

Learning Likelihood Ratio Tests: A Practical Guide in Python

Authored by
Mohammed loot

November 1, 2025

RECOMMENDED CITATION

Mohammed loot (2025). *Learning Likelihood Ratio Tests: A Practical Guide in Python*. PSYCHOLOGICAL STATISTICS. Retrieved from <https://statistics.arabpsychology.com/?p=7810>

The [Likelihood Ratio Test](#) (LRT) stands as a cornerstone method in frequentist statistics, primarily utilized for comparing the relative quality of two competing [regression models](#). The fundamental goal of the LRT is to formally assess whether the complexity introduced by a larger, more intricate model is statistically justified compared to a simpler, parsimonious alternative. This test provides a rigorous framework for model selection when data analysis involves multiple potential explanatory factors.

A crucial prerequisite for employing the LRT is that the models must be **nested models**. Nesting means that one model, referred to as the reduced model, must be a special case of the other, known as the full model. Specifically, the [nested model](#) is structurally simpler, incorporating only a subset of the independent or [predictor variables](#) included in the comprehensive full model. If the models are not nested, the LRT cannot be accurately applied.

To illustrate the concept of nesting, let us consider a scenario involving a full regression model that utilizes four predictor variables (x_1, x_2, x_3, x_4) to explain the dependent variable (Y):

$$Y = \beta_0 + \beta_1x_1 + \beta_2x_2 + \beta_3x_3 + \beta_4x_4 + \varepsilon$$

A corresponding nested, or reduced, model would systematically eliminate some of those predictors. For example, a reduced model might retain only x_1 and x_2 , demonstrating its subset relationship to the full model:

$$Y = \beta_0 + \beta_1x_1 + \beta_2x_2 + \varepsilon$$

The ultimate goal of the LRT is to formally test whether the exclusion of the extra variables (x_3 and x_4 in this example) leads to a statistically significant deterioration in the model's ability to fit the observed data. This outcome dictates whether the simpler model is statistically sufficient when weighed against the complex one.

Executing the test requires defining clear [null and alternative hypotheses](#) to guide our statistical decision-making process. The framework directly addresses the utility of the additional parameters in the full model:

H₀ (Null Hypothesis): This hypothesis asserts that the restricted (nested) model fits the data equally well as the full model. Statistically, this means the coefficients for the additional variables excluded from the reduced model are effectively zero. If we fail to reject H₀, we conclude that the simpler, **nested model** is sufficient.

H_A (Alternative Hypothesis): This hypothesis claims that the full model achieves a significantly better fit than the nested model. This implies that the additional parameters meaningfully contribute to explaining the variation in the data. Rejecting H₀ in favor of H_A means we should adopt the **full model**.

The decision threshold is determined by comparing the resulting test statistic's [p-value](#) against a pre-selected significance level, typically denoted as α (alpha), where $\alpha = 0.05$ is standard practice. If the [p-value](#) is smaller than this threshold, we possess sufficient evidence to reject the [null hypothesis](#) (H_0). This rejection signifies that the complexity added by the full model is statistically warranted, making it the superior choice for analysis.

We will now transition to a practical implementation guide, detailing how to perform this crucial statistical procedure using powerful Python libraries like `statsmodels` and `scipy`.

Step 1: Preparing and Loading the Data

For this hands-on tutorial, we utilize the classic **mtcars** dataset, which is frequently used for statistical modeling examples. Our objective is to predict the vehicle's miles per gallon (mpg) by comparing a full model that incorporates several explanatory variables against a reduced model that uses a smaller subset of predictors. This setup perfectly satisfies the requirement for nested models necessary for the [likelihood ratio test](#).

We formally define the mathematical structure of our two competing regression models:

Full Model (Four Predictors): The comprehensive model includes displacement (`disp`), carburetor count (`carb`), horsepower (`hp`), and number of cylinders (`cyl`).

$$\text{mpg} = \beta_0 + \beta_1 \text{disp} + \beta_2 \text{carb} + \beta_3 \text{hp} + \beta_4 \text{cyl} + \varepsilon$$

Reduced Model (Two Predictors): The simpler, nested model eliminates `hp` and `cyl`, retaining only `disp` and `carb`.

$$\text{mpg} = \beta_0 + \beta_1 \text{disp} + \beta_2 \text{carb} + \varepsilon$$

The initial step in Python requires importing the necessary statistical and data manipulation libraries, including `pandas` for data handling, `statsmodels` for regression fitting, and `scipy` for statistical testing. Following the imports, we load the `mtcars` dataset directly into a `pandas DataFrame`:

```
from sklearn.linear_model import LinearRegression
import statsmodels.api as sm
import pandas as pd
import scipy
```

```
#Define URL where dataset is located
```

```
url = "https://raw.githubusercontent.com/Statology/Python-Guides/main/mtcars.csv"
```

```
#Read in data using pandas
```

```
data = pd.read_csv(url)
```

Step 2: Fitting the Models and Calculating Log-Likelihood

The statistical foundation of the [likelihood ratio test](#) relies entirely on comparing the maximum achievable [log-likelihood](#) (LLF) values derived from both the full and the reduced models. The LLF quantifies how well the model parameters explain the observed data; intuitively, a higher LLF suggests a better fit.

We begin by defining and fitting the full model using `statsmodels.OLS`. It is essential to remember that `statsmodels` requires explicitly adding a constant (intercept) term using the `sm.add_constant()` function before fitting the regression. After fitting, we extract and store the resulting LLF value for the full model:

```
#Define response variable
```

```
y1 = data
```

```
#Define predictor variables for the full model
```

```
x1 = data[
```

```
#Add necessary constant for OLS regression
```

```
x1 = sm.add_constant(x1)
```

```
#Fit the full regression model
```

```
full_model = sm.OLS(y1, x1).fit()
```

```
#Calculate log-likelihood (llf) of the full model
```

```
full_ll = full_model.llf
```

```
print(full_ll)
```

```
-77.55789711787898
```

Subsequently, we define and fit the reduced model. This step involves selecting only the subset of predictors--`disp` and `carb`--and excluding `hp` and `cyl`. We repeat the process of adding the constant, fitting the model, and extracting the LLF. Since the reduced model has fewer parameters, its LLF value is mathematically expected to be less than or equal to that of the full model, reflecting a potentially poorer fit:

```
#Define response variable
```

```
y2 = data
```

```
#Define predictor variables for the reduced model
x2 = data]

#Add constant to predictor variables
x2 = sm.add_constant(x2)

#Fit the reduced regression model
reduced_model = sm.OLS(y2, x2).fit()

#Calculate log-likelihood (llf) of the reduced model
reduced_ll = reduced_model.llf

print(reduced_ll)

-78.60301334355185
```

Step 3: Performing the Likelihood Ratio Test Calculation

The core of the Likelihood Ratio Test involves calculating the LR test statistic, which quantifies the evidence against the null hypothesis based on the difference in likelihoods. This statistic is defined by the formula: $LR = -2 * (LLF_{reduced} - LLF_{full})$. Because the LLF of the full model is always greater than or equal to the LLF of the reduced model, this calculation results in a non-negative Chi-Squared value.

Under the null hypothesis (H_0), this LR statistic is asymptotically distributed according to a [Chi-Squared distribution](#). To determine the statistical significance of the difference observed, we calculate the corresponding [p-value](#) using the `scipy.stats.chi2.sf` function (survival function, which gives the upper tail probability). Crucially, this function requires us to specify the correct [degrees of freedom](#) (df), which we define as 2 for this comparison, corresponding to the two restricted parameters (hp and cyl).

The following Python code executes the LR statistic calculation and determines the final p-value:

```
#Calculate likelihood ratio Chi-Squared test statistic
LR_statistic = -2*(reduced_ll-full_ll)

print(LR_statistic)

2.0902324513457415

#Calculate p-value of test statistic using 2 degrees of freedom
p_val = scipy.stats.chi2.sf(LR_statistic, 2)
```

```
print(p_val)
```

```
0.35165094613502257
```

Step 4: Interpreting the Statistical Results

The execution of the Python script provided two critical values: a Chi-Squared test statistic of approximately **2.0902** and a resulting [p-value](#) of **0.3517**. The interpretation hinges entirely on comparing this p-value against our predetermined significance level, α , which we set at 0.05.

Since 0.3517 is substantially greater than 0.05, we lack sufficient statistical evidence to reject the [null hypothesis](#) (H_0). Failing to reject H_0 leads to the conclusion that the two models--the complex full model and the simpler reduced model--fit the data equally well. In practical terms, the increase in predictive power gained by adding `hp` and `cyl` is not statistically significant.

This outcome strongly supports the fundamental statistical principle of **parsimony**, which dictates that if two models perform similarly, the simpler model should always be preferred. Thus, the increased complexity of the full model is unwarranted. The [likelihood ratio test](#) confirms that the additional predictor variables (`hp` and `cyl`) do not justify their inclusion, and we should revert to the simpler model for future predictions and interpretations.

The preferred model, based on the findings of the test, is the reduced model:

$$\text{mpg} = \beta_0 + \beta_1 \text{disp} + \beta_2 \text{carb}$$

Understanding Degrees of Freedom (df)

It is paramount to correctly specify the [degrees of freedom](#) (df) when calculating the p-value from the Chi-Squared distribution. The df for the LRT is defined as the difference in the number of parameters between the full model and the reduced model. In our specific case, the full model employed 4 predictor variables (plus the intercept), while the reduced model employed 2 predictor variables (plus the intercept).

Therefore, the calculation is: $\text{df} = (\text{Number of predictors in Full Model}) - (\text{Number of predictors in Reduced Model})$. This resulted in $\text{df} = 4 - 2 = 2$. This value represents the number of coefficients that were restricted to zero in the transition from the full model to the nested model.

Additional Statistical Resources

For readers interested in expanding their knowledge of statistical modeling and implementing advanced regression techniques using Python, the following external resources offer valuable

insights and further tutorials: