Variance Inflation Factor (VIF) Explained

08 March 2017 / Python / Data Wrangling

Colinearity is the state where two variables are highly correlated and contain similiar information about the variance within a given dataset. To detect colinearity among variables, simply create a correlation matrix and find variables with large absolute values. In R use the `corr` (http://www.sthda.com/english/wiki/correlation-matrix-a-quick-start-guide-to-analyze-format-and-visualize-a-correlation-matrix-using-r-software) function and in python this can by accomplished by using numpy's `corrcoef` (https://docs.scipy.org/doc/numpy/reference/generated/numpy.corrcoef.html) function.

Multicolinearity (https://en.wikipedia.org/wiki/Multicollinearity) on the other hand is more troublesome to detect because it emerges when three or more variables, which are highly correlated, are included within a model. To make matters worst multicolinearity can emerge even when isolated pairs of variables are not colinear.

A common R function used for testing regression assumptions and specifically multicolinearity is "VIF()" and unlike many statistical concepts, its formula is straightforward:

$$ V.I.F. = 1 / (1 - R^2). $$

The Variance Inflation Factor (VIF) is a measure of colinearity among predictor variables within a multiple regression. It is calculated by taking the the ratio of the variance of all a given model's betas divide by the variane of a single beta if it were fit alone.

Steps for Implementing VIF

1. Run a multiple regression.
2. Calculate the VIF factors.
3. Inspect the factors for each predictor variable, if the VIF is between 5-10, multicolinearity is likely present and you should consider dropping the variable.

```
#Imports
import pandas as pd
import numpy as np
from patsy import dmatrices
import statsmodels.api as sm
from statsmodels.stats.outliers_influence import variance_inflation_factor

df = pd.read_csv('loan.csv')
df.dropna()
df = df._get_numeric_data() #drop non-numeric cols

df.head()
```

	id	member_id	loan_amnt	funded_amnt	funded_amnt_inv	int_rate	installment	annual_inc	dti	delinq_2yrs	...	total_bal_il	il_util	open_rv_12m	open_rv_24m	max_bal_bc	all_util	total_rev_hi_lim	inq_fi	total_cu_tl	inq_last_12m
0	1077501	1296599	5000.0	5000.0	4975.0	10.65	162.87	24000.0	27.65	0.0	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
1	1077430	1314167	2500.0	2500.0	2500.0	15.27	59.83	30000.0	1.00	0.0	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2	1077175	1313524	2400.0	2400.0	2400.0	15.96	84.33	12252.0	8.72	0.0	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
3	1076863	1277178	10000.0	10000.0	10000.0	13.49	339.31	49200.0	20.00	0.0	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
4	1075358	1311748	3000.0	3000.0	3000.0	12.69	67.79	80000.0	17.94	0.0	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

5 rows × 51 columns

```
df = df[['annual_inc','loan_amnt', 'funded_amnt','annual_inc','dti']].dropna() #subset the dat
```

Step 1: Run a multiple regression

```
%%capture
#gather features
features = "+".join(df.columns - ["annual_inc"])

# get y and X dataframes based on this regression:
y, X = dmatrices('annual_inc ~' + features, df, return_type='dataframe')
```

Step 2: Calculate VIF Factors

```
# For each X, calculate VIF and save in dataframe
vif = pd.DataFrame()
vif["VIF Factor"] = [variance_inflation_factor(X.values, i) for i in range(X.shape[1])]
vif["features"] = X.columns
```

Step 3: Inspect VIF Factors

```
vif.round(1)
```

	VIF Factor	features
0	5.1	Intercept
1	1.0	dti
2	678.4	funded_amnt
3	678.4	loan_amnt

As expected, the total funded amount for the loan and the amount of the loan have a high variance inflation factor because they "explain" the same variance within this dataset. We would need to discard one of these variables before moving on to model building or risk building a model with high multicolinearity.

Find an error or bug? Have a suggestion?

Everything on this site is avaliable on GitHub. Head on over and submit an issue. (https://github.com/etav/) You can also message me directly on Twitter (https://twitter.com/etav).

This project contains 16 pages and is available on GitHub (https://github.com/etav/).