolute Percentage Error (SMAPE) with Python

Authored by

Mohammed looti

November 7, 2025

Evaluating the **performance of predictive models** is a core discipline within data science and forecasting. While numerous metrics exist, the **Symmetric Mean Absolute Percentage Error (SMAPE)** has gained significant traction as a robust and reliable measure. SMAPE is particularly valuable in complex scenarios where data scale varies widely or when dealing with instances of zero values, a frequent challenge in intermittent demand forecasting. Unlike the traditional Mean Absolute Percentage Error (MAPE), SMAPE provides a more balanced assessment by normalizing the absolute error against the average of the actual and forecasted values. This normalization effectively mitigates the inherent bias present in MAPE, which tends to inflate errors when actual values are close to zero. This comprehensive tutorial will provide an in-depth exploration of the SMAPE metric and detail the precise methodology required for its calculation using the **Python** programming language, leveraging the high-performance capabilities of the **NumPy** library for efficient array manipulation.

Defining the SMAPE Formula

The mathematical definition of the **SMAPE** serves as the essential foundation for its implementation in code. A thorough understanding of each component of this equation is critical for accurate interpretation and proper application. The definition is specifically designed to ensure the metric is symmetric; this means the measured error remains balanced, regardless of whether the forecast overshoots or undershoots the actual value. This symmetry is a crucial feature for ensuring reliable evaluation across diverse datasets and modeling contexts. Since standard **Python** libraries do not typically include a native SMAPE function, we must first define this formula clearly before translating it directly into a custom, high-performance function.

The calculation of SMAPE, commonly scaled to range from 0% to 100% for practical use in industry, is defined by the following expression:

SMAPE = (1/n) * Σ(|forecast - actual| / ((|actual| + |forecast|)/2)) * 100

To implement this formula correctly, it is necessary to identify the role of each key variable:

Σ - Represents the mathematical operation of "summation," indicating that the calculation must be performed iteratively across every single data point in the sample.
n - Denotes the total **sample size**, which corresponds to the number of paired actual and forecasted values used in the evaluation.
actual - Refers to the true, observed data value recorded at a specific time or observation point.
forecast - Refers to the predicted or estimated data value generated by the **forecasting model** for that exact observation point.

Why SMAPE Offers Superior Forecasting Assessment

The primary advantage of the **SMAPE** is its ability to deliver an error measurement expressed as a percentage. This format is inherently intuitive and allows for seamless comparison across entirely different forecasting problems, even if the underlying data magnitudes vary drastically. Metrics such as Mean Squared Error (MSE) or Root Mean Squared Error (RMSE) produce results in the original units of the data, which makes cross-comparison difficult. SMAPE, conversely, provides a normalized, unit-free metric that focuses solely on proportional error.

Crucially, the symmetry introduced in SMAPE--by dividing the absolute error by the average magnitude of the data points involved (the average of absolute actual and absolute forecast)--ensures fairness. This design dictates that the penalty applied to a scenario where the model over-predicts by a certain percentage is mathematically equivalent to the penalty applied when it under-predicts by the same relative percentage. This differs markedly from traditional percentage errors like MAPE, which are prone to severe drawbacks, particularly when the actual value approaches zero. If the actual value is zero, MAPE becomes mathematically undefined, a common, problematic scenario in real-world applications like intermittent demand forecasting.

SMAPE elegantly resolves the zero-value problem. By including the absolute forecast value in the denominator (specifically, using the sum of absolute actual and absolute forecast), the denominator remains non-zero unless both the actual and the forecast are simultaneously zero. In the rare event that both are zero, the error is typically and logically defined as zero. This robust handling of edge cases maintains the integrity and reliability of the metric, making it highly suitable for complex, real-world data science challenges. Fundamentally, SMAPE calculates the average absolute difference between the prediction and the actual, scaled by the average magnitude of the data, ensuring the resulting percentage error is less sensitive to outliers compared to simpler metrics.

Implementing a Vectorized SMAPE Function in Python

Given that the standard **Python** statistical ecosystem, including core packages like SciPy, generally lacks a built-in function for SMAPE, developing a custom, efficient function is the industry standard. This implementation hinges upon the use of the **NumPy** library, which is indispensable for executing high-performance numerical operations. NumPy's powerful array capabilities enable vectorized operations, allowing us to compute the complex mathematical calculations defined by the SMAPE formula across entire arrays of actual and forecasted values simultaneously. This vectorized approach is vastly superior to relying on slow, explicit Python loops, offering critical efficiency when dealing with the large datasets typical of modern forecasting projects.

The custom function must be designed to accept two primary inputs: the array of actual values (often symbolized as 'a') and the array of forecasted values (labeled 'f'). The process involves a

direct translation of the summation formula into sequential, vectorized steps. First, the absolute difference (numerator) is calculated. Second, the average magnitude of the data points (denominator) is determined. Finally, these results are combined, summed, and scaled by the necessary constants (2 and 100) to yield the final percentage result.

The following code snippet defines an efficient, highly readable, and fully vectorized SMAPE function utilizing **NumPy**. Observe how the function's structure precisely mirrors the mathematical definition, guaranteeing both accuracy and computational speed:

```
import numpy as np

def smape(a, f):
return 1/len(a) * np.sum(2 * np.abs(f-a) / (np.abs(a) + np.abs(f))*100)
```

Note the critical factor of 2 within the numerator of the **Python** code. Mathematically, this is equivalent to the multiplication by 1/2 that appears in the denominator of the original SMAPE formula; it is strategically moved here for superior computational efficiency and cleaner code architecture. The final multiplication by 100 ensures the resulting ratio is presented as a percentage, which is the universally accepted format for interpreting forecasting accuracy. Once this function is integrated into your environment, it provides a seamless tool for evaluating the performance of any forecasting model, regardless of data scale.

Practical Application with Sample Forecast Data

With the SMAPE calculation function successfully defined, the next logical step involves applying it to a realistic scenario comprising observed ground truth data and corresponding model forecasts. In professional data analysis, these data points are typically managed using high-performance structures like **NumPy** arrays or Pandas objects. For this demonstration, we will define two distinct arrays: `actual`, representing the true observations, and `forecast`, representing the output of a hypothetical **forecasting model**. It is absolutely essential that both arrays possess the same length, maintaining a consistent **sample size** (n) to ensure a precise one-to-one pairing required for calculating the error at every observation point.

We present a small, illustrative dataset below. This example highlights how the metric responds to varying degrees of prediction error. For instance, the initial observation pair (actual 12, forecast 11) results in a small absolute error of 1. In contrast, the final data point (actual 27, forecast 18) demonstrates a much larger absolute difference, and the overall SMAPE calculation will naturally reflect this greater proportional deviation within the final mean.

Executing the following **Python** code snippet will perform three actions: first, it imports the necessary library; second, it defines the data vectors; and finally, it invokes the custom `smape`

function we created earlier. This operation delivers the averaged percentage error across the entire forecasting period in a single, concise output:

#define arrays of actual and forecasted data values using NumPy arrays
actual = np.array()
forecast = np.array()

#calculate SMAPE by calling the defined function
smape(actual, forecast)

12.45302

Interpreting the Result and Understanding Context

The resulting scalar value from the execution, `12.45302`, indicates that the symmetric mean absolute percentage error for this particular **forecasting model** is approximately **12.45302%**. Interpreting this percentage requires critical contextualization, as acceptability criteria vary widely across different industries and business objectives. A 12% error might be considered excellent accuracy in predicting highly volatile variables like cryptocurrency prices, but it could be deemed unacceptable in scenarios demanding high precision, such as complex supply chain or inventory management systems. A foundational principle remains: a lower SMAPE value always signifies superior accuracy, with 0% representing a perfect alignment between predictions and observations.

It is essential to understand why this symmetric result is more reliable than a non-symmetric MAPE score would be. If the true values were consistently near zero, the corresponding MAPE score would be severely distorted or undefined. By normalizing the error using the average of the actual and forecast in the denominator, the 12.45% figure provides a much fairer and more balanced representation of the model's average proportional accuracy. This inherent stability makes **SMAPE** an invaluable tool for comparing the effectiveness of different models across multiple time series that may exhibit distinct volatility profiles or magnitude shifts.

To gain a complete picture, practitioners often perform error decomposition. By examining the largest individual error in our sample (actual 27, forecast 18), we see its symmetric percentage error is calculated as 40%. This single, substantial deviation significantly raises the overall mean error of 12.45%. This illustrates a key point: while **SMAPE** provides a crucial, aggregated summary statistic, it should always be complemented by residual analysis and visualization tools. Relying solely on the mean metric can obscure specific periods or data points where the model struggled catastrophically, emphasizing the need for a holistic approach to model performance monitoring.

Key Considerations and Known Limitations of SMAPE

Despite the considerable advantages the **Symmetric Mean Absolute Percentage Error** holds over older relative error metrics, it is not without its own specific caveats. One notable critique relates to the handling of the edge case where both the actual and forecast values are exactly zero. As previously mentioned, the common convention defines this error as zero, which is mathematically robust but can potentially mask instances where the model correctly predicting zero might be a complex achievement given the underlying data generation process. Furthermore, although SMAPE is defined as symmetric, some quantitative researchers argue that its definition (scaling error based on the average of the two values) still introduces a subtle implicit bias compared to metrics based purely on the absolute error, especially when forecasts consistently deviate significantly from the truth.

A critical practical consideration for data professionals is SMAPE's behavior when encountering extremely large numerical values (outliers). Since the metric scales the error relative to the data's magnitude, large, anomalous observations can disproportionately influence the overall error calculation. This means that a minor proportional error on a massive data point might overshadow a highly significant proportional error on a smaller, but more frequent, data point within the time series. This challenge is inherent to most relative error measures, underlining the necessity for rigorous data preprocessing, including thorough outlier detection and potential data transformation, before relying on **SMAPE** to accurately assess the model's overall utility.

Ultimately, the selection of the most suitable error metric--whether it is SMAPE, RMSE, or another--must be driven by the specific business objectives and the unique statistical characteristics of the data under analysis. If the primary goal is to minimize relative errors and deliver a result easily digestible by business stakeholders, SMAPE is an excellent choice due to its percentage format and robustness against zero values. However, if the business priority is to heavily penalize large, catastrophic errors (e.g., in systems where prediction errors lead to massive financial losses), metrics like RMSE, which exponentially amplify larger error terms through squaring, might be more appropriate. A proficient data scientist must always employ a portfolio of metrics, recognizing that **SMAPE** serves as a highly valuable, but non-exhaustive, measure of forecasting accuracy.

Resources for Further Learning

To deepen your understanding of forecasting metrics, model evaluation techniques, and advanced implementation methods in **Python**, the following resources are recommended. These links offer academic context, practical discussions, and complementary methods for evaluating predictive accuracy, enabling you to build a comprehensive toolkit for time series analysis.

The following external links provide detailed information and comparative analysis of SMAPE and related error measurements:

Wikipedia Entry for SMAPE
Rob J. Hyndman's thoughts on SMAPE
Detailed explanation of the Mean Absolute Percentage Error (MAPE)

For those interested in implementing alternative percentage error metrics in Python, the following tutorials offer practical guidance:

How to Calculate MAPE in Python
How to Calculate MAPE in R
How to Calculate MAPE in Excel