How can I perform weighted least squares regression in Python?

Authored by

stats writer

June 26, 2024

Weighted least squares regression is a statistical method used to fit a linear model to a dataset by giving more weight to certain data points based on their importance or reliability. In Python, this can be performed by using the Statsmodels library, which provides a function called WLS that allows the user to specify the weights for each data point. The WLS function then calculates the coefficients for the linear model by minimizing the weighted sum of squared residuals. This method is useful when dealing with datasets containing outliers or unequal variances among the data points. By performing weighted least squares regression in Python, one can obtain a more accurate and robust model that takes into account the varying importance of the data points.

Perform Weighted Least Squares Regression in Python

One of the key is that the are distributed with equal variance at each level of the predictor variable. This assumption is known as homoscedasticity.

When this assumption is violated, we say that is present in the residuals. When this occurs, the results of the regression become unreliable.

One way to handle this issue is to instead use weighted least squares regression, which places weights on the such that those with small error variance are given more weight since they contain more information compared to observations with larger error variance.

This tutorial provides a step-by-step example of how to perform weight least squares regression in Python.

Step 1: Create the Data

First, let's create the following pandas DataFrame that contains information about the number of hours studied and the final exam score for 16 students in some class:

```
import pandas as pd

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

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

hours score
0 1 48
1 1 78
2 2 72
3 2 70
4 2 66
```

Step 2: Fit Simple Linear Regression Model

Next, we'll use functions from the statsmodels module to fit a simple linear regression model using hours as

the predictor variable and score as the response variable:

```
import statsmodels.api as sm

#define predictor and response variables
y = df
X = df

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

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

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

```
OLS Regression Results
==============================================================================
Dep. Variable: score R-squared: 0.630
Model: OLS Adj. R-squared: 0.603
Method: Least Squares F-statistic: 23.80
Date: Mon, 31 Oct 2022 Prob (F-statistic): 0.000244
Time: 11:19:54 Log-Likelihood: -57.184
```

No. Observations: 16 AIC: 118.4

Df Residuals: 14 BIC: 119.9

Df Model: 1

Covariance Type: nonrobust

==========

==========

coef std err t P>|t|

----

const 60.4669 5.128 11.791 0.000 49.468 71.465

hours 5.5005 1.127 4.879 0.000 3.082 7.919

==========

==========

Omnibus: 0.041 Durbin-Watson: 1.910

Prob(Omnibus): 0.980 Jarque-Bera (JB): 0.268

Skew: -0.010 Prob(JB): 0.875

Kurtosis: 2.366 Cond. No. 10.5

From the model summary we can see that the R-squared value of the model is 0.630.

Related:

Step 3: Fit Weighted Least Squares Model

Next, we can use the WLS() function from statsmodels

to perform weighted least squares by defining the weights in such a way that the observations with lower variance are given more weight:

```
#define weights to use
wt = 1 / smf.ols('fit.resid.abs() ~ fit.fittedvalues', data=df).fit().fittedvalues**2

#fit weighted least squares regression model
fit_wls = sm.WLS(y, X, weights=wt).fit()

#view summary of weighted least squares regression model
print(fit_wls.summary())
```

```
WLS Regression Results
==============================================================================
Dep. Variable: score R-squared: 0.676
Model: WLS Adj. R-squared: 0.653
Method: Least Squares F-statistic: 29.24
Date: Mon, 31 Oct 2022 Prob (F-statistic): 9.24e-05
Time: 11:20:10 Log-Likelihood: -55.074
No. Observations: 16 AIC: 114.1
Df Residuals: 14 BIC: 115.7
```

Df Model: 1

Covariance Type: nonrobust

==========

==========

coef std err t P>|t|

----

const 63.9689 5.159 12.400 0.000 52.905 75.033

hours 4.7091 0.871 5.407 0.000 2.841 6.577

==========

==========

Omnibus: 2.482 Durbin-Watson: 1.786

Prob(Omnibus): 0.289 Jarque-Bera (JB): 1.058

Skew: 0.029 Prob(JB): 0.589

Kurtosis: 1.742 Cond. No. 17.6

==========

==========

From the output we can see that the R-squared value for this weighted least squares model increased to 0.676.

This indicates that the weighted least squares model is able to explain more of the variance in exam scores compared to the simple linear regression model.

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