How can I perform a Likelihood Ratio Test using Python?

Authored by

stats writer

July 1, 2024

A Likelihood Ratio Test is a statistical method used to compare the fit of two different models for a given dataset. It allows for determining which model provides a better explanation for the observed data. In Python, this test can be performed using the "statsmodels" library, specifically the "stats" module. The first step is to fit the two competing models using the "fit" function. Then, the likelihood ratio can be calculated using the "compare_lr_test" function, which takes the two fitted models as arguments. The result of this function is a test statistic and a p-value, which can be used to determine if there is a significant difference in the fit of the two models. This test can provide valuable insights into the underlying relationships in a dataset and aid in making informed decisions.

Perform a Likelihood Ratio Test in Python

A likelihood ratio test compares the goodness of fit of two nested .

A is simply one that contains a subset of the predictor variables in the overall regression model.

For example, suppose we have the following regression model with four predictor variables:

$Y = \beta 0 + \beta 1x1 + \beta 2x2 + \beta 3x3 + \beta 4x4 + \varepsilon$

One example of a nested model would be the following model with only two of the original predictor variables:

$Y = \beta 0 + \beta 1x1 + \beta 2x2 + \varepsilon$

To determine if these two models are significantly

different, we can perform a likelihood ratio test which uses the following null and alternative hypotheses:

H0: The full model and the nested model fit the data equally well. Thus, you should use the nested model.

HA: The full model fits the data significantly better than the nested model. Thus, you should use the full model.

If the of the test is below a certain significance level (e.g. 0.05), then we can reject the null hypothesis and conclude that the full model offers a significantly better fit.

The following step-by-step example shows how to perform a likelihood ratio test in Python.

Step 1: Load the Data

In this example, we'll show how to fit the following two regression models in Python using data from the mtcars dataset:

Full model: mpg = β0 + β1disp + β2carb + β3hp + β4cyl

Reduced model: mpg = β0 + β1disp + β2carb

First, we'll load the dataset:

```
from sklearn.linear_modelimport LinearRegression
import statsmodels.apias 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
data = pd.read_csv(url)
```

Step 2: Fit the Regression Models

First, we'll fit the full model and calculate the log-likelihood of the model:

```
#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
```

Then, we'll fit the reduced model and calculate the log-likelihood of the model:

```
#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
```

Step 3: Perform the Log-Likelihood Test

Next, we'll use the following code to perform the log-likelihood test:

```
#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)

0.3516509461350225 7
```

From the output we can see that the Chi-Squared test-statistic is 2.0902 and the corresponding p-value is 0.3517.

Since this p-value is not less than .05, we will fail to reject the null hypothesis.

This means the full model and the nested model fit the data equally well. Thus, we should use the nested model because the additional predictor variables in the full model don't offer a significant improvement in fit.

Thus, our final model would be:

mpg = β0 + β1disp + β2carb

Note: We used 2 degrees of freedom when calculating the p-value because this represented the difference between the total predictor variables used between the two models.

Additional Resources

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