# Get Regression Model Summary from Scikit-Learn

Authored by

**Mohammed looti**

October 30, 2025

In the realm of data science and statistical modeling, the ability to extract a comprehensive summary of a fitted regression model is essential for evaluation and inference. When working in Python, especially when utilizing powerful libraries like scikit-learn, practitioners often seek detailed reports that go beyond simple coefficients and score metrics.

However, it is crucial to understand that scikit-learn is fundamentally designed for machine learning tasks--primarily prediction and generalization--rather than statistical inference. Consequently, it does not provide many of the built-in diagnostic functions, such as standard errors, t-statistics, and associated p-values, that are standard in traditional statistical packages. This limitation often forces data analysts to look for alternative solutions when seeking a complete model overview.

If your goal is to obtain a detailed statistical summary of a regression model in Python, you essentially have two main methodological paths, each with its own advantages and drawbacks:

**1.** Utilize the limited, but efficient, functionality provided directly by scikit-learn.
**2.** Employ the specialized statistical package, statsmodels, which is purpose-built for statistical inference and detailed reporting.

The following sections will demonstrate how to implement both methods using a standard pandas DataFrame, allowing for a clear comparison of the resulting outputs.

## Preparing the Dataset for Modeling

To illustrate these methods, we will first define and prepare a sample dataset. This dataset is stored as a pandas DataFrame and includes two predictor variables (x1 and x2) and one continuous response variable (y), suitable for fitting a multiple linear regression model.

The setup involves importing the necessary libraries and constructing the DataFrame as shown below. This foundational step ensures both modeling approaches operate on identical input data.

**import pandas as pd**

#create DataFrame
df = pd.DataFrame({'x1': ,
'x2': ,
'y': })

#view first five rows of DataFrame
df.head()

x1 x2 y
0 1 1 76

1 2 3 78
2 2 3 85
3 4 5 88
4 2 2 72

## Method 1: Analyzing Regression Results with Scikit-learn

When using scikit-learn, our primary focus is on fitting the model and extracting key parameters that define the regression equation. The library's `LinearRegression` class is efficient and straightforward for this purpose.

The following code snippet demonstrates the process of initializing the linear regression object, defining the feature matrix (X) and the target vector (y), and subsequently training the model using the `fit()` method.

**from sklearn.linear_model import LinearRegression**

#initiate linear regression model
model = LinearRegression()

#define predictor and response variables
X, y = df], df.y

#fit regression model
model.fit(X, y)

Once the model is fitted, we can easily access the calculated intercept, the coefficients for each predictor variable, and the overall goodness-of-fit metric, the R-squared value, using the model's attributes and the `score()` method.

**#display regression coefficients and R-squared value of model**
**print(model.intercept_, model.coef_, model.score(X, y))**

70.4828205704 0.766742556527

The output above provides all the necessary components to formulate the fitted linear regression equation, which relates the response variable (y) to the two predictors (x1 and x2):

y = 70.48 + 5.79x1 - 1.16x2

Furthermore, the extracted R-squared value is approximately 0.7667. Interpreting this, we can

state that **76.67%** of the total variance observed in the response variable (y) can be statistically explained by the combined influence of the two predictor variables included in this regression model.

## Limitations of the Scikit-learn Approach

While the coefficients and the R-squared value are highly valuable for understanding the model's structure and explanatory power, this output falls short of what is required for rigorous statistical inference. In statistical analysis, we need to assess the reliability and significance of the model parameters.

Specifically, the scikit-learn output lacks crucial diagnostics such as the standard error for each coefficient, the associated t-statistics, and the individual p-values. Without these metrics, we cannot confidently determine whether the relationship between an individual predictor variable and the response variable is statistically significant.

Moreover, we are missing the overall F-statistic of the model, which tests the null hypothesis that all regression coefficients are simultaneously equal to zero. This overall test of significance is fundamental to determining the model's overall utility. To access these detailed inferential statistics, we must pivot to a tool designed specifically for this purpose.

## Method 2: Comprehensive Model Summary using Statsmodels

For analysts focused on statistical inference and detailed model diagnostics, the **statsmodels** package is the definitive choice in Python. Unlike scikit-learn, statsmodels is built around the traditional statistical paradigm, offering rich, detailed summary tables familiar to statisticians.

To use statsmodels, a subtle but important difference in model setup must be addressed: the inclusion of the intercept term. While scikit-learn automatically includes an intercept, statsmodels requires the user to explicitly add a constant column of ones to the predictor matrix (X) to represent the intercept term. This is achieved using the `sm.add_constant()` function.

The following code demonstrates how to use the Ordinary Least Squares (OLS) class within statsmodels to fit the identical multiple linear regression model and generate the exhaustive summary report:

**import statsmodels.api as sm**

#define response variable
y = df

#define predictor variables

```
x = df]

#add constant to predictor variables (required for intercept)
x = sm.add_constant(x)

#fit linear regression model (OLS = Ordinary Least Squares)
model = sm.OLS(y, x).fit()

#view model summary
print(model.summary())
```

OLS Regression Results

```
==============================================================================
===
Dep. Variable: y R-squared: 0.767
Model: OLS Adj. R-squared: 0.708
Method: Least Squares F-statistic: 13.15
Date: Fri, 01 Apr 2022 Prob (F-statistic): 0.00296
Time: 11:10:16 Log-Likelihood: -31.191
No. Observations: 11 AIC: 68.38
Df Residuals: 8 BIC: 69.57
Df Model: 2
Covariance Type: nonrobust
==============================================================================
===
coef std err t P>|t|
------------------------------------------------------------------------------
const 70.4828 3.749 18.803 0.000 61.839 79.127
x1 5.7945 1.132 5.120 0.001 3.185 8.404
x2 -1.1576 1.065 -1.087 0.309 -3.613 1.298
==============================================================================
===
Omnibus: 0.198 Durbin-Watson: 1.240
Prob(Omnibus): 0.906 Jarque-Bera (JB): 0.296
Skew: -0.242 Prob(JB): 0.862
Kurtosis: 2.359 Cond. No. 10.7
==============================================================================
===
```

## Interpreting the Statsmodels Output

A close examination of the statsmodels summary reveals that the core outputs--the regression coefficients (e.g., const = 70.4828, x1 = 5.7945, x2 = -1.1576) and the R-squared value (0.767)--perfectly align with the results calculated using scikit-learn. However, statsmodels delivers a wealth of other metrics vital for comprehensive statistical evaluation.

The first section of the report provides overall model fit statistics. We can immediately see the F-statistic (13.15) and its corresponding probability (Prob (F-statistic): 0.00296). Since this p-value is extremely low (less than 0.05), we reject the null hypothesis and conclude that the model, as a whole, is statistically significant in predicting the response variable. Additionally, the adjusted R-squared value (0.708) provides a more conservative measure of fit, accounting for the number of predictors used.

The middle section, the coefficient table, is most important for inference. For each predictor, we are provided the coefficient estimate, the standard error, the t-statistic, and the corresponding p-value (P>|t|). These p-values allow us to assess the individual significance of each predictor:

The p-value for x1 is 0.001. Since this is less than the standard significance level of 0.05, we conclude that x1 is a statistically significant predictor of y.
The p-value for x2 is 0.309. Since this is much greater than 0.05, we conclude that x2 is not a statistically significant predictor in the presence of x1.

Finally, the bottom section of the summary provides advanced diagnostic tests, such as the Omnibus test and the Durbin-Watson statistic, which help assess model assumptions like normality of residuals and autocorrelation. These tests are critical for validating the robustness of the fitted regression model.

## Additional Resources

For those interested in exploring linear modeling further in Python, the following tutorials explain how to perform other common regression operations:

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