Learn How to Calculate Adjusted R-Squared in Python for Model Evaluation

Authored by

Mohammed looti

November 6, 2025

Evaluating the efficacy of a predictive model is fundamental to data science and statistical inference. At the core of this evaluation, particularly within the domain of **linear regression model** development, is the assessment of fit. The most widely recognized metric for this purpose is the **R-squared** (R2), also formally known as the Coefficient of Determination. This metric provides a simple measure of how well the predictor variables account for the variance observed in the response variable.

The standard R-squared value is easily interpreted, ranging from 0 to 1. A value approaching 0 suggests that the independent variables fail to explain the variability of the response variable, indicating a poor model fit. Conversely, a value close to 1 implies that the response variable is almost perfectly predicted by the independent variables, signifying an excellent fit. While intuitively appealing, reliance solely on R-squared can lead to significant modeling errors, particularly when comparing models of varying complexity.

The Limitations of Standard R-Squared

Despite its popularity, the standard R-squared metric harbors a critical structural flaw: it is inherently biased toward complexity. This means that R2 will inevitably increase every time a new predictor variable is introduced into the model, even if that variable offers no genuine statistical contribution or explanatory power. This phenomenon occurs because the addition of any variable--even random noise--can only decrease the sum of squared residuals, thereby inflating the R-squared value.

This inherent flaw poses a significant challenge in model building. If a practitioner selects a model based purely on the highest R-squared value, they risk overfitting the data. An overfit model captures noise and random fluctuations in the training data rather than the underlying signal, rendering the model useless when applied to new, unseen data. Consequently, R-squared is a highly unreliable metric for performing rigorous model comparison or for distinguishing between a robust model and one that is unnecessarily complex.

To address this fundamental limitation, statisticians developed an adjusted metric that introduces a necessary penalty for complexity. This adjusted approach ensures that model improvement is only recognized if the newly added variables contribute significantly more explanatory power than the penalty imposed for their inclusion.

Defining and Understanding Adjusted R-Squared

The solution to the R-squared bias is the **adjusted R-squared**. This refined metric modifies the standard R-squared calculation by factoring in the number of predictors (k) used in the model relative to the total number of observations (n). By introducing this adjustment, the metric effectively penalizes the model for including variables that do not contribute meaningfully to the

overall explanatory power.

The primary utility of the adjusted R-squared lies in its role in comprehensive model selection. When a new predictor variable is introduced, the adjusted R-squared value will only increase if the gain in R-squared (i.e., the reduction in residual variance) is substantial enough to outweigh the penalty imposed by adding another degree of freedom. If the new variable is irrelevant or insignificant, the penalty will cause the adjusted R-squared to decrease, signaling that the added complexity is not justified.

For this reason, adjusted R-squared is widely considered a superior diagnostic tool for building parsimonious and robust models. It helps analysts strike a critical balance between achieving a good fit and maintaining model simplicity, ensuring that every variable included is necessary and statistically relevant.

The Mathematical Foundation of the Adjusted R-Squared Formula

Understanding the mathematical formulation is crucial for appreciating how the adjusted R-squared applies its penalty. The formula leverages the standard R2 value alongside the concept of degrees of freedom associated with the error and the total variance of the response variable.

The calculation for the **adjusted R-squared** is presented as follows, explicitly incorporating the number of observations (n) and the count of predictor variables (k):

Adjusted R2 = 1 -

The mechanism of the penalty is contained within the adjustment factor: (n-1) / (n-k-1). When the number of predictors (k) increases, the denominator (n-k-1) shrinks, causing the entire adjustment factor fraction to increase. Since this factor multiplies the unexplained variance (1-R2), a larger fraction leads to a greater subtraction from 1, thereby reducing the overall adjusted R2 value. This reduction only fails to occur if the standard R2 increases substantially enough to compensate for the growing adjustment factor.

The variables used in this equation are precisely defined to ensure accurate comparison across models:

R2: The standard R-squared value (Coefficient of Determination) derived from the model fit.
n: The total number of observations or data points in the analyzed dataset.
k: The number of predictor variables (regressors) incorporated into the multiple regression model.

Practical Calculation of Adjusted R-Squared in Python

When transitioning from theory to implementation, Python provides two primary, powerful libraries

for statistical modeling: **scikit-learn** (sklearn) and **statsmodels**. While both libraries are capable of fitting high-quality linear regression models, they differ significantly in their primary focus and how they handle performance metrics. Sklearn is engineered for predictive modeling and efficiency, meaning the adjusted R-squared must often be calculated manually. In contrast, statsmodels is designed for statistical inference, providing the adjusted R-squared directly within its comprehensive summary output.

We will demonstrate both methodologies using a consistent example: fitting a multiple linear regression model to predict vehicle horsepower (hp) based on four characteristics (mpg, wt, drat, qsec) from a sample dataset. This dual approach illustrates the flexibility required when working across different computational environments in regression analysis.

Method 1: Manual Calculation Using scikit-learn (sklearn)

As the industry standard for machine learning in Python, scikit-learn offers streamlined tools for fitting models via its LinearRegression class. After the model is fitted, the standard R-squared value is easily retrieved using the .score() method. However, because sklearn focuses on prediction quality rather than statistical inference, the adjusted R-squared must be computed explicitly by applying the mathematical formula previously discussed.

The following code snippet demonstrates the complete process. First, the data is loaded and the model is fitted. Subsequently, the adjusted R-squared formula is implemented directly, utilizing the model's R2 score, the length of the response variable (len(y), which is n), and the number of features (X.shape, which is k). This manual calculation ensures compliance with the statistical definition of the adjusted metric.

```
from sklearn.linear_model import LinearRegression
import pandas as pd

#define URL where dataset is located
url = "https://raw.githubusercontent.com/Statology/Python-Guides/main/mtcars.csv"

#read in data
data = pd.read_csv(url)

#fit regression model
model = LinearRegression()
X, y = data], data.hp
model.fit(X, y)

#display adjusted R-squared
1 - (1-model.score(X, y))*(len(y)-1)/(len(y)-X.shape-1)
```

0.7787005290062521

Executing this manual calculation results in an adjusted R-squared value of **0.7787**. This outcome signifies that, after accounting for the four predictor variables utilized, approximately 77.87% of the total variance observed in horsepower is explained by the regression model.

Method 2: Automated Reporting with statsmodels

When the primary objective is statistical inference--focused on understanding parameter significance, standard errors, and model fit diagnostics--the **statsmodels** library is often the preferred choice. A major benefit of using statsmodels is that it automatically computes and reports the adjusted R-squared value, along with a wealth of other critical statistical outputs, as part of its standard model summary.

It is important to note a slight methodological difference when using the Ordinary Least Squares (OLS) method within statsmodels: the constant term (the intercept) must be manually added to the predictor matrix X using sm.add_constant(X). Once the model is fitted, the resulting object holds the adjusted R-squared value, which can be accessed immediately via the .rsquared_adj attribute.

```
import statsmodels.api as sm
import pandas as pd

#define URL where dataset is located
url = "https://raw.githubusercontent.com/Statology/Python-Guides/main/mtcars.csv"

#read in data
data = pd.read_csv(url)

#fit regression model
X, y = data], data.hp
X = sm.add_constant(X)
model = sm.OLS(y, X).fit()

#display adjusted R-squared
print(model.rsquared_adj)
```

0.7787005290062521

As demonstrated by accessing the .rsquared_adj attribute, the result is precisely the same value of **0.7787**. This consistency across scikit-learn and statsmodels confirms the validity of the underlying statistical principle. While R-squared provides a straightforward measure of fit, the adjusted R-

squared is the essential metric for rigorous model comparison and evaluation, as it correctly accounts for the trade-off between explanatory power and model complexity in statistical analysis.

Additional Resources for Regression Analysis

To further refine your skills in linear modeling and interpretation, the following resources provide detailed guidance on implementing and optimizing various regression techniques in Python.

How to Perform Simple Linear Regression in Python
How to Perform Multiple Linear Regression in Python