How can an Anderson-Darling test be performed in Python?

Authored by

stats writer

April 17, 2024

The Anderson-Darling test is a statistical test used to assess whether a given data set follows a particular distribution. In Python, this test can be performed by importing the "scipy" library and using the "anderson" function from the "stats" module. This function takes in the data set as an input and returns the Anderson-Darling test statistic and corresponding critical values. The test can then be interpreted by comparing the test statistic to the critical values and determining the level of significance. This allows for a quick and efficient way to assess the goodness-of-fit of a data set to a specific distribution in Python.

Perform an Anderson-Darling Test in Python

An Anderson-Darling Test is a goodness of fit test that measures how well your data fit a specified distribution.

This test is most commonly used to determine whether or not your data follow a .

This type of test is useful for testing for normality, which is a common assumption used in many statistical tests including , , , and many others.

Example: Anderson-Darling Test in Python

To conduct an Anderson-Darling Test in Python, we can use the from the scipy.stats library, which uses the following syntax:

anderson(x, dist='norm')

where:

x: array of sample datadist: the type of distribution to test against. Default is 'norm' but you can also specify 'expon' or 'logistic.'

For example, here's how to perform an Anderson-Darling Test on a sample of 50 normally distributed random variables:

```
import numpy as np

#create data
np.random.seed(0)
data = np.random.normal(size=50)

#perform Anderson-Darling Testfrom scipy.stats import anderson
anderson(data)

AndersonResult(statistic=0.1500699953338866 5,
critical_values=array(),
significance_level=array())
```

The test statistic is 0.150. We can compare this value to each critical value that corresponds to each significance level to see if the test results are

significant. For example:

The critical value for α = 0.01 is 1.021. Because the test statistic (0.150) is not greater than this critical value, the results are not significant at a significance level of 0.01.The critical value for α = 0.025 is 0.858. Because the test statistic (0.150) is not greater than this critical value, the results are not significant at a significance level of 0.025.

And so on.

We can see that the test results are not significant at any significance level, which means we would not reject the null hypothesis of the test. Thus, we don't have sufficient evidence to say that the sample data is not normally distributed.

This result shouldn't be surprising considering we used the np.rand.normal() function to generate a sample of 50 normally distributed values.

Consider instead if we performed the Anderson-Darling Test on a sample of 50 random integers between 0 and 10:

```
import numpy as np

#create data
np.random.seed(0)
data = np.random.randint(0, 10, size=50)

#perform Anderson-Darling Testfrom scipy.stats import anderson
anderson(data)

AndersonResult(statistic=1.1926463985076836,
critical_values=array(),
significance_level=array())
```

The critical value for α = 0.01 is 1.021. Because the test statistic (1.1926) is greater than this critical value, the results are significant at a significance level of 0.01.The critical value for α = 0.025 is 0.858. Because the test statistic (1.1926) is greater than this critical value, the results are significant at a significance level of 0.025.

And so on.

We can see that the test results are significant at every significance level, which means we would reject the null

hypothesis of the test no matter which significance level we choose to use. Thus, we have sufficient evidence to say that the sample data is not normally distributed.

This result also shouldn't be surprising considering we used the np.rand.randint() function to generate a sample of 50 random integers between 0 and 10, which is unlikely to follow a normal distribution.

You can find more Python tutorials .