How to Perform a Likelihood Ratio Test in Python Using Statsmodels

Authored by

stats writer

December 2, 2025

The likelihood ratio test (LRT) is a powerful and widely utilized statistical method designed to rigorously compare the goodness of fit between two competing statistical models, provided that one model is a restricted, or "nested," version of the other. This test is foundational in fields ranging from econometrics to machine learning, offering a formal framework for model selection based on the principle of maximum likelihood. When implementing statistical analyses in Python, the necessary calculations for the LRT are typically managed through libraries like statsmodels, which provides robust tools for obtaining the log-likelihood values required for the computation.

While specialized functions exist for certain model types, the fundamental mechanism involves calculating the log-likelihood values for both the full (unrestricted) and the nested (restricted) models. The resulting test statistic, which asymptotically follows a Chi-Squared distribution, allows practitioners to determine whether the constraints imposed by the nested model significantly diminish the model's ability to explain the observed data. Understanding how to execute this test precisely in Python is essential for drawing accurate conclusions about model efficiency and parsimony.

The Statistical Foundation: Understanding Nested Models

The core requirement for applying the likelihood ratio test is that the two models being compared must be **nested**. This condition dictates that one model--the simpler, restricted model--must be derivable from the other model--the more complex, full model--by imposing specific constraints on the parameters, typically setting certain coefficients to zero. If the models are not nested, the LRT is statistically invalid, and alternative model comparison techniques, such as the Akaike Information Criterion (AIC) or Bayesian Information Criterion (BIC), should be considered.

A **nested model**, also frequently referred to as a reduced model, is characterized by containing a strict subset of the predictor variables found in the corresponding full model. This simplification is often performed to test the marginal utility of a group of variables. If removing those variables (i.e., constraining their coefficients to zero) does not significantly worsen the model fit, the nested model is preferred for its parsimony.

Consider a practical illustration using a multiple linear regression framework. Suppose we start with a comprehensive model containing four potential predictor variables: $x_1, x_2, x_3,$ and x_4:

$$Y = \beta0 + \beta1x1 + \beta2x2 + \beta3x3 + \beta4x4 + \varepsilon$$

In this scenario, a nested model could be constructed by setting $beta_3 = 0$ and $beta_4 = 0$, effectively removing x_3 and x_4 from the prediction equation. This results in the reduced form:

$$Y = \beta0 + \beta1x1 + \beta2x2 + \varepsilon$$

Formulating the Null and Alternative Hypotheses

To properly conduct the LRT, we must define the statistical hypotheses that guide our decision-making process. The test seeks to determine whether the additional complexity of the full model provides a statistically significant improvement in the model's overall fit compared to the nested model. The null hypothesis (H_0) represents the state of parsimony, asserting that the extra predictors are unnecessary. Conversely, the alternative hypothesis (H_A) suggests that the full model is indeed superior.

The formal hypotheses are stated as follows:

H0 (Null Hypothesis): The full model and the nested model possess statistically equivalent goodness-of-fit to the data. This implies that the restricted parameters (e.g., $beta_3$ and $beta_4$ in the example above) are simultaneously equal to zero. Therefore, if we fail to reject H_0, the preferred outcome is to **use the nested model** due to its simplicity.
HA (Alternative Hypothesis): The full model provides a fit that is significantly better than the nested model. This suggests that at least one of the restricted parameters is non-zero. If we reject H_0 in favor of H_A, the appropriate decision is to **use the full model**.

The decision criterion hinges on the resulting p-value. If the p-value derived from the test is below a predetermined level of significance (e.g., 0.05), then we possess sufficient evidence to reject the null hypothesis. Rejecting H_0 confirms that the increment in log-likelihood achieved by the full model is statistically meaningful, justifying the inclusion of the additional predictor variables.

The following step-by-step example shows how to perform a likelihood ratio test in Python by fitting real-world data to compare two linear regression specifications.

Step 1: Data Acquisition and Preparation in Python

To demonstrate the Likelihood Ratio Test, we will utilize the well-known **mtcars** dataset, a classic dataset containing information on 32 automobiles. Our goal is to compare two ordinary least squares (OLS) regression models predicting miles per gallon (`mpg`) based on various engine characteristics.

The two models under consideration are defined as follows:

Full Model (Unrestricted): $mpg = beta_0 + beta_1disp + beta_2carb + beta_3hp + beta_4cyl$
Reduced Model (Nested): $mpg = beta_0 + beta_1disp + beta_2carb$

The first crucial step involves importing the necessary Python libraries--including `pandas` for data handling, `statsmodels.api` for regression modeling, and `scipy` for statistical calculations--and loading the dataset from the specified URL.

```
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/arabpsychology/Python-Guides/main/mtcars.csv"

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

After executing this code block, the `data` DataFrame will hold the motor trend car data, prepared for the subsequent modeling stages. It is important to confirm data integrity before proceeding.

Step 2: Fitting the Models and Extracting Log-Likelihoods

The Likelihood Ratio Test fundamentally relies on the maximum value of the log-likelihood function achieved by each model. In OLS regression, the log-likelihood calculation quantifies how well the estimated parameters explain the observed data. A higher log-likelihood indicates a better fit. We must fit both the full and the reduced models separately to obtain these crucial values.

Fitting the Full (Unrestricted) Model

First, we define the response variable (`mpg`) and the complete set of predictor variables for the full model (`disp`, `carb`, `hp`, `cyl`). Using statsmodels.api.OLS, we fit the linear regression and then access the `llf` attribute (log-likelihood function value) from the fitted model object.

```
#define response variable
y1 = data

#define predictor variables
x1 = data]

#add constant to predictor variables
x1 = sm.add_constant(x1)

#fit regression model
full_model = sm.OLS(y1, x1).fit()

#calculate log-likelihood of model
full_ll = full_model.llf
```

```
print(full_ll)
```

```
-77.55789711787898
```

The resulting log-likelihood value, approximately -77.56, represents the highest achievable likelihood when all four predictor variables are included.

Fitting the Reduced (Nested) Model

Next, we repeat the fitting process for the reduced model, which only includes the predictors `disp` and `carb`. Crucially, the reduced model must be fitted using the same response variable and the same underlying data structure to ensure a valid comparison using the LRT methodology.

```
#define response variable
y2 = data

#define predictor variables
x2 = data]

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

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

#calculate log-likelihood of model
reduced_ll = reduced_model.llf

print(reduced_ll)
```

```
-78.60301334355185
```

The log-likelihood for the reduced model is slightly lower, approximately -78.60. While the full model necessarily achieves a higher log-likelihood, the LRT is required to determine if this difference is statistically significant.

Step 3: Calculating the Likelihood Ratio Test Statistic

The formal likelihood ratio test statistic, G, is calculated based on the difference between the log-likelihoods of the two models. The formula utilizes the ratio of the two likelihoods, which translates to a difference when using logarithms:

$$G = -2 times (ln(L_{reduced}) - ln(L_{full}))$$

Where $ln(L_{reduced})$ is the log-likelihood of the nested model, and $ln(L_{full})$ is the log-likelihood of the full model. Under the null hypothesis, this test statistic G asymptotically follows a Chi-Squared distribution. We implement this calculation directly in Python:

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

print(LR_statistic)

2.0902324513457415

The calculated likelihood ratio test statistic, or G value, is approximately **2.0902**. This value quantifies the evidence against the null hypothesis, but it requires conversion to a p-value for statistical interpretation.

Step 4: Determining Degrees of Freedom and P-Value

To calculate the p-value, we must determine the degrees of freedom (df) associated with the test statistic. The degrees of freedom for the LRT statistic are equal to the difference in the number of parameters between the full model and the nested model. This corresponds precisely to the number of predictor variables constrained to zero under the null hypothesis.

In our specific comparison, the parameters are counted as follows:

Full Model parameters (including intercept): 5 (disp, carb, hp, cyl, Intercept)
Reduced Model parameters (including intercept): 3 (disp, carb, Intercept)
Degrees of Freedom (df): $5 - 3 = 2$.

We utilize the `scipy.stats.chi2.sf` function (the survival function, which gives $1 - CDF$) to calculate the p-value corresponding to the Chi-Squared test statistic ($G = 2.0902$) with 2 degrees of freedom.

#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

From the output, we observe that the Chi-Squared test-statistic is **2.0902** and the corresponding p-

value is **0.3517**.

Step 5: Interpretation and Final Conclusion

The final step involves comparing the calculated p-value to our predetermined significance level ($alpha = 0.05$).

Since the p-value (0.3517) is not less than the critical threshold of 0.05, we **fail to reject the null hypothesis**.

This outcome signifies that the increase in log-likelihood provided by the full model is not statistically significant. In other words, the full model and the nested model fit the observed data equally well when accounting for the penalty of adding extra parameters. Thus, based on the principle of parsimony (preferring the simpler model when fit is equivalent), we should choose the nested model because the additional predictor variables (`hp` and `cyl`) do not offer a significant improvement in fit.

Thus, our final recommended model for predicting MPG would be the simpler formulation:

$$mpg = beta_0 + beta_1disp + beta_2carb$$

Note: The selection of 2 degrees of freedom was accurate because it represents the difference between the number of parameters in the full model and the nested model--specifically, the two coefficients that were constrained to zero under the null hypothesis.

Further Resources on Regression Modeling

The successful execution of the Likelihood Ratio Test confirms its utility in rigorous model comparison. For those interested in advancing their skills in statistical modeling and regression using Python, exploring further documentation on the statsmodels library and understanding the underlying statistical distributions, such as the Chi-Squared distribution, is highly recommended.

The following tutorials provide additional information about how to use regression models in Python: