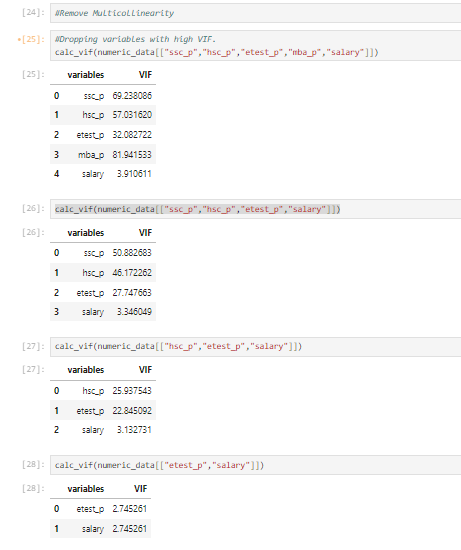
**Ways to Remove Multicollinearity**

One effective method to **remove multicollinearity**: **dropping variables with high VIF**.

That’s usually the first and most direct approach.



Here are **other common ways** to deal with multicollinearity (beyond just dropping variables):

**1. Principal Component Analysis (PCA)**

* PCA transforms your original correlated variables into **a smaller set of uncorrelated components** (principal components).
* Use these components in regression instead of the original variables.

|  |
| --- |
| **from sklearn.decomposition import PCA**  **from sklearn.preprocessing import StandardScaler**  **# Standardize the data first**  **scaler = StandardScaler()**  **X\_scaled = scaler.fit\_transform(X)**  **# Apply PCA**  **pca = PCA()**  **X\_pca = pca.fit\_transform(X\_scaled)** |

**Drawback**: The resulting components are **not interpretable** like the original variables.

### 2. ****Regularization (Ridge or Lasso Regression)****

* These techniques add a penalty term to the regression to reduce the effect of multicollinearity.

**Ridge Regression** is especially good at handling multicollinearity:

|  |
| --- |
| **from sklearn.linear\_model import Ridge**  **ridge = Ridge(alpha=1.0)**  **ridge.fit(X, y)** |

**Lasso** also helps by shrinking coefficients, and can even eliminate some features:

|  |
| --- |
| **from sklearn.linear\_model import Lasso**  **lasso = Lasso(alpha=0.1)**  **lasso.fit(X, y)** |

### 3. ****Combine Correlated Variables****

* If two or more variables are highly correlated, you can **combine them** into a single feature (e.g., taking their mean).

Example:

|  |
| --- |
| **X['combined\_score'] = (X['ssc\_p'] + X['hsc\_p']) / 2**  **X = X.drop(['ssc\_p', 'hsc\_p'], axis=1)** |

### 4. ****Centering the Variables****

* Centering (subtracting the mean) doesn't remove multicollinearity but can help **reduce** it when it's due to interaction terms.

|  |
| --- |
| **X\_centered = X - X.mean()** |

### 5. ****Domain Knowledge****

* Use business or domain knowledge to decide which variable to keep or drop instead of blindly relying on VIF.

### 6. ****Use Partial Least Squares Regression (PLS)****

* PLS is similar to PCA but is **supervised** and considers the target variable while reducing dimensions.

|  |
| --- |
| **from sklearn.cross\_decomposition import PLSRegression**  **pls = PLSRegression(n\_components=2)**  **pls.fit(X, y)** |

****