# **1. Remove One of the Correlated Features (Basic Approach)**

If two variables are highly correlated, we can manually drop one.

### **Example:**

corr\_matrix = df.corr()

print(corr\_matrix["salary"].sort\_values(ascending=False))

* salary and mba\_p → correlation = 0.92  
   Then keep the one and drop the other

df.drop("mba\_p", axis=1, inplace=True)

# **2. Combine Correlated Features (Feature Engineering)**

Instead of dropping, combine them.

### **Example:**

If ssc\_p and hsc\_p are correlated:

df["academic\_avg"] = (df["ssc\_p"] + df["hsc\_p"]) / 2

df.drop(["ssc\_p", "hsc\_p"], axis=1, inplace=True)

This keeps the information while reducing collinearity.

# **3. Use Regularization (Ridge, Lasso, Elastic Net)**

Multicollinearity affects **linear regression**, but **Ridge and Lasso** handle it automatically.

### **Ridge Regression Example:**

from sklearn.linear\_model import Ridge

model = Ridge(alpha=1)

model.fit(X, y)

Ridge reduces the impact of correlated features instead of deleting them.

### **Lasso Regression Example (Feature Selection):**

from sklearn.linear\_model import Lasso

model = Lasso(alpha=0.01)

model.fit(X, y)

Lasso makes some coefficients **zero** → automatically removes features.

# **4. Principal Component Analysis (PCA)**

Instead of using original correlated columns, you create **new independent components**.

### **Example:**

from sklearn.decomposition import PCA

pca = PCA(n\_components=2)

X\_pca = pca.fit\_transform(df[["ssc\_p", "hsc\_p", "degree\_p"]])

Now you use X\_pca instead of the original features.

# **5. Drop Features Based on Domain Knowledge**

If you know which feature is more meaningful, drop the other.

### **Example:**

* ssc\_p and hsc\_p are highly correlated.
* Domain expert says: Keep hsc\_p, drop ssc\_p.

df.drop("ssc\_p", axis=1, inplace=True)

# **6. Use Tree-Based Models (They are immune to multicollinearity)**

Decision Trees, Random Forest, and XGBoost are not affected by multicollinearity.

### **Example:**

from sklearn.ensemble import RandomForestRegressor

model = RandomForestRegressor()

model.fit(X, y)

No need to drop features or calculate VIF for these models.