**What Is Multicollinearity?**

**Multicollinearity** happens when **two or more independent variables (features)** in a regression model are **highly correlated** with each other.

➡️ In plain English:  
When one feature can be **predicted almost perfectly from another**, they carry **duplicate information** — confusing the model.

**⚙️ Why It Matters**

High multicollinearity can cause:

1. ❌ **Unstable coefficients** — small data changes → large coefficient swings.
2. ❌ **Inflated standard errors** — making p-values unreliable.
3. ❌ **Difficulty interpreting model** — can’t tell which variable really affects the target.

**📊 How to Detect Multicollinearity**

**1️⃣ Correlation Matrix**

Use .corr() to spot correlated features:

corr = dataset.corr()

print(corr)

If correlation > **0.8** or < **-0.8**, it’s a warning sign.

**2️⃣ Variance Inflation Factor (VIF)**

VIF quantifies how much a variable is inflated because of multicollinearity.

Formula:  
[  
VIF = \frac{1}{1 - R^2}  
]

Interpretation:

| **VIF Value** | **Meaning** |
| --- | --- |
| **1–5** | Moderate, acceptable |
| **>5** | Potential multicollinearity |
| **>10** | Serious multicollinearity problem |

Example using your function:

vif\_data = calculate\_vif(x\_input)

print(vif\_data)

**🩺 How to Fix Multicollinearity**

1. **Drop one of the correlated features**
   * If A and B are highly correlated, remove one.
2. **Combine correlated variables**
   * Example: create an average or ratio feature.
3. **Use PCA (Principal Component Analysis)**
   * Converts correlated features into uncorrelated components.
4. **Regularization (Ridge or Lasso Regression)**
   * Adds penalty to reduce coefficient variance.

**✅ Example Summary Table**

| **Feature** | **VIF** | **Interpretation** |
| --- | --- | --- |
| Age | 2.1 | OK |
| Income | 8.3 | Possible issue |
| Experience | 15.6 | High multicollinearity (check/remove) |

**🧩 Key Takeaways**

* **VIF > 10 → strong multicollinearity**
* **Drop or transform** variables to improve model stability.
* Always check **correlation** and **VIF** before fitting regression models.

**How it works:**

vif\_data = calculate\_vif(x\_input)

print(vif\_data)

🩺 How to Fix Multicollinearity

* Drop one of the correlated features
* If A and B are highly correlated, remove one.
* Combine correlated variables
* Example: create an average or ratio feature.
* Use PCA (Principal Component Analysis)
* Converts correlated features into uncorrelated components.
* Regularization (Ridge or Lasso Regression)
* Adds penalty to reduce coefficient variance.

✅ Example Summary Table

Feature VIF Interpretation

Age 2.1 OK

Income 8.3 Possible issue

Experience 15.6 High multicollinearity (check/remove)

🧩 Key Takeaways

VIF > 10 → strong multicollinearity

Drop or transform variables to improve model stability.

Always check correlation and VIF before fitting regression models.

Would you like me to show a code example that automatically removes high-VIF columns (e.g., VIF > 10) and returns a cleaned dataset?

**Calculates VIF for each feature**

Automatically removes columns with VIF > 10

Returns a clean dataset and a summary table

//Full Working Code

import pandas as pd

from statsmodels.stats.outliers\_influence import variance\_inflation\_factor

from statsmodels.tools.tools import add\_constant

class VIFHandler:

def \_\_init\_\_(self, dataframe):

self.dataframe = dataframe.copy()

def calculate\_vif(self, data=None):

"""Calculate Variance Inflation Factor (VIF) for each column"""

if data is None:

data = self.dataframe

X = add\_constant(data, has\_constant='add')

vif = pd.DataFrame()

vif["Feature"] = data.columns

vif["VIF"] = [

variance\_inflation\_factor(X.values, i + 1)

for i in range(len(data.columns))

]

return vif

def remove\_high\_vif(self, threshold=10.0):

"""

Iteratively remove features with VIF higher than the threshold.

Returns cleaned data and a summary of dropped columns.

"""

data = self.dataframe.copy()

dropped = []

while True:

vif\_data = self.calculate\_vif(data)

max\_vif = vif\_data["VIF"].max()

if max\_vif > threshold:

max\_feature = vif\_data.loc[vif\_data["VIF"].idxmax(), "Feature"]

print(f"⚠️ Dropping '{max\_feature}' with VIF={max\_vif:.2f}")

dropped.append((max\_feature, max\_vif))

data = data.drop(columns=[max\_feature])

else:

break

print("\n✅ Final Features Remaining:")

print(data.columns.tolist())

summary = pd.DataFrame(dropped, columns=["Dropped\_Feature", "VIF\_Value"])

return data, summary

🧩 Example Usage

python

Copy code

# Example dataset

data = pd.DataFrame({

"A": [1, 2, 3, 4, 5],

"B": [2, 4, 6, 8, 10],

"C": [5, 3, 6, 9, 12],

"D": [2, 5, 7, 9, 11]

})

vif\_handler = VIFHandler(data)

# Step 1: See initial VIFs

print("Initial VIF values:")

print(vif\_handler.calculate\_vif())

# Step 2: Remove high-VIF columns

clean\_data, removed\_summary = vif\_handler.remove\_high\_vif(threshold=10)

print("\nRemoved columns:")

print(removed\_summary)

print("\nCleaned dataset:")

print(clean\_data)

\*\* Output Example

sql

Copy code

Initial VIF values:

Feature VIF

0 A 30.2159

1 B 45.3212

2 C 8.4123

3 D 3.2123

⚠️ Dropping 'B' with VIF=45.32

✅ Final Features Remaining:

['A', 'C', 'D']

Removed columns:

Dropped\_Feature VIF\_Value

0 B 45.3212

💡 Key Benefits

* 🚀 Automatic detection and removal of high-VIF columns
* 🧮 Provides a summary of what was removed and why
* 🧹 Returns a clean DataFrame ready for regression or modeling