How can I obtain a regression model summary from Scikit-Learn?

Authored by

stats writer

June 29, 2024

To obtain a regression model summary from Scikit-Learn, follow these steps:
1. Import the necessary modules and libraries such as sklearn.linear_model and sklearn.metrics.
2. Load your dataset and split it into training and testing sets.
3. Choose a regression model from Scikit-Learn, such as LinearRegression or RandomForestRegressor.
4. Fit the model on the training data and make predictions on the testing data.
5. Use the sklearn.metrics module to evaluate the performance of your model by calculating metrics such as mean squared error and R-squared.
6. To obtain a summary of your model's performance, use the model's .summary() or .summary_params() method.
This will provide key information such as the coefficients, intercept, and evaluation metrics of your regression model.

Get Regression Model Summary from Scikit-Learn

Often you may want to extract a summary of a regression model created using in Python.

Unfortunately, scikit-learn doesn't offer many built-in functions to analyze the summary of a regression model since it's typically only used for .

So, if you're interested in getting a summary of a regression model in Python, you have two options:

1. Use limited functions from scikit-learn.

2. Use instead.

The following examples show how to use each method

in practice with the following pandas DataFrame:

```
import pandas as pd

#create DataFrame
df = pd.DataFrame({'x1': ,
'x2': ,
'y': })

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

x1 x2 y
0 1 1 76
1 2 3 78
2 2 3 85
3 4 5 88
4 2 2 72
```

Method 1: Get Regression Model Summary from Scikit-Learn

We can use the following code to fit a model using scikit-learn:

```
from sklearn.linear_modelimport LinearRegression
```

#initiate linear regression model
model = LinearRegression()

#define predictor and response variables
X, y = df], df.y#fit regression model
model.fit(X, y)

We can then use the following code to extract the regression coefficients of the model along with the of the model:

#display regression coefficients and R-squared value of modelprint(model.intercept_, model.coef_, model.score(X, y))

70.4828205704 0.766742556527

Using this output, we can write the equation for the fitted regression model:

y = 70.48 + 5.79x1 - 1.16x2

We can also see that the R2 value of the model is 76.67.

This means that 76.67% of the variation in the response

variable can be explained by the two predictor variables in the model.

Although this output is useful, we still don't know the of the model, the p-values of the individual , and other useful metrics that can help us understand how well the model fits the dataset.

Method 2: Get Regression Model Summary from Statsmodels

If you're interested in extracting a summary of a regression model in Python, you're better off using the statsmodels package.

The following code shows how to use this package to fit the same multiple linear regression model as the previous example and extract the model summary:

```
import statsmodels.apias sm

#define response variable
y = df

#define predictor variables
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: y R-squared: 0.767
Model: OLS Adj. R-squared: 0.708
Method: Least Squares F-statistic: 13.15
Date: Fri, 01 Apr 2022 Prob (F-statistic): 0.00296
Time: 11:10:16 Log-Likelihood: -31.191
No. Observations: 11 AIC: 68.38
Df Residuals: 8 BIC: 69.57
Df Model: 2
Covariance Type: nonrobust
==============================================================================
coef std err t P>|t|
------------------------------------------------------------------------------
const 70.4828 3.749 18.803 0.000 61.839 79.127
```

x1 5.7945 1.132 5.120 0.001 3.185 8.404

x2 -1.1576 1.065 -1.087 0.309 -3.613 1.298

==========

==========

Omnibus: 0.198 Durbin-Watson: 1.240

Prob(Omnibus): 0.906 Jarque-Bera (JB): 0.296

Skew: -0.242 Prob(JB): 0.862

Kurtosis: 2.359 Cond. No. 10.7

==========

==========

Notice that the regression coefficients and the R-squared value match those calculated by scikit-learn, but we're also provided with a ton of other useful metrics for the regression model.

For example, we can see the p-values for each individual predictor variable:

p-value for x1 = .001p-value for x2 = 0.309

We can also see the overall F-statistic of the model, the value, the of the model, and much more.

Additional Resources

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

How to Perform Simple Linear Regression in Python

How to Perform Multiple Linear Regression in Python