Ways to handle multicollinearity

🔧 1. Remove Highly Correlated Features

- Use the **correlation matrix** to identify highly correlated pairs (e.g., correlation > 0.8 or < -0.8).
- Drop one of them to reduce redundancy.

```python
import seaborn as sns
import matplotlib.pyplot as plt

corr_matrix = X.corr()
sns.heatmap(corr_matrix, annot=True, cmap="coolwarm")
plt.show()
```

🧮 2. Use Variance Inflation Factor (VIF)

- Calculate VIF for each feature.
- If VIF > 5 (or 10), consider removing or transforming that variable.

```python
from statsmodels.stats.outliers_influence import variance_inflation_factor

# VIF requires a constant
from statsmodels.tools.tools import add_constant
X_with_const = add_constant(X)

vif = pd.DataFrame()
vif["Variable"] = X.columns
vif["VIF"] = [variance_inflation_factor(X_with_const.values, i + 1) for i in range(X.shape[1])]
print(vif)
```

🔁 3. Principal Component Analysis (PCA)

- PCA transforms correlated features into **uncorrelated components**.
- Useful when you want to keep most of the information with fewer features.

```python
from sklearn.decomposition import PCA

pca = PCA(n_components=2)
X_pca = pca.fit_transform(X)

print("Explained Variance Ratio:", pca.explained_variance_ratio_)
```

🧠 4. Regularization Techniques

- Use models that **handle multicollinearity automatically** by penalizing large coefficients:
  - **Ridge Regression**: L2 penalty (shrinks coefficients)
  - **Lasso Regression**: L1 penalty (can shrink some coefficients to zero)

```python
from sklearn.linear_model import Ridge, Lasso

model = Ridge(alpha=1.0)  # or Lasso(alpha=0.1)
model.fit(X, y)
```

📉 5. Combine Features

- Create a **new feature** that combines two or more highly correlated features (e.g., average or weighted sum).

```python
import pandas as pd
import numpy as np

# Create correlated data
np.random.seed(0)
X1 = np.random.rand(100)
X2 = X1 * 0.9 + np.random.rand(100) * 0.1  # Highly correlated
X3 = np.random.rand(100)
y = 3 * X1 + 2 * X3 + np.random.rand(100)

df = pd.DataFrame({'X1': X1, 'X2': X2, 'X3': X3, 'y': y})
```

📋 6. Domain Knowledge

- Use your **understanding of the problem** to choose the most meaningful features.
- Sometimes two variables are correlated, but one is clearly more relevant to the target.

Scenario	Action
One feature is derived from others	Keep the main ones
One feature is more **interpretable**	Prefer it
Two features are highly correlated but only one is **used in practice**	Drop the other
Units overlap (e.g., km & miles)	Convert or choose one

❌ 7. Drop One of the Variables

- If you don't want to transform or combine features, **just drop** one from each highly correlated pair.

```python
X_reduced = X.drop(columns='X2')  # Drop either X1 or X2
```

🧠 8. Ridge Regression (handles multicollinearity)

Ridge Regression is a type of **regularized linear regression**.
It's used when:

- You have **multicollinearity** (high correlation between independent variables)
- You want to **keep all features**, but **stabilize** their coefficients

```python
from sklearn.linear_model import Ridge
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error

X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=0)

ridge = Ridge(alpha=1.0)
ridge.fit(X_train, y_train)
y_pred = ridge.predict(X_test)

print("Ridge MSE:", mean_squared_error(y_test, y_pred))
```

Method	When to Use
Drop features	Simple, fast fix when two variables are highly correlated
PCA	When you want to reduce dimensionality and remove all correlation
Ridge Regression	When you want to keep all variables but prevent unstable coefficients

---

📉 Recap of All Options

Method	Removes Variables?	Keeps Interpretability?	Handles Automatically?
Drop Correlated Features	✅ Yes	✅ Yes	❌ No
PCA / PLS	✅ Yes	❌ No	✅ Yes
Ridge / Lasso	❌ Sometimes	✅ Yes (Ridge) / ⚠️ (Lasso)	✅ Yes
Tree-Based Models	❌ No	❌ No	✅ Yes
Feature Selection (RFE, etc.)	✅ Yes	✅ Yes	✅ Yes
Dummy Trap Fix (`drop_first=True`)	✅ Yes	✅ Yes	✅ Yes
Standardization	❌ No	✅ Yes	✅ Yes
Bayesian Regression	❌ No	✅ Yes	✅ Yes