How can we extract p-values from a linear regression model in Statsmodels?

Authored by

stats writer

June 27, 2024

The process of extracting p-values from a linear regression model in Statsmodels involves using the "summary" function to obtain a summary table of the model. This table includes the p-values for each variable in the model, which can be used to determine the significance of the relationship between the variables and the dependent variable. These p-values can be interpreted as the probability of obtaining a result as extreme as the observed one, assuming that the null hypothesis is true. By comparing the p-values to a chosen significance level, typically 0.05, we can determine if the variables have a significant impact on the outcome of the model. This process allows for a thorough analysis and interpretation of the results of a linear regression model in Statsmodels.

Extract P-Values from Linear Regression in Statsmodels

You can use the following methods to extract p-values for the coefficients in a linear regression model fit using the module in Python:

#extract p-values for all predictor variables
for x in range (0, 3):
print(model.pvalues)

#extract p-value for specific predictor variable name
model.pvalues.loc

#extract p-value for specific predictor variable position
model.pvalues

The following examples show how to use each method in practice.

Example: Extract P-Values from Linear Regression in Statsmodels

Suppose we have the following pandas DataFrame that contains information about hours studied, prep exams taken, and final score received by students in a certain class:

```
import pandas as pd

#create DataFrame
df = pd.DataFrame({'hours': ,
'exams': ,
'score': })

#view head of DataFrame
df.head()
```

	hours	exams	score
0	1	1	76
1	2	3	78
2	2	3	85
3	4	5	88
4	2	2	72

We can use the OLS() function from the statsmodels

module to fit a , using "hours" and "exams" as the predictor variables and "score" as the :

```
import statsmodels.apias sm

#define predictor and response variables
y = df
x = df]

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

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

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

OLS Regression Results

```
==============================================================================
Dep. Variable: score R-squared: 0.718
Model: OLS Adj. R-squared: 0.661
Method: Least Squares F-statistic: 12.70
Date: Fri, 05 Aug 2022 Prob (F-statistic): 0.00180
Time: 09:24:38 Log-Likelihood: -38.618
```

No. Observations: 13 AIC: 83.24

Df Residuals: 10 BIC: 84.93

Df Model: 2

Covariance Type: nonrobust

==========

==========

coef std err t P>|t|

----

const 71.4048 4.001 17.847 0.000 62.490 80.319

hours 5.1275 1.018 5.038 0.001 2.860 7.395

exams -1.2121 1.147 -1.057 0.315 -3.768 1.344

==========

==========

Omnibus: 1.103 Durbin-Watson: 1.248

Prob(Omnibus): 0.576 Jarque-Bera (JB): 0.803

Skew: -0.289 Prob(JB): 0.669

Kurtosis: 1.928 Cond. No. 11.7

==========

==========

By default, the summary() function displays the p-values of each predictor variable up to three decimal places:

P-value for intercept: 0.000P-value for hours: 0.001P-value for exams: 0.315

However, we can extract the full p-values for each predictor variable in the model by using the following syntax:

```
#extract p-values for all predictor variables
for x in range (0, 3):
print(model.pvalues)
```

```
6.514115622692573e-09
0.0005077783375870773
0.3154807854805659
```

This allows us to see the p-values to more decimal places:

P-value for intercept: 0.000000006514115622692573P-value for hours: 0.0005077783375870773P-value for exams: 0.3154807854805659

Note: We used 3 in our range() function because there were three total coefficients in our regression model.

We can also use the following syntax to extract the p-

value for the 'hours' variable specifically:

#extract p-value for 'hours' only
model.pvalues.loc

0.0005077783375870773

#extract p-value for coefficient in index position 0
model.pvalues

6.514115622692573e-09

The following tutorials explain how to perform other common tasks in Python: