How do we make predictions using a regression model in Statsmodels?

Authored by

stats writer

June 27, 2024

In order to make predictions using a regression model in Statsmodels, we first need to have a dataset with the variables we want to analyze. Then, we can use the Statsmodels library in Python to fit the dataset to a chosen regression model, such as linear or logistic regression. This involves finding the best parameters for the model that will minimize the error between the predicted values and the actual values in the dataset. Once the model is fitted, we can use it to make predictions for new data by inputting the relevant variables and using the model's coefficients to calculate the predicted outcome. This process allows us to analyze relationships between variables and make accurate predictions based on the data.

Make Predictions Using Regression Model in Statsmodels

You can use the following basic syntax to use a regression model fit using the module in Python to make predictions on new observations:

model.predict(df_new)

This particular syntax will calculate the predicted response values for each row in a new DataFrame called df_new, using a regression model fit with statsmodels called model.

The following example shows how to use this syntax in practice.

Example: Make Predictions Using Regression Model 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 response

variable:

```
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

==========

==========

From the coef column in the output, we can write the fitted regression model:

Score = 71.4048 + 5.1275(hours) - 1.2121(exams)

Now suppose we would like to use the fitted regression model to predict the "score" for five new students.

First, let's create a DataFrame to hold the five new observations:

```
#create new DataFrame
df_new = pd.DataFrame({'hours': ,
'exams': })

#add column for constant
df_new = sm.add_constant(df_new)

#view new DataFrame
print(df_new)

const hours exams
0 1.0 1 1
1 1.0 2 1
2 1.0 2 4
3 1.0 4 3
4 1.0 5 3
```

Next, we can use the predict() function to predict the "score" for each of these students, using "hours" and

"exams" as the values for the predictor variables in our fitted regression model:

#predict scores for the five new students
model.predict(df_new)

0 75.320242
1 80.447734
2 76.811480
3 88.278550
4 93.406042
dtype: float64

Here's how to interpret the output:

The first student in the new DataFrame is predicted to get a score of 75.32.The second student in the new DataFrame is predicted to get a score of 80.45.

And so on.

To understand how these predictions were calculated, we need to refer to the fitted regression model from earlier:

Score = 71.4048 + 5.1275(hours) - 1.2121(exams)

By plugging in the values for "hours" and "exams" for the new students, we can calculate their predicted score.

For example, the first student in the new DataFrame had a value of 1 for hours and a value of 1 for exams.

Thus, their predicted score was calculated as:

Score = 71.4048 + 5.1275(1) - 1.2121(1) = 75.32.

The score of each student in the new DataFrame was calculated in a similar manner.

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