Anderson-Darling Goodness-of-Fit Test Tutorial in Python

Authored by

Mohammed looti

November 7, 2025

The **Anderson-Darling Test** is recognized as a powerful and widely utilized statistical procedure for assessing the Goodness-of-Fit. This test quantifies the discrepancy between the empirical cumulative distribution function (ECDF) of your observed data and the cumulative distribution function (CDF) of a theoretical distribution that you are testing against. Unlike older tests, the Anderson-Darling method places greater weight on the tails of the distribution, making it particularly sensitive to deviations in these extreme regions, which is often crucial for robust statistical modeling.

While the test can be adapted for various distribution types, it is most commonly employed to determine whether or not your data accurately follow a normal distribution. Establishing this normality assumption is a prerequisite for many classical parametric statistical tests. If the data significantly deviate from normality, the results derived from these parametric methods may be unreliable or outright invalid. Consequently, mastering this test is foundational for robust data analysis.

This type of goodness-of-fit assessment is essential before proceeding with procedures such as t-tests, ANOVA (Analysis of Variance), linear regression modeling, and numerous other inferential techniques. Ensuring the underlying distributional assumptions are met guarantees that the calculated p-values and confidence intervals accurately reflect the true uncertainty and relationships within the data. We will now explore how to efficiently implement and interpret this critical test using the powerful Python ecosystem.

Theoretical Foundations of the Anderson-Darling Test

The core mechanism of the Anderson-Darling test relies on comparing the sample data's cumulative distribution against the hypothesized theoretical distribution. The test statistic, often denoted as A^2, is derived from the squared difference between the empirical and theoretical cumulative distribution functions. A higher A^2 value indicates a larger deviation from the hypothesized distribution, suggesting a poorer fit.

The process begins with formulating the null hypothesis (H_0) and the alternative hypothesis (H_a). For the Anderson-Darling test, when testing for normality, these are defined as follows: H_0: The sample data is drawn from a population that follows the specified distribution (e.g., the normal distribution). H_a: The sample data is not drawn from a population that follows the specified distribution. The goal of the test is to determine if there is enough statistical evidence to reject H_0 in favor of H_a.

The calculation of the test statistic is intricate, involving integration and weighting factors that emphasize the distribution tails. The critical step in interpreting the result involves comparing the calculated A^2 statistic against pre-defined critical values associated with various significance levels ($alpha$). If the calculated statistic exceeds the critical value for a given $alpha$, we reject

the null hypothesis, concluding that the data does not fit the theoretical distribution at that specific level of confidence.

Implementing the Anderson-Darling Test in Python

To conduct an Anderson-Darling Test efficiently in Python, data scientists rely on the widely respected SciPy library, specifically utilizing the statistical module, `scipy.stats`. This module provides the `anderson` function, which seamlessly calculates the test statistic, critical values, and significance levels necessary for interpretation. Before running the test, ensure both SciPy and NumPy are installed in your environment, as NumPy handles the underlying array manipulation.

The basic function call for the Anderson-Darling test adheres to the following syntax, demonstrating its simplicity and ease of use:

```
anderson(x, dist='norm')
```

This function requires minimal input but offers powerful flexibility. The parameters are defined as follows:

`x`: This required parameter is the array or sequence of sample data upon which the goodness-of-fit test will be performed. This should be a NumPy array or a similar structure containing the observations.
`dist`: This optional parameter specifies the type of theoretical distribution to test against. By default, the function assumes `'norm'` (the normal distribution). However, the SciPy implementation also supports testing against other common distributions, such as `'expon'` (the exponential distribution) or `'logistic'` (the logistic distribution), allowing for broader applicability beyond simple normality testing.

Understanding these parameters is key to running a successful test. The default setting of `dist='norm'` is sufficient for the vast majority of applications where normality must be verified, which we will demonstrate in the following examples.

Case Study 1: Testing Normally Distributed Data

To illustrate the proper usage and interpretation of the `anderson` function, we will first generate a sample of data that is known to follow a normal distribution. This serves as an excellent control case, allowing us to confirm that the test correctly identifies data that conforms to the null hypothesis. We will generate 50 random variables using NumPy's standard normal function.

The following code snippet demonstrates the creation of the synthetic data, followed by the execution of the Anderson-Darling test:

import numpy as np

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

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

AndersonResult(statistic=0.15006999533388665,
critical_values=array(),
significance_level=array())

The output of the function is an `AndersonResult` object, which contains three key components: the calculated test statistic, an array of critical values, and the corresponding array of significance levels (expressed as percentages). This structure allows for immediate comparison without requiring manual lookups in statistical tables. For this specific run, the test statistic is approximately **0.150**. Our subsequent step involves comparing this single statistic against the array of critical values to determine significance.

Interpreting the Test Results and Critical Values

The interpretation of the Anderson-Darling test hinges on a straightforward decision rule: if the calculated test statistic (A^2) is greater than the critical value associated with a chosen significance level ($alpha$), we reject the null hypothesis. If the test statistic is less than or equal to the critical value, we fail to reject the null hypothesis.

In our first case study, the test statistic was calculated as $A^2 approx 0.150$. The SciPy output provides a set of critical values corresponding to standard significance levels:

The critical value for a significance level of $alpha = 0.01$ (or 1%) is **1.021**. Because the test statistic (0.150) is not greater than this critical value, the results are not statistically significant at this level. We do not reject H_0.
The critical value for $alpha = 0.025$ (or 2.5%) is **0.858**. Since $0.150 le 0.858$, the results are again not significant.
This pattern continues across all provided significance levels (5%, 10%, and 15%).

Crucially, because the calculated test statistic (0.150) is substantially smaller than the smallest critical value (0.538, corresponding to $alpha = 0.15$), we conclude that the test results are not significant at any conventionally accepted significance level. Therefore, we do not reject the null

hypothesis. This outcome means we do not have sufficient statistical evidence to conclude that the sample data is anything other than normally distributed. This result is entirely expected and serves as a validation check, given that the data was intentionally generated using the `np.random.normal()` function to follow a standard normal distribution.

Case Study 2: Testing Non-Normal Data

A more challenging and realistic scenario involves testing data that is unlikely to follow a normal distribution. In this second case study, we will generate a sample of 50 random integers uniformly distributed between 0 and 10. This discrete, uniform distribution is fundamentally different from the continuous, symmetric bell curve of the normal distribution, and we expect the Anderson-Darling test to yield a highly significant result, leading to the rejection of the null hypothesis.

We use the `np.random.randint()` function to create this non-normal dataset, and then apply the `anderson` function exactly as before:

import numpy as np

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

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

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

In this second test, the calculated test statistic has increased dramatically to $A^2 approx 1.1926$. This significantly higher value immediately suggests a strong deviation from the normal distribution. We now apply the same critical value comparison rule:

The critical value for $alpha = 0.01$ (1%) is **1.021**. Because the test statistic (1.1926) is greater than this critical value, the results are significant at the 0.01 level. We reject H_0.
The critical value for $alpha = 0.025$ (2.5%) is **0.858**. Since $1.1926 > 0.858$, the results are also significant at the 0.025 level. We reject H_0.

In fact, the test statistic (1.1926) is greater than every single critical value provided in the result object, meaning the test results are statistically significant at all standard significance levels (up to

15%). This obligates us to reject the null hypothesis regardless of the $alpha$ level chosen. Consequently, we have strong and sufficient evidence to state definitively that the sample data is not normally distributed.

This result is consistent with our expectation, as the data originated from the `np.random.randint()` function, generating a sample of 50 random integers which forms a discrete uniform distribution, making it highly unlikely to conform to the continuous, symmetric properties of the normal distribution. The two case studies perfectly demonstrate how the Anderson-Darling test successfully discriminates between normally distributed and non-normally distributed datasets.

You can find more Python tutorials here.