How to calculate VIF in Python?

Authored by

stats writer

December 24, 2025

The reliability and interpretability of a regression model hinge critically on the independence of its predictor variables. A common statistical challenge that undermines this independence is multicollinearity. This phenomenon occurs when two or more explanatory variables in the model are highly correlated with one another, meaning they effectively provide redundant or non-unique information to the estimation process. Identifying and quantifying this correlation is essential for robust statistical modeling, and the primary tool for this purpose is the **Variance Inflation Factor (VIF)**, a metric that quantifies the severity of multicollinearity in ordinary least squares regression.

Calculating the **VIF** is a standard procedure in econometric and data science workflows. Fortunately, the statsmodels library in Python offers robust tools, specifically the `variance_inflation_factor` function, which streamlines this calculation. This tutorial will provide an expert walkthrough on implementing this function, demonstrating how to prepare your data, calculate VIF for each predictor, and accurately interpret the resulting values to ensure the validity of your regression findings.

Understanding the Concept of Multicollinearity

Multicollinearity literally means "multiple correlations." When independent variables are correlated, the resulting regression coefficients become unstable and highly sensitive to small changes in the model or the data set. Imagine trying to model house price using both the square footage of the living area and the total square footage of the house; these two variables are inherently linked, and including both introduces severe multicollinearity. If the correlation is perfect (perfect multicollinearity), the regression model cannot be estimated at all because the matrix inversion required for coefficient estimation becomes impossible.

Even if the correlation is not perfect (high, but not perfect, multicollinearity), the statistical consequences can be severe. The standard errors of the affected coefficients inflate dramatically--hence the term **Variance Inflation Factor**. Large standard errors lead to smaller t-statistics and higher p-values, making it difficult to reject the null hypothesis even when the predictor variable is truly important. Furthermore, high multicollinearity makes it nearly impossible to isolate the unique effect of any single correlated predictor variable on the response variable, severely complicating the interpretation of the model results.

Therefore, before finalizing any multiple linear regression, practitioners must assess the degree of correlation among the predictors. While simple correlation matrices can detect bivariate relationships, **VIF** is superior because it measures how much the variance of an estimated regression coefficient is increased due to collinearity with all other predictor variables in the model simultaneously. It provides a single, easy-to-interpret numerical metric for this complex multivariate issue.

Calculating VIF: A Python Approach

To systematically calculate the **Variance Inflation Factor** in a programming environment like Python, we leverage specialized statistical libraries. The statsmodels package is the industry standard for this task, containing the necessary functions to analyze the influence of individual predictors. Specifically, we utilize `variance_inflation_factor`, which requires the design matrix (the matrix of predictor variables) and the index of the specific predictor for which the VIF is being calculated.

The fundamental mathematical definition of VIF for a predictor X_k is $VIF_k = 1 / (1 - R^2_k)$, where R^2_k is the coefficient of determination obtained from regressing X_k on all the other explanatory variables in the model. This means that to calculate VIF for a specific variable, we must first run an auxiliary regression where that variable acts as the response, and all other predictors act as explanatory variables. The `statsmodels` function simplifies this process significantly by handling these auxiliary regressions internally.

For efficient calculation across all predictors, we typically need to iterate through every column (predictor) in our design matrix, calculate the VIF for that column, and store the result. Before we can apply the VIF function, however, we must structure our data appropriately, often involving the use of the `patsy` library to construct the necessary design matrices, as demonstrated in the practical example below.

Example: Setting Up the Dataset in Python

To illustrate the VIF calculation, we will work with a synthetic dataset created using the pandas and numpy libraries. This dataset contains attributes for ten hypothetical basketball players, where we intend to model a player's overall rating based on their statistical contributions: points, assists, and rebounds. The goal is to determine if any of these performance metrics exhibit significant collinearity before proceeding with the main regression analysis.

We begin by importing the necessary libraries and constructing the dataframe. The data definition step is crucial as it defines the raw input upon which our statistical tests will operate. Note that in this example, 'rating' will be our response variable, while 'points', 'assists', and 'rebounds' are the explanatory variables whose VIF values we seek to quantify.

The following code snippet demonstrates the creation and immediate display of this initial dataframe, confirming the structure of our data before we proceed to model setup:

import numpy as np
import pandas as pd

#create dataset
df = pd.DataFrame({'rating': ,
'points': ,
'assists': ,
'rebounds': })

#view dataset
df

rating points assists rebounds
0 90 25 5 11
1 85 20 7 8
2 82 14 7 10
3 88 16 8 6
4 94 27 5 6
5 90 20 7 9
6 76 12 6 6
7 75 15 9 10
8 87 14 9 10
9 86 19 5 7

Implementing VIF Calculation using Statsmodels

Our objective is to fit a multiple linear regression model where the 'rating' is the dependent variable predicted by 'points', 'assists', and 'rebounds'. Before the final model fitting, we must isolate the design matrix (the predictors) and apply the VIF calculation. We utilize the `dmatrices` function from the `patsy` library, which is excellent for creating design matrices from formula strings, ensuring that constants (like the intercept term) are correctly included.

The `dmatrices` function processes the formula string `'rating ~ points + assists + rebounds'`, separating the response variable (y) from the explanatory variables (X), including an intercept term automatically in the design matrix X. Once we have the matrix X, we iterate through its columns. For each column index `i`, we call `variance_inflation_factor(X.values, i)` to compute the specific VIF value for that predictor.

The resulting VIF values, along with their corresponding variable names, are then organized into a new pandas DataFrame for clear visualization and subsequent interpretation. This systematic approach ensures that every predictor's level of collinearity is accurately assessed against all others in the model.

```
from patsy import dmatrices
from statsmodels.stats.outliers_influence import variance_inflation_factor

#find design matrix for linear regression model using 'rating' as response variable
y, X = dmatrices('rating ~ points+assists+rebounds', data=df, return_type='dataframe')

#calculate VIF for each explanatory variable
vif = pd.DataFrame()
vif = )]
vif = X.columns

#view VIF for each explanatory variable
vif
```

```
VIF variable
0 101.258171 Intercept
1 1.763977 points
2 1.959104 assists
3 1.175030 rebounds
```

Analyzing the Calculated VIF Results

Upon executing the VIF calculation script, we obtain the VIF scores displayed in the resulting table. These scores quantify the extent to which the variance of each coefficient is inflated due to its linear relationship with the other predictors. A VIF value close to 1 indicates near-zero collinearity, while higher values signify increasing degrees of correlation.

Observing the results, the VIF values for our key explanatory variables are:

points: 1.76
assists: 1.96
rebounds: 1.18

It is important to acknowledge the presence of the 'Intercept' term in the results, which often exhibits a very high VIF score (101.26 in this instance). According to best practices in statistical modeling, the VIF value associated with the model intercept should generally be disregarded, as it is largely meaningless in the context of assessing predictor collinearity. Our focus remains solely on the predictor variables: points, assists, and rebounds.

The scores derived for the basketball metrics are all relatively low, suggesting that these variables provide independent information to the regression model. To properly contextualize these scores,

however, we must refer to established guidelines for VIF interpretation, which define thresholds for acceptable and unacceptable levels of collinearity.

Expert Guidelines for VIF Interpretation

The **Variance Inflation Factor** starts at a minimum value of 1 and theoretically has no upper limit. The closer the VIF score is to 1, the less correlated the predictor is with its counterparts, indicating highly stable coefficient estimates. Conversely, as the VIF score increases, the degree of multicollinearity becomes more pronounced, leading to variance inflation and instability.

While there is no universally accepted critical threshold, statisticians commonly rely on a set of rules of thumb to classify the severity of collinearity. These guidelines help practitioners decide whether remediation steps are necessary to stabilize the model coefficients. The standard interpretation thresholds are:

A VIF of 1 indicates **zero correlation**. This is the ideal scenario where the variance of the coefficient is not inflated by collinearity.
A VIF between 1 and 5 suggests **moderate correlation**. While some researchers suggest attention here, these values are typically acceptable for most regression models and do not usually warrant intervention unless model interpretation is extremely sensitive.
A VIF greater than 5 signals **potentially severe multicollinearity**. When a predictor's VIF exceeds this threshold, the resulting coefficient estimates and associated p-values should be treated with extreme caution, as they are likely unreliable, unstable, and highly susceptible to sampling error.

In certain complex fields, such as econometrics, a higher threshold (sometimes up to 10) may occasionally be tolerated, but a score exceeding 5 is generally considered a strong warning sign. Given that each of the VIF values for the explanatory variables in our regression model are close to 1, multicollinearity is not a problem in our example.

Remedial Actions for High VIF Scores

Although our example dataset demonstrated low VIF scores, it is crucial to understand the necessary corrective measures when high multicollinearity is detected. If one or more predictor variables exhibit VIF scores significantly above the acceptable limit (e.g., > 5 or > 10), several strategies can be employed to mitigate the issue and restore stability to the regression analysis.

One common solution is to **remove the highly correlated variable(s)**. If two variables measure essentially the same concept, removing one often resolves the collinearity without significant loss of explanatory power. Another approach involves **combining the correlated variables** into a single composite variable or index. For instance, if 'age' and 'years of experience' are highly correlated, they might be averaged or transformed into a single latent factor. Furthermore,

collecting more data can sometimes reduce the severity of multicollinearity, as larger sample sizes often lead to better estimates and smaller standard errors.

Advanced techniques such as **Principal Component Analysis (PCA)** or **Ridge Regression** offer mathematical ways to deal with high VIF. PCA transforms the correlated variables into a new set of orthogonal (uncorrelated) variables, which can then be used in the regression. Ridge Regression introduces a slight bias to the coefficient estimates to drastically reduce the variance inflation, making the model more stable and reliable when faced with severe multicollinearity. Selecting the appropriate remediation depends heavily on the context of the data and the specific goals of the statistical analysis.