How to Easily Calculate AUC (Area Under the Curve) in Python with Scikit-Learn

Authored by

stats writer

December 3, 2025

The Area Under the Curve (AUC) is a powerful and widely adopted metric for assessing the performance of classification models, particularly in scenarios involving **imbalanced datasets** or varying classification thresholds. When working in Python, calculating the AUC is straightforward thanks to the robust functionalities provided by the Scikit-Learn library.

Specifically, Scikit-Learn offers the `metrics.roc_auc_score()` function. This function requires two key inputs: the true labels (`y_test`) and the predicted probabilities of the positive class (`y_pred_proba`). The output is a single numerical value representing the AUC score, which quantifies the model's ability to distinguish between positive and negative classes within a binary classification context. This metric serves as a crucial benchmark for evaluating model quality and selecting optimal models for deployment.

Understanding Logistic Regression and Binary Classification

Before diving into the calculation of AUC, it is essential to establish the context of the model we are evaluating. Logistic Regression is a fundamental statistical algorithm utilized when the dependent or response variable is categorical, typically binary (e.g., 0 or 1, Default or Non-Default). Unlike linear regression, it models the probability of a specific outcome occurring using the logistic function, ensuring that its output is always bound between 0 and 1.

The primary goal of fitting a Logistic Regression model is to accurately predict the probability of the positive class based on a set of independent predictor variables. Because the model outputs probabilities rather than hard classifications, we require specialized metrics that can effectively evaluate how well these probabilities align with the true outcomes across various potential decision thresholds. This reliance on probability scores is precisely why the ROC curve and AUC score become indispensable tools for assessment.

Evaluating Classification Model Performance

To properly assess the quality and discriminatory power of a classification model, particularly one designed for binary classification tasks, we traditionally examine performance metrics derived from the confusion matrix. Two highly influential metrics that form the foundation of the ROC curve analysis are Sensitivity and Specificity. These metrics provide insight into how well the model handles both positive and negative cases across varying thresholds.

The trade-off between identifying true positives and avoiding false positives is central to model optimization. A model that is highly sensitive might correctly identify most positive cases but might also incorrectly flag many negative cases as positive, leading to poor specificity. Understanding this delicate balance is critical for selecting an appropriate decision threshold tailored to the specific business or scientific application.

We assess model fit by examining the following two key ratios:

Sensitivity (True Positive Rate): This is the probability that the model correctly predicts a positive outcome for an observation when the true outcome is indeed positive. High sensitivity is vital in applications where missing a positive case (a False Negative) is costly or dangerous, such as disease detection.
Specificity (True Negative Rate): This represents the probability that the model correctly predicts a negative outcome for an observation when the true outcome is negative. High specificity is crucial when incorrectly flagging a negative case as positive (a False Positive) carries significant penalties.

Visualizing Model Discrimination: The ROC Curve

One powerful way to visualize the inherent trade-off between Sensitivity and Specificity across all possible classification thresholds is by constructing the **ROC curve**, which stands for "Receiver Operating Characteristic" curve. This visualization provides a comprehensive view of the diagnostic ability of a binary classifier system as its discrimination threshold is varied from 0 to 1.

The ROC curve plots the True Positive Rate (Sensitivity) along the y-axis against the False Positive Rate (FPR), which is calculated as (1 - Specificity), along the x-axis. A perfect classifier would hug the top-left corner of the plot (high sensitivity, low FPR), whereas a diagonal line from coordinate (0,0) to (1,1) represents a model that performs no better than random guessing.

Quantifying Performance: The Area Under the Curve (AUC)

While the ROC curve provides a visual assessment, we need a single, quantifiable metric to summarize the model's overall performance across all possible thresholds. This metric is the **AUC**, which literally stands for "area under curve." The AUC score is equivalent to the probability that a randomly chosen positive instance is ranked higher (given a higher predicted probability) than a randomly chosen negative instance.

The AUC score ranges strictly from 0 to 1. An AUC of 1.0 indicates a perfect classifier that can perfectly separate positive and negative classes. Conversely, an AUC of 0.5 suggests that the model's performance is indistinguishable from random chance. Therefore, a higher AUC value signifies a better performing model with superior discriminatory power. The following section provides a detailed, step-by-step example showing how to calculate this critical metric for a Logistic Regression model implemented using Python's Scikit-Learn library.

Step 1: Setting Up the Python Environment and Dependencies

The first crucial step in any machine learning workflow is importing the necessary Python libraries

that facilitate data manipulation, model building, and metric calculation. We rely heavily on the powerful data science ecosystem provided by the **Scikit-Learn** library, alongside Pandas and NumPy for efficient data handling.

These packages allow us to load data, manage arrays, split our dataset into training and testing partitions using `train_test_split`, instantiate the linear model, and finally, utilize the dedicated metrics module for calculating the AUC score. Ensuring all packages are correctly imported before starting the computation is paramount to a successful and reproducible execution.

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn import metrics
```

Step 2: Data Preparation and Model Training

Once the environment is set, the next phase involves loading the dataset and preparing it for model training. For this demonstration, we are utilizing a sample dataset pertaining to customer default prediction, hosted remotely via a CSV file on Github. Pandas is used to quickly read this data into a DataFrame structure, enabling easy feature selection and manipulation.

We explicitly define our predictor variables (features, X), which include 'student', 'balance', and 'income', and our response variable (target, y), which is the 'default' status. A critical step for robust model evaluation is splitting the dataset into distinct training and testing subsets. We allocate 70% of the data for training the model and reserve 30% for unbiased evaluation of its performance, ensuring the final AUC score reflects how the model generalizes to unseen data.

Finally, we instantiate the **LogisticRegression** object and fit it using only the training data (`X_train` and `y_train`). This training process determines the optimal coefficients that minimize the error in predicting the probability of default based on the input features.

```
#import dataset from CSV file on Github
url = "https://raw.githubusercontent.com/arabpsychology/Python-Guides/main/default.csv"
data = pd.read_csv(url)

#define the predictor variables and the response variable
X = data]
y = data

#split the dataset into training (70%) and testing (30%) sets
```

```
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.3,random_state=0)

#instantiate the model
log_regression = LogisticRegression()

#fit the model using the training data
log_regression.fit(X_train,y_train)
```

Step 3: Calculating the AUC Score

To calculate the AUC, we must first obtain the predicted probabilities for the test set observations, not the final binary predictions (0 or 1). This is crucial because the AUC evaluates performance across all possible thresholds, which requires the underlying probability scores. We achieve this using the `predict_proba()` method on our fitted model.

The `predict_proba()` method returns an array where each row contains two probabilities: the probability of class 0 and the probability of class 1 (the positive class). For robust AUC calculation, we strictly need the probabilities associated with the positive class, which corresponds to the second column (index 1) of the output array, denoted by ``.

Finally, the `metrics.roc_auc_score()` function takes the true labels (`y_test`) and our predicted probabilities of the positive class (`y_pred_proba`) as arguments, delivering the final AUC metric.

```
#use model to predict probability that given y value is 1
y_pred_proba = log_regression.predict_proba(X_test)

#calculate AUC of model
auc = metrics.roc_auc_score(y_test, y_pred_proba)

#print AUC score
print(auc)
```

0.5602104030579559

Interpreting the AUC Result

The resulting AUC score of **0.5602** provides a definitive measure of our model's performance on the test data. To understand this result, we must benchmark it against the two extremes of the scale. As previously noted, an AUC of **0.5** signifies a model that is no better than randomly guessing the outcome--it has zero discriminatory power. On the opposite end, an AUC of 1.0 represents perfect discrimination.

While 0.5602 is marginally better than random guessing, it is generally considered a **poor score** for predictive modeling in most practical contexts. A robust classifier aiming for real-world application typically targets an AUC score significantly higher, often above 0.75 or 0.8, depending on the domain and task difficulty. This specific result indicates that the current Logistic Regression model, using only these three features, struggles to reliably distinguish between customers who will default and those who will not, suggesting a need for model refinement or better feature engineering.

For those seeking deeper theoretical understanding, the relationship between the ROC curve and the AUC is foundational to model evaluation. The AUC summarizes the entire curve, offering a single metric that represents the classifier's performance across all possible thresholds, making it a preferred evaluation metric over simpler measures like accuracy, especially in binary classification scenarios where class imbalance is present.

The following tutorials offer additional information about ROC curves and AUC scores: