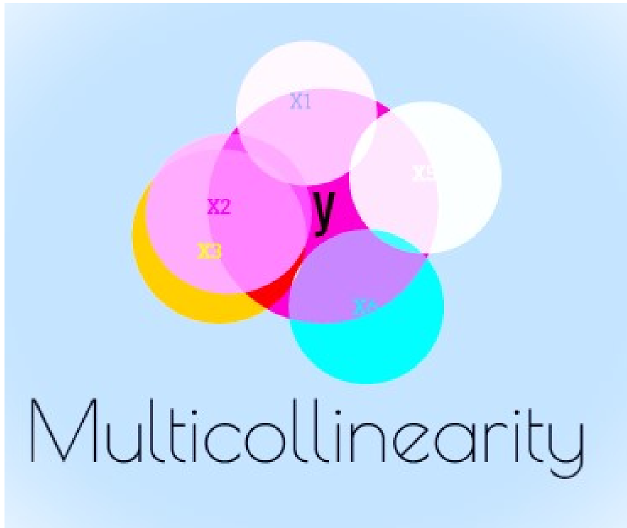
What is Multicollinearity ?

Multicollinearity is a statistical phenomenon that occurs when two or more independent variables in a multiple regression model are **highly correlated**. In other words, these variables exhibit a strong linear relationship, making it difficult to isolate the individual effects of each variable on the dependent variable.



When is Multicollinearity bad?

1. **Inference**:

- Inference focuses on understanding the relationships between the variables in a model. It aims to draw conclusions about the underlying population or process that generated the data.
- Inference often involves hypothesis testing, confidence intervals, and determining the significance of predictor variables.
- The primary goal is to provide insights about the structure of the data and the relationships between variables.
- Interpretability is a key concern when performing inference, as the objective is to understand the underlying mechanisms driving the data.

Examples of inferential techniques include linear regression, logistic regression, and ANOVA.

2. **Prediction:**

- Prediction focuses on using a model to make accurate forecasts or estimates for new, unseen data.
- It aims to generalize the model to new instances, based on the patterns observed in the training data.
- Prediction often involves minimizing an error metric, such as mean squared error or cross- entropy loss, to assess the accuracy of the model.
- The primary goal is to create an accurate and reliable model for predicting outcomes, rather than understanding the relationships between variables.
- Interpretability may be less important in predictive modelling, as the main objective is to create accurate forecasts rather than understanding the underlying structure of the data.

In summary, inference focuses on understanding the relationships between variables and interpreting the underlying structure of the data, while prediction focuses on creating accurate forecasts for new, unseen data based on the patterns observed in the training data.

100 → cgpa | iq | lpa

$\begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \end{bmatrix}$

Finance → banking → model

Customer data → loan

user activity → ban/allow

$x_1 \mid x_2 \mid y$

$[x_1 = a_0 + a_1 x_2 + \eta]$ error

$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \epsilon$ reducible error

$$Y = \beta_0 + \beta_1 (a_0 + a_1 x_2 + \eta) + \epsilon$$

$$y = \beta_0 + \beta_1 a_0 + \beta_1 a_1 x_2 + \beta_1 \eta + \epsilon$$

$$\rightarrow y = (\beta_0 + \beta_1 a_0) + \beta_1 a_1 x_2 + (\beta_1 \eta + \epsilon)$$

$\boxed{Y \rightarrow x_2}$ X_1

optimal → $\boxed{y = c_0 + c_1 x_2 + \gamma}$

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.033
Model:                            OLS   Adj. R-squared:                  0.013
Method:                 Least Squares   F-statistic:                     1.643
Date:                Mon, 08 May 2023   Prob (F-statistic):              0.199
Time:                        11:07:58   Log-Likelihood:                -582.02
No. Observations:                 100   AIC:                             1170.
Df Residuals:                      97   BIC:                             1178.
Df Model:                           2
Covariance Type:            nonrobust
==================================================================================
                     coef    std err          t      P>|t|      [0.025      0.975]
----------------------------------------------------------------------------------
const              5.2059      8.443      0.617      0.539     -11.552      21.963
X1_X2_combined    15.9730      9.594      1.665      0.099      -3.068      35.014
X3                 6.8469      8.512      0.804      0.423     -10.047      23.740
==============================================================================
Omnibus:                        8.244   Durbin-Watson:                   2.077
Prob(Omnibus):                  0.016   Jarque-Bera (JB):                8.431
Skew:                           0.536   Prob(JB):                       0.0148
Kurtosis:                       3.935   Cond. No.                         1.27
```

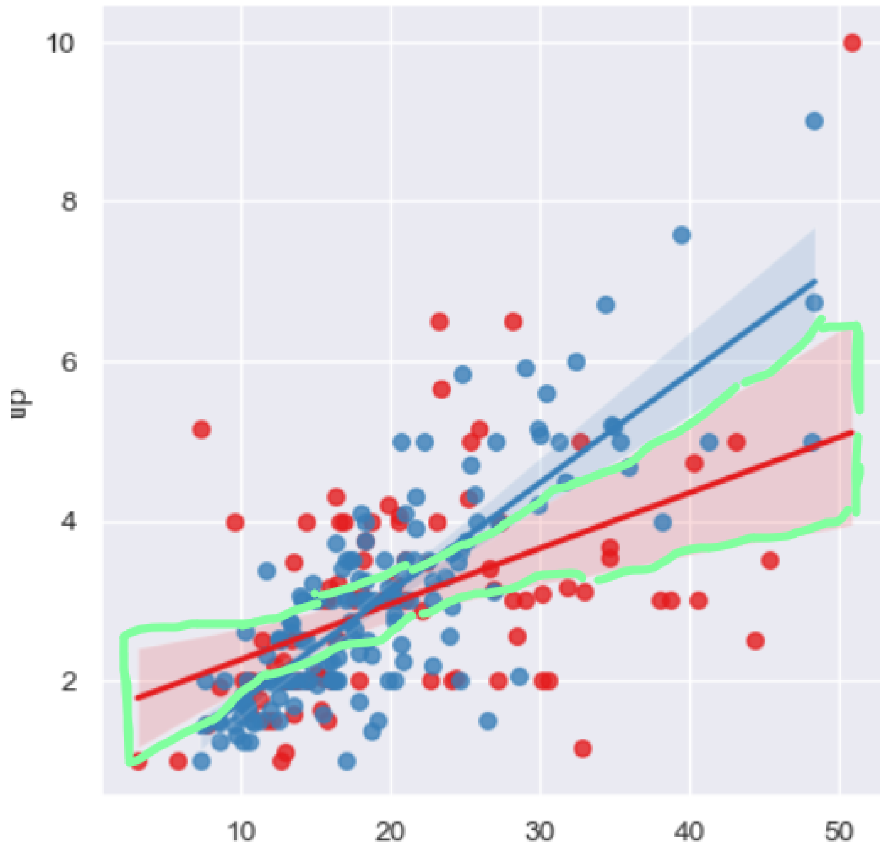
What exactly happens in Multicollinearity(Mathematically?)

- **When multicollinearity is present in a model, it can lead to several issues, including:**

1. **Difficulty in identifying the most important predictors**: Due to the high correlation between independent variables, it becomes challenging to determine which variable has the most significant impact on the dependent variable.

2. **Inflated standard errors**: Multicollinearity can lead to larger standard errors for the regression coefficients, which decreases the statistical power and can make it challenging to determine the true relationship between the independent and dependent variables.

3. **Unstable and unreliable estimates**: The regression coefficients become sensitive to small changes in the data, making it difficult to interpret the results accurately.



In [3]:
```
import numpy as np

# design Matrix for Multi Collinearity

arr = np.array([[1, 1, 1, 1, 1, 1, 1, 1, 1, 1], # x
                [1, 2, 3, 4, 5, 6, 7, 8, 9, 10], # example: CGPA
                [1.2, 2.1, 3.1, 4.1, 5.0, 6.0, 7.0, 8.0, 9.1, 10.2]]).T # example : IQ

arr


# Strong Correlation
```

Out[3]:
```
array([[ 1. ,  1. ,  1.2],
       [ 1. ,  2. ,  2.1],
       [ 1. ,  3. ,  3.1],
       [ 1. ,  4. ,  4.1],
       [ 1. ,  5. ,  5. ],
       [ 1. ,  6. ,  6. ],
       [ 1. ,  7. ,  7. ],
       [ 1. ,  8. ,  8. ],
       [ 1. ,  9. ,  9.1],
       [ 1. , 10. , 10.2]])
```

In [5]:
```
 # Calculate the Beta's inverse of the transpose of X multiplied by X


np.linalg.inv(np.dot(arr.T,arr))
```

Out[5]:
```
array([[  0.67713004,   1.89686099,  -1.97309417],
       [  1.89686099,  18.3309417 , -18.40807175],
       [ -1.97309417, -18.40807175,  18.49775785]])
```

In [6]:
```
# Changed Only One value ( 1 to 1.1)

arr = np.array([[1,1,1,1,1,1,1,1,1,1],
                [1.1, 2, 3, 4, 5, 6, 7, 8, 9, 10], # Here 1.1
                [1.2, 2.1, 3.1, 4.1, 5.0, 6.0, 7.0, 8.0, 9.1, 10.2]]).T


np.linalg.inv(np.dot(arr.T,arr))
```

Out[6]:
```
array([[  0.579715  ,   1.55493605,  -1.62140011],
       [  1.55493605,  24.45470752, -24.42659041],
       [ -1.62140011, -24.42659041,  24.41073715]])
```

In [7]:
```
# Here we can observe, changing only value can impact more, where first we have 18.33, now we have

# Beta's are very sensitive to Input data
```

We took Similar data but without any multi collinearity

In [8]:
```
arr = np.array([[1,1,1,1,1,1,1,1,1,1],
                [1, 2, 3, 4, 5, 6, 7, 8, 9, 10],
                [2.4, 0.7, 4.3, 3.5, 1.6, 5.1, 6.9, 7.5, 8.1, 9.8]]).T

np.linalg.inv(np.dot(arr.T,arr))
```

Out[8]:
```
array([[ 0.46684896, -0.06359872, -0.00341803],
       [-0.06359872,  0.06375331, -0.05752395],
       [-0.00341803, -0.05752395,  0.06408812]])
```

In [9]:
```
arr = np.array([[1,1,1,1,1,1,1,1,1,1],
                [1.1, 2, 3, 4, 5, 6, 7, 8, 9, 10], # changed 1.1
                [2.4, 0.7, 4.3, 3.5, 1.6, 5.1, 6.9, 7.5, 8.1, 9.8]]).T

np.linalg.inv(np.dot(arr.T,arr))
```

Out[9]:
```
array([[ 0.47202303, -0.06705466, -0.00051139],
       [-0.06705466,  0.06552638, -0.05891698],
       [-0.00051139, -0.05891698,  0.06515911]])
```

- Observation : 1 (Observation: 1 (values are small because the determinent is near to 0, which explains why the inverse values are so small. as we divide with determinant)

- Observation : 2(Observation: 2 (when we alter the values 1 to 1.1, the values do not fluctuate)

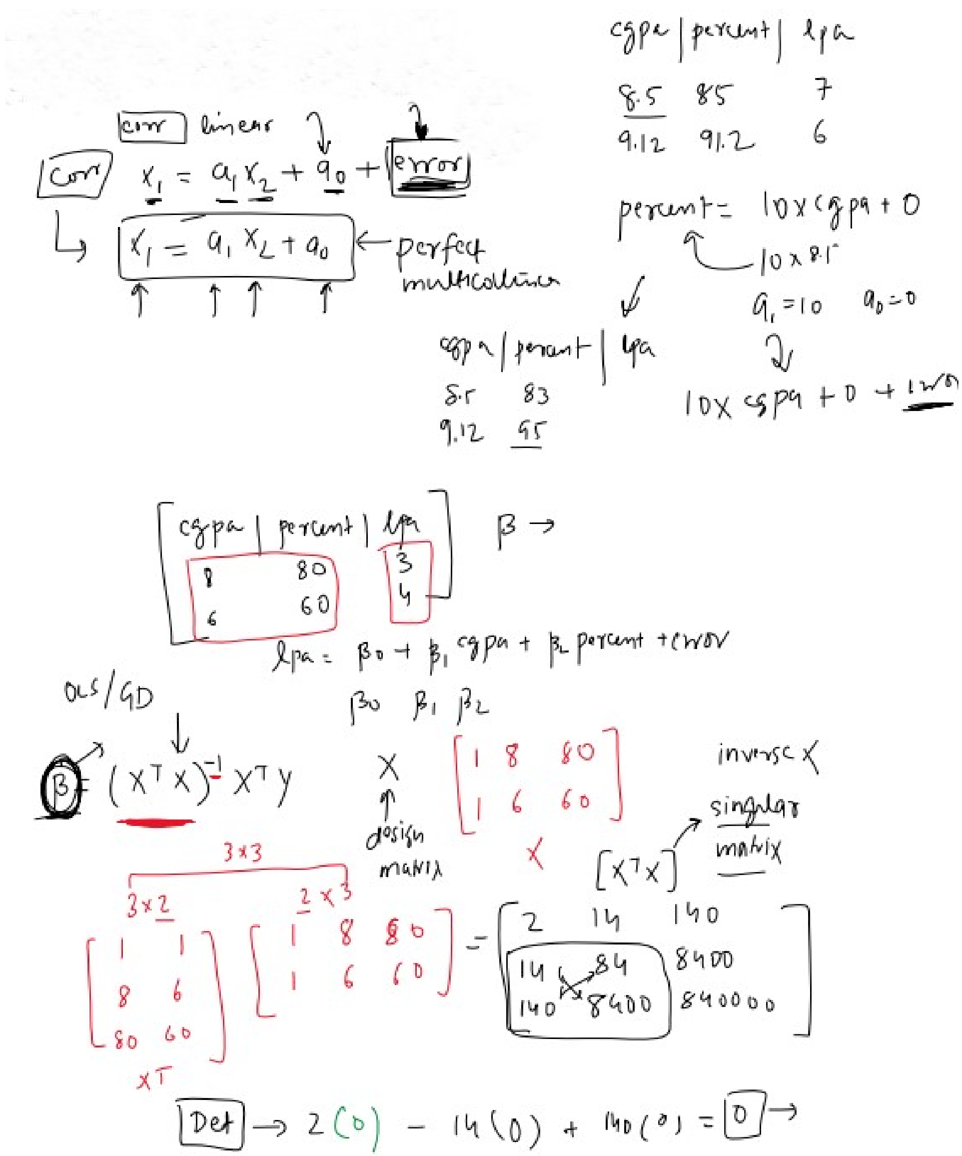
Here we can say , when we have collinearity in data

- we have 2 problems
  - increase of standard Errors
  - Beta will get very sensitive

- Thats Why Multi collinearity fails in **Inference**

Perfect Multicollinearity

Perfect multicollinearity occurs when one independent variable in a multiple regression model is an exact linear combination of one or more other independent variables. In other words, there is an exact linear relationship between the independent variables, making it impossible to uniquely estimate the individual effects of each variable on the dependent variable.



Types of Multicollinearity

1. **Structural multicollinearity**: Structural multicollinearity arises due to the way in which the variables are defined or the model is constructed. It occurs when one independent variable is created as a linear combination of other independent variables or when the model includes interaction terms or higher-order terms (such as polynomial terms) without proper scaling or centering.

2. **Data-driven multicollinearity**: Data-driven multicollinearity occurs when the independent variables in the dataset are highly correlated due to the specific data being analysed. In this case, the high correlation between the variables is not a result of the way the variables are defined or the model is constructed but rather due to the observed data patterns.

How to Detect Multicollinearity

- 1. Correlation
- 2. VIF (Variance Inflation Factor)
- 3. Condition No.

1. Correlation

- Correlation is a measure of the linear relationship between two variables, and it is commonly used to identify multicollinearity in multiple linear regression models. Multicollinearity occurs when two or more predictor variables in the model are highly correlated, making it difficult to determine their individual contributions to the output variable.
- To detect multicollinearity using correlation, you can calculate the correlation matrix of the predictor variables. The correlation matrix is a square matrix that shows the pairwise correlations between each pair of predictor variables. The diagonal elements of the matrix are always equal to 1, as they represent the correlation of a variable with itself. The off-diagonal elements represent the correlation between different pairs of variables.
- In the context of multicollinearity, you should look for off-diagonal elements with high absolute values (e.g., greater than 0.8 or 0.9, depending on the specific application and the level of concern about multicollinearity). High correlation values indicate that the corresponding predictor variables are highly correlated and may be causing multicollinearity issues in the regression model.
- It's important to note that while correlation can be a useful tool for detecting multicollinearity, it doesn't provide a complete picture of the severity of the issue or its impact on the regression model. Other diagnostic measures, such as Variance Inflation Factor (VIF) and condition number, can also be used to assess the presence and severity of multicollinearity in a regression model.

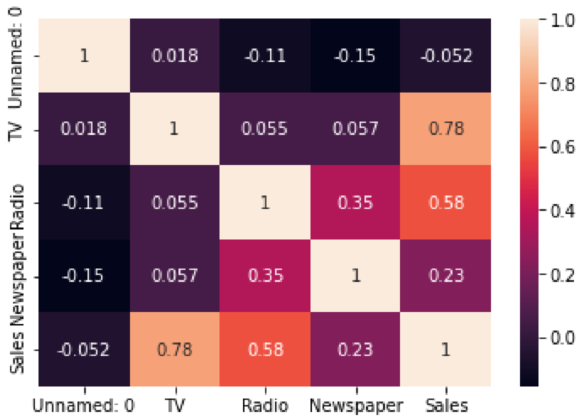
In [11]:
```
# code

import pandas as pd
import seaborn as sns

df = pd.read_csv('https://raw.githubusercontent.com/justmarkham/scikit-learn-videos/master/data/Adv

df.head()
```

Out[11]:

	Unnamed: 0	TV	Radio	Newspaper	Sales
0	1	230.1	37.8	69.2	22.1
1	2	44.5	39.3	45.1	10.4
2	3	17.2	45.9	69.3	9.3
3	4	151.5	41.3	58.5	18.5
4	5	180.8	10.8	58.4	12.9

In [12]:
```
sns.heatmap(df.corr(),annot=True)  # df.corr()
```

Out[12]: <AxesSubplot:>



When the Values are greater than 0.8 or 0.9 can cause Multi Collinearity Problem

2. VIF (Variance Inflation Factor)

Variance Inflation Factor (VIF) is a metric used to quantify the severity of multicollinearity in a multiple linear regression model. It measures the extent to which the variance of an estimated regression coefficient is increased due to multicollinearity.

- For each predictor variable in the regression model, VIF is calculated by performing a separate linear regression using that predictor as the response variable and the remaining predictor variables as the independent variables. The VIF for the predictor variable is then calculated as the reciprocal of the variance explained by the other predictors, which is equal to 1 / (1 - R2). Here, R2 is the coefficient of determination for the linear regression using the predictor variable as the response variable.

The VIF calculation can be summarized in the following steps:

1. For each predictor variable Xi in the regression model, perform a linear regression using Xi as the response variable and the remaining predictor variables as the independent variables.
2. Calculate the R2 value for each of these linear regressions.
3. Compute the VIF for each predictor variable Xi as VIFi = 1 / (1 - R2i).

- A VIF value **close to 1 indicates that there is very little multicollinearity** for the predictor variable, whereas a high VIF value (e.g., greater than 5 or 10, depending on the context) suggests that multicollinearity may be a problem for the predictor variable, and its estimated coefficient might be less reliable.
- Keep in mind that VIF only provides an indication of the presence and severity of multicollinearity and does not directly address the issue. Depending on the VIF values and the goals of the analysis, you might consider using techniques like variable selection, regularization, or dimensionality reduction methods to address multicollinearity.

Formula :

$$VIF = \frac{1}{\qquad}$$

In [15]:
```
# Code

from statsmodels.stats.outliers_influence import variance_inflation_factor

vif = []

for i in range(3):
    vif.append(variance_inflation_factor(df.iloc[:, 1:4].values, i))
```

In [16]:
```
pd.DataFrame({'vif': vif}, index=df.columns[1:4]).T
```

Out[16]:

	TV	Radio	Newspaper
vif	2.486772	3.285462	3.055245

In [17]:
```
# Here all values are Normal , none of columns crossing 5 OR 10 , so there is no Multi Collinearity
```

3. Condition No.

- In the context of multicollinearity, the condition number is a diagnostic measure used to assess the stability and potential numerical issues in a multiple linear regression model. It provides an indication of the severity of multicollinearity by examining the sensitivity of the linear regression to small changes in the input data.
- The condition number is calculated as the ratio of the largest eigenvalue to the smallest eigenvalue of the matrix XTX, where X is the design matrix of the regression model (each row representing an observation and each column representing a predictor variable). A high condition number suggests that the matrix XTX is **ill-conditioned and can lead to numerical instability when solving the normal equations for the regression coefficients**.
- In the presence of multicollinearity, the design matrix X has highly correlated columns, which can cause the eigenvalues of XTX to be very different in magnitude (one or more very large eigenvalues and one or more very small eigenvalues). As a result, the condition number becomes large, indicating that the regression model may be sensitive to small changes in the input data, leading to unstable coefficient estimates.
- Typically, a condition number larger than 30 (or sometimes even larger than 10 or 20) is considered a warning sign of potential multicollinearity issues. However, the threshold for the condition number depends on the specific application and the level of concern about multicollinearity.
- It's important to note that a high condition number alone is not definitive proof of multicollinearity. It is an indication that multicollinearity might be a problem, and further investigation (e.g., using VIF, correlation matrix, or tolerance values) may be required to confirm the presence and severity of multicollinearity.

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.034
Model:                            OLS   Adj. R-squared:                  0.004
Method:                 Least Squares   F-statistic:                     1.122
Date:                Mon, 08 May 2023   Prob (F-statistic):              0.344
Time:                        16:21:33   Log-Likelihood:                -581.96
No. Observations:                 100   AIC:                             1172.
Df Residuals:                      96   BIC:                             1182.
Df Model:                           3
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          5.6185      8.572      0.655      0.514     -11.397      22.634
X1             0.9804     21.591      0.045      0.964     -41.877      43.838
X2            13.9157     18.451      0.754      0.453     -22.710      50.542
X3             6.8796      8.552      0.804      0.423     -10.095      23.854
==============================================================================
Omnibus:                        8.100   Durbin-Watson:                   2.070
Prob(Omnibus):                  0.017   Jarque-Bera (JB):                8.233
Skew:                           0.531   Prob(JB):                       0.0163
Kurtosis:                       3.921   Cond. No.                         4.13
==============================================================================
```

In [18]:
```
# code

A = np.array([[1, 2],
              [3, 4]])

condition_number = np.linalg.cond(arr)
print("Condition Number:", condition_number)
```

```
Condition Number: 18.593103585331082
```

If the codition is Greater than 30 , its Ill conditioned = Have Multi Collinearity Problem

How to remove multicollinearity

1. **Collect more data**: In some cases, multicollinearity might be a result of a limited sample size. Collecting more data, if possible, can help reduce multicollinearity and improve the stability of the model.

2. **Remove one of the highly correlated variables**: If two or more independent variables are highly correlated, consider removing one of them from the model. This step can help eliminate redundancy in the model and reduce multicollinearity. Choose the variable to remove based on domain knowledge, variable importance, or the one with the highest VIF.

In [22]:
```
import numpy as np
import pandas as pd
import statsmodels.api as sm
from sklearn.datasets import make_regression

# Generate a synthetic dataset with multicollinearity
np.random.seed(42)
X, y = make_regression(n_samples=100, n_features=3, noise=0.5, random_state=42)
X[:, 1] = X[:, 0] + 0.5 * np.random.normal(size=100)  # Introduce multicollinearity between column

# Convert data to a pandas DataFrame
data = pd.DataFrame(X, columns=['X1', 'X2', 'X3'])
data['y'] = y

data
```

Out[22]:

	X1	X2	X3	y
0	-0.792521	-0.544164	-0.114736	13.480582
1	0.280992	0.211860	-0.622700	-18.902685
2	0.791032	1.114876	-0.909387	110.450979
3	0.625667	1.387182	-0.857158	-78.162124
4	-0.342715	-0.459791	-0.802277	-35.728094
...	...	...	...	...
95	0.651391	-0.080366	-0.315269	68.841646
96	1.586017	1.734077	-1.237815	183.634164
97	0.010233	0.140761	-0.981509	17.531189
98	-0.234587	-0.232030	-1.415371	-63.202789
99	-0.327662	-0.444956	-0.392108	-125.373405

100 rows × 4 columns

In [23]:
```
data.corr() # Correlation
```

Out[23]:

	X1	X2	X3	y
X1	1.000000	0.882948	-0.048636	0.148108
X2	0.882948	1.000000	-0.054696	0.165352
X3	-0.048636	-0.054696	1.000000	0.071536
y	0.148108	0.165352	0.071536	1.000000

In [24]:
```
# Add a constant term to the predictor variables
data_with_constant_all = sm.add_constant(data[['X1', 'X2', 'X3']])
data_with_constant_reduced = sm.add_constant(data[['X1', 'X3']])

# Create and fit an OLS model using all three predictor variables
model_all = sm.OLS(data['y'], data_with_constant_all).fit()

# Print the summary for the model with all predictors
print("Regression summary for the model with all predictors:")
print(model_all.summary())

# Create and fit an OLS model using only X1 and X3 (removing the highly correlated variable X2)
model_reduced = sm.OLS(data['y'], data_with_constant_reduced).fit()

# Print the summary for the model with reduced predictors (X1 and X3)
print("\nRegression summary for the model with reduced predictors (X1 and X3):")
print(model_reduced.summary())
```

```
Regression summary for the model with all predictors:
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.034
Model:                            OLS   Adj. R-squared:                  0.004
Method:                 Least Squares   F-statistic:                     1.122
Date:                Thu, 22 Jun 2023   Prob (F-statistic):              0.344
Time:                        20:25:56   Log-Likelihood:                -581.96
No. Observations:                 100   AIC:                             1172.
Df Residuals:                      96   BIC:                             1182.
Df Model:                           3
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          5.6185      8.572      0.655      0.514     -11.397      22.634
X1             0.9804     21.591      0.045      0.964     -41.877      43.838
X2            13.9157     18.451      0.754      0.453     -22.710      50.542
X3             6.8796      8.552      0.804      0.423     -10.095      23.854
==============================================================================
Omnibus:                        8.100   Durbin-Watson:                   2.070
Prob(Omnibus):                  0.017   Jarque-Bera (JB):                8.233
Skew:                           0.531   Prob(JB):                       0.0163
Kurtosis:                       3.921   Cond. No.                         4.13
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

Regression summary for the model with reduced predictors (X1 and X3):
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.028
Model:                            OLS   Adj. R-squared:                  0.008
Method:                 Least Squares   F-statistic:                     1.405
Date:                Thu, 22 Jun 2023   Prob (F-statistic):              0.250
Time:                        20:25:56   Log-Likelihood:                -582.25
No. Observations:                 100   AIC:                             1171.
Df Residuals:                      97   BIC:                             1178.
Df Model:                           2
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          4.8245      8.488      0.568      0.571     -12.023      21.672
X1            15.3528     10.126      1.516      0.133      -4.744      35.450
X3             6.7179      8.530      0.788      0.433     -10.212      23.647
==============================================================================
Omnibus:                        8.800   Durbin-Watson:                   2.077
Prob(Omnibus):                  0.012   Jarque-Bera (JB):                9.219
Skew:                           0.554   Prob(JB):                      0.00996
Kurtosis:                       3.992   Cond. No.                         1.35
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

C:\Users\user\anaconda3\lib\site-packages\statsmodels\tsa\tsatools.py:142: FutureWarning: In a fu
ture version of pandas all arguments of concat except for the argument 'objs' will be keyword-onl
y
  x = pd.concat(x[::order], 1)
```

In [25]:
```
# Here we can see 2 OLS  results , 1 consits of Multi collinearity Prolem , thats why its Std.erro
# condition no.is changed
#   where compare to OlS results of 2
```

3. **Combine correlated variables**: If correlated independent variables represent similar information, consider combining them into a single variable. This combination can be done by averaging, summing, or using other mathematical operations, depending on the context and the nature of the variables.

In [26]:
```
import numpy as np
import pandas as pd
import statsmodels.api as sm
from sklearn.datasets import make_regression

# Generate a synthetic dataset with multicollinearity
np.random.seed(42)
X, y = make_regression(n_samples=100, n_features=3, noise=0.5, random_state=42)
X[:, 1] = X[:, 0] + 0.5 * np.random.normal(size=100)  # Introduce multicollinearity between column

# Convert data to a pandas DataFrame
data = pd.DataFrame(X, columns=['X1', 'X2', 'X3'])
data['y'] = y

# Calculate correlation matrix
corr_matrix = data.corr()
print("Correlation matrix:\n", corr_matrix)

# Combine the correlated variables X1 and X2 by taking their average
data['X1_X2_combined'] = (data['X1'] + data['X2']) / 2

# Add a constant term to the predictor variables
data_with_constant_all = sm.add_constant(data[['X1', 'X2', 'X3']])
data_with_constant_combined = sm.add_constant(data[['X1_X2_combined', 'X3']])

# Create and fit an OLS model using all three predictor variables
model_all = sm.OLS(data['y'], data_with_constant_all).fit()

# Print the summary for the model with all predictors
print("Regression summary for the model with all predictors:")
print(model_all.summary())

# Create and fit an OLS model using the combined variable and X3
model_combined = sm.OLS(data['y'], data_with_constant_combined).fit()

# Print the summary for the model with combined predictors (X1_X2_combined and X3)
print("\nRegression summary for the model with combined predictors (X1_X2_combined and X3):")
print(model_combined.summary())
```

```
Correlation matrix:
          X1        X2        X3         y
X1  1.000000  0.882948 -0.048636  0.148108
X2  0.882948  1.000000 -0.054696  0.165352
X3 -0.048636 -0.054696  1.000000  0.071536
y   0.148108  0.165352  0.071536  1.000000
Regression summary for the model with all predictors:
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.034
Model:                            OLS   Adj. R-squared:                  0.004
Method:                 Least Squares   F-statistic:                     1.122
Date:                Thu, 22 Jun 2023   Prob (F-statistic):              0.344
Time:                        20:34:33   Log-Likelihood:                -581.96
No. Observations:                 100   AIC:                             1172.
Df Residuals:                      96   BIC:                             1182.
Df Model:                           3
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          5.6185      8.572      0.655      0.514     -11.397      22.634
X1             0.9804     21.591      0.045      0.964     -41.877      43.838
X2            13.9157     18.451      0.754      0.453     -22.710      50.542
X3             6.8796      8.552      0.804      0.423     -10.095      23.854
==============================================================================
Omnibus:                        8.100   Durbin-Watson:                   2.070
Prob(Omnibus):                  0.017   Jarque-Bera (JB):                8.233
Skew:                           0.531   Prob(JB):                       0.0163
Kurtosis:                       3.921   Cond. No.                         4.13
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

Regression summary for the model with combined predictors (X1_X2_combined and X3):
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.033
Model:                            OLS   Adj. R-squared:                  0.013
Method:                 Least Squares   F-statistic:                     1.643
Date:                Thu, 22 Jun 2023   Prob (F-statistic):              0.199
Time:                        20:34:33   Log-Likelihood:                -582.02
No. Observations:                 100   AIC:                             1170.
Df Residuals:                      97   BIC:                             1178.
Df Model:                           2
Covariance Type:            nonrobust
==================================================================================
                     coef    std err          t      P>|t|      [0.025      0.975]
----------------------------------------------------------------------------------
const              5.2059      8.443      0.617      0.539     -11.552      21.963
X1_X2_combined    15.9730      9.594      1.665      0.099      -3.068      35.014
X3                 6.8469      8.512      0.804      0.423     -10.047      23.740
==============================================================================
Omnibus:                        8.244   Durbin-Watson:                   2.077
Prob(Omnibus):                  0.016   Jarque-Bera (JB):                8.431
Skew:                           0.536   Prob(JB):                       0.0148
Kurtosis:                       3.935   Cond. No.                         1.27
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

C:\Users\user\anaconda3\lib\site-packages\statsmodels\tsa\tsatools.py:142: FutureWarning: In a fu
ture version of pandas all arguments of concat except for the argument 'objs' will be keyword-onl
y
  x = pd.concat(x[::order], 1)
```

4. **Use partial least squares regression (PLS)**: PLS is a technique that combines features of both principal component analysis and multiple regression. It identifies linear combinations of the predictor variables (called latent variables) that have the highest covariance with the response variable, reducing multicollinearity while retaining most of the predictive power.

In [30]:
```
from sklearn.cross_decomposition import PLSRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.preprocessing import StandardScaler
import numpy as np

# Step 1: Data Collection
# Assume X contains independent variables (area, number of bedrooms, age) and y contains the depend
X = np.array([[100, 3, 10],
              [150, 4, 5],
              [120, 3, 8],
              [180, 5, 15],
              [90, 2, 12]])

y = np.array([250000, 400000, 300000, 500000, 200000])

# Step 2: Data Preparation
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Step 3: Data Splitting
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, random_state=42)

# Step 4: Model Building
n_components = 2  # Number of components for PLS regression

pls = PLSRegression(n_components=n_components)
pls.fit(X_train, y_train)

# Step 5: Model Evaluation
y_pred = pls.predict(X_test)

mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print("Mean Squared Error (MSE):", mse)
print("R-squared (R2):", r2)

# Step 6: Model Interpretation
# Coefficients of the PLS regression model
coefficients = pls.coef_

print("PLS Coefficients:")
for i, feature in enumerate(['Area', 'Bedrooms', 'Age']):
    print(f"{feature}: {coefficients[i]}")

# Step 7: Prediction
# Predict the price for a new house
new_house = np.array([[110, 3, 7]])  # New house with area=110, bedrooms=3, age=7

new_house_scaled = scaler.transform(new_house)
predicted_price = pls.predict(new_house_scaled)

print("Predicted Price for the new house:", predicted_price)
```

```
Mean Squared Error (MSE): 45632994.04288025
R-squared (R2): nan
PLS Coefficients:
Area: [67163.54631866]
Bedrooms: [64115.76716221]
Age: [1571.45287204]
Predicted Price for the new house: [[276698.35353589]]
```

```
C:\Users\user\anaconda3\lib\site-packages\sklearn\metrics\_regression.py:918: UndefinedMetricWarn
ing: R^2 score is not well-defined with less than two samples.
  warnings.warn(msg, UndefinedMetricWarning)
C:\Users\user\anaconda3\lib\site-packages\sklearn\cross_decomposition\_pls.py:507: FutureWarning:
The attribute `coef_` will be transposed in version 1.3 to be consistent with other linear models
in scikit-learn. Currently, `coef_` has a shape of (n_features, n_targets) and in the future it w
ill have a shape of (n_targets, n_features).
  warnings.warn(
```

In [ ]: