Calculate Cook’s Distance in Python

Authored by

Mohammed looti

November 6, 2025

Identifying influential observations is a critical step in validating any statistical analysis. The **Cook's distance** metric is a widely utilized tool specifically designed to help analysts pinpoint data points that significantly alter the results of a **regression model**.

When an observation exhibits a large Cook's distance, it suggests that removing that single point from the dataset would substantially change the parameter estimates of the model. Understanding this influence is essential for ensuring the robustness and reliability of your predictive findings.

The Mathematical Foundation of Cook's Distance

The calculation of Cook's distance, denoted as Di, combines two core statistical concepts: the magnitude of the **residual** and the **leverage value** of the observation. The standard formula is presented as follows:

Di = (ri2 / p*MSE) * (hii / (1-hii)2)

In this formula, the variables represent key properties derived from the regression analysis:

ri is the ith **residual**, measuring the difference between the observed and predicted values.
p is the number of coefficients (including the intercept) estimated in the **regression model**.
MSE is the **Mean Squared Error**, used as an estimate of the error variance.
hii is the ith **leverage value**, which measures how far the observation's explanatory variables are from the mean of the explanatory variables.

Interpreting Influence and Identifying Outliers

Essentially, **Cook's distance** measures the magnitude of change across all of the fitted values in the model when the ith observation is hypothetically deleted. A larger value for Cook's distance directly correlates to a more influential observation.

To set a threshold for identifying problematic observations, a generally accepted **rule of thumb** suggests that any observation with a Cook's distance greater than 4/n (where *n* equals the total number of observations) should be considered highly influential and flagged for review.

This tutorial provides a clear, step-by-step example of how to calculate **Cook's distance** for a given **regression model** using the robust statistical tools available in **Python**.

Step 1: Preparing the Data in Python

First, we must prepare the data in a suitable structure. We will create a small sample dataset using the **Pandas** library in **Python** to hold our independent variable (x) and dependent variable (y):

import pandas as pd

#create dataset
df = pd.DataFrame({'x': ,
'y': })

Step 2: Fitting the Regression Model

Next, we need to fit an Ordinary Least Squares (OLS) **regression model**. We will use the **Statsmodels** library, which provides comprehensive classes for estimating statistical models.

It is important to properly define our response and explanatory variables, and crucially, add a constant term to the predictors to account for the intercept in the model equation.

import statsmodels.api as sm

#define response variable
y = df

#define explanatory variable
x = df

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

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

Step 3: Calculating Cook's Distance

With the model fitted, we can now calculate the **Cook's distance** for each observation. The Statsmodels results object provides the `get_influence()` method, which contains the necessary diagnostic statistics.

We suppress scientific notation for cleaner output and then extract the array of Cook's distance values:

#suppress scientific notation
import numpy as np
np.set_printoptions(suppress=True)

```
#create instance of influence
influence = model.get_influence()

#obtain Cook's distance for each observation
cooks = influence.cooks_distance

#display Cook's distances
print(cooks)

(array(),
array())
```

By default, the `cooks_distance()` function displays a tuple containing two arrays. The first array lists the calculated Cook's distance value for each observation, followed by the second array which contains the corresponding diagnostic p-values.

For instance, the first three observations yield the following results:

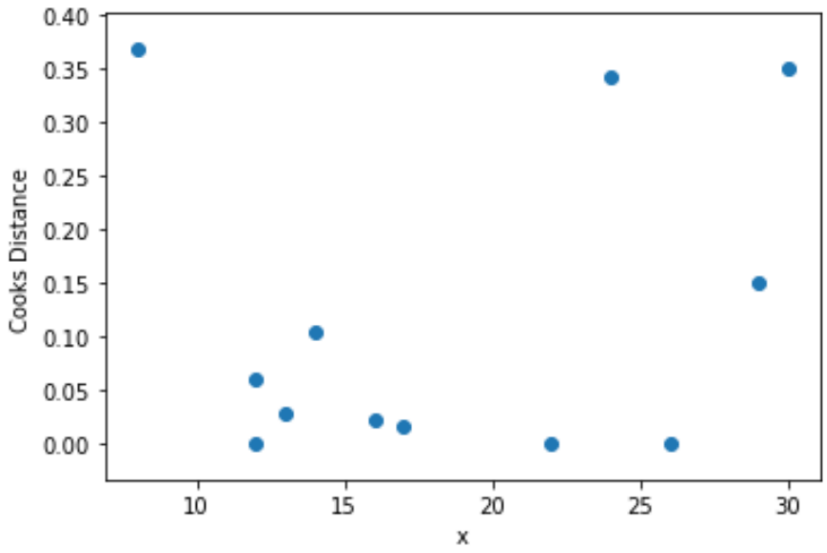
Cook's distance for observation #1: **0.368** (p-value: 0.701)
Cook's distance for observation #2: **0.061** (p-value: 0.941)
Cook's distance for observation #3: **0.001** (p-value: 0.999)

Step 4: Visualizing Cook's Distances

To gain a clearer perspective on which observations are most influential, it is useful to visualize the results. We can create a scatterplot to compare the predictor variable (x) against the calculated Cook's distance for each point using the **Matplotlib** library in **Python**.

```
import matplotlib.pyplot as plt

plt.scatter(df.x, cooks)
plt.xlabel('x')
plt.ylabel('Cooks Distance')
plt.show()
```



The resulting visualization makes it easy to spot observations that have a disproportionately high Cook's distance, indicating areas of high influence on the **regression model**.

Conclusion: Handling Influential Observations

It is crucial to note that **Cook's distance** should be used strictly as a diagnostic tool to identify potentially influential observations, not as an automatic trigger for deletion. Just because a data point is influential does not necessarily mean it is erroneous or should be removed from the dataset.

Before taking any action, you should first investigate the influential observation. Verify that the value is not the result of a data entry error, a measurement anomaly, or some other technical oddity.

If the observation proves to be a legitimate, though unusual, real-world value, you can then make an informed decision on how to proceed: whether it is appropriate to delete it, retain it and acknowledge its influence, or possibly replace it with a more robust statistic, such as the median value. This careful approach ensures the integrity and statistical validity of the final model.