```
!pip install pandas
```

Show hidden output

```
import pandas as pd
import statsmodels.api as sm

df = pd.read_csv('dataset.csv', encoding='cp1252')
```

```
print(df.columns)
```

Show hidden output

3

```
print(df[['Human Development Index (HDI) ',
         'Gender Inequality Index',
         'Life expectancy at birth (years)',
         'Population age 15-64 years (millions)',
         'Gross national income (GNI) per capita (2017 PPP $)']].dtypes)
```

```
Human Development Index (HDI)                          object
Gender Inequality Index                                object
Life expectancy at birth (years)                       object
Population age 15-64 years (millions)                  object
Gross national income (GNI) per capita (2017 PPP $)    object
dtype: object
```

```
# Convert columns to numeric, forcing errors to NaN
df['Human Development Index (HDI) '] = pd.to_numeric(df['Human Development Index (HDI) '], errors='coerce')
df['Gender Inequality Index'] = pd.to_numeric(df['Gender Inequality Index'], errors='coerce')
df['Life expectancy at birth (years)'] = pd.to_numeric(df['Life expectancy at birth (years)'], errors='coerce')
df['Population age 15-64 years (millions)'] = pd.to_numeric(df['Population age 15-64 years (millions)'], errors='coerce')
df['Gross national income (GNI) per capita (2017 PPP $)'] = pd.to_numeric(df['Gross national income (GNI) per capita (2017 PPP $)'], errors='coerce')
```

```
# Drop rows with NaN values
df = df.dropna(subset=['Human Development Index (HDI) ',
                       'Gender Inequality Index',
                       'Life expectancy at birth (years)',
                       'Population age 15-64 years (millions)',
                       'Gross national income (GNI) per capita (2017 PPP $)'])
```

```
# Define your independent variables with the correct column names
X = df[['Human Development Index (HDI) ',
        'Gender Inequality Index',
        'Life expectancy at birth (years)',
        'Population age 15-64 years (millions)']]

# Add a constant to the model (intercept)
X = sm.add_constant(X)
# Define your dependent variable
y = df['Gross national income (GNI) per capita (2017 PPP $)']

# Fit the model
model = sm.OLS(y, X).fit()

# Print the summary
print(model.summary())
```

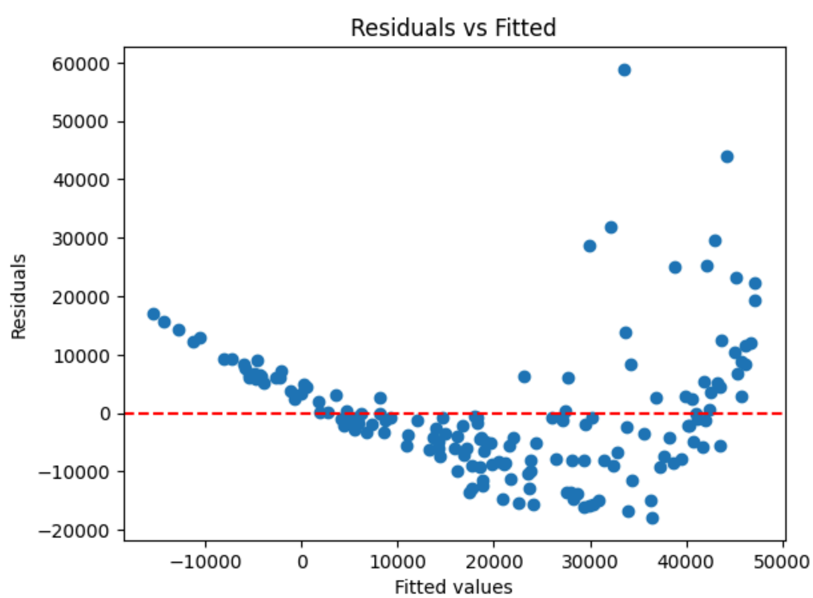
```
                            OLS Regression Results
==============================================================================================
Dep. Variable:     Gross national income (GNI) per capita (2017 PPP $)   R-squared:        0.684
Model:                                                OLS   Adj. R-squared:                 0.676
```

```
Method:                       Least Squares   F-statistic:                     84.85
Date:                      Thu, 15 Aug 2024   Prob (F-statistic):           3.12e-38
Time:                              17:00:42   Log-Likelihood:                -1741.0
No. Observations:                       162   AIC:                             3492.
Df Residuals:                           157   BIC:                             3508.
Df Model:                                 4
Covariance Type:                  nonrobust
==============================================================================================
                                     coef    std err          t      P>|t|      [0.025      0.975]
----------------------------------------------------------------------------------------------
const                           4862.5341   2.08e+04      0.233      0.816   -3.63e+04    4.6e+04
Human Development Index (HDI)   8.552e+04   1.98e+04      4.329      0.000    4.65e+04   1.25e+05
Gender Inequality Index        -3.777e+04   1.21e+04     -3.120      0.002   -6.17e+04  -1.39e+04
Life expectancy at birth (years) -458.8253   311.848     -1.471      0.143   -1074.784    157.134
Population age 15-64 years (millions) -6.5676   8.239    -0.797      0.427     -22.842      9.707
==============================================================================
Omnibus:                       70.466   Durbin-Watson:                   1.495
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              255.494
Skew:                           1.676   Prob(JB):                     3.31e-56
Kurtosis:                       8.158   Cond. No.                     3.18e+03
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 3.18e+03. This might indicate that there are
strong multicollinearity or other numerical problems.
```

```python
import matplotlib.pyplot as plt

plt.scatter(model.fittedvalues, model.resid)
plt.axhline(y=0, color='r', linestyle='--')
plt.xlabel('Fitted values')
plt.ylabel('Residuals')
plt.title('Residuals vs Fitted')
plt.show()
```

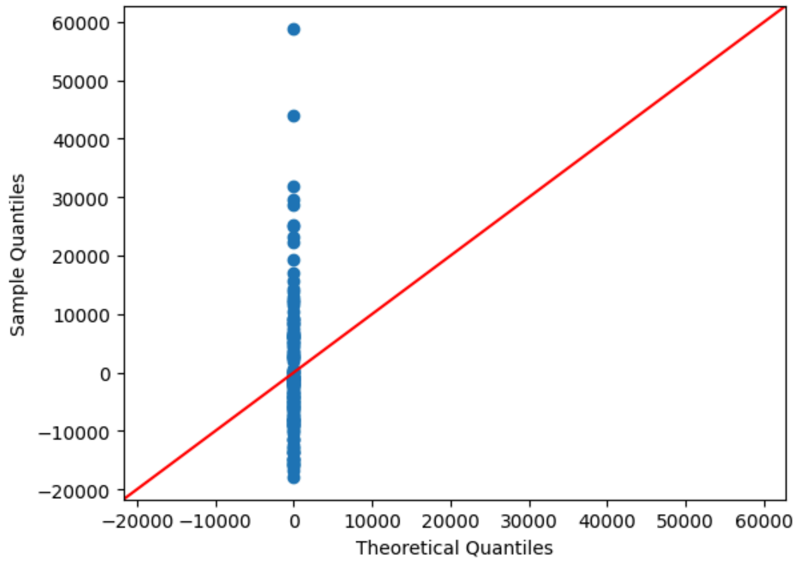



```python
from statsmodels.stats.diagnostic import het_breuschpagan

test = het_breuschpagan(model.resid, model.model.exog)
labels = ['LM Statistic', 'LM Test p-value', 'F-Statistic', 'F-Test p-value']
print(dict(zip(labels, test)))
```

```
{'LM Statistic': 8.669127461722518, 'LM Test p-value': 0.06992328095234364, 'F-Statistic': 2.2191437851999827, 'F-Test p-value': 0.06935281133931234}
```

```python
sm.qqplot(model.resid, line ='45')
plt.show()
```




```
from statsmodels.stats.outliers_influence import variance_inflation_factor

vif_data = pd.DataFrame()
vif_data["feature"] = X.columns
vif_data["VIF"] = [variance_inflation_factor(X.values, i) for i in range(len(X.columns))]
print(vif_data)
```

```
                          feature         VIF
0                           const  539.100071
1       Human Development Index (HDI)   10.658392
2           Gender Inequality Index    6.586116
3      Life expectancy at birth (years)    6.363189
4  Population age 15-64 years (millions)    1.005062
```

```
# Check the unique categories in the HDI Group column
hdi_groups = df['HDI Group'].unique()
print(hdi_groups)
print(f"Number of HDI categories: {len(hdi_groups)}")
```

```
['Very high' 'High' 'Medium' 'Low']
Number of HDI categories: 4
```

```
df['HDI Group'] = pd.Categorical(df['HDI Group'], categories=['Low', 'Medium', 'High', 'Very high'], ordered=True)

# Now create dummy variables, ensuring 'Low' is the reference category
df = pd.get_dummies(df, columns=['HDI Group'], drop_first=True)

# Check which dummy variables were created
print(df.columns)
```

Show hidden output

```
# Check the data types of the columns
print(df[['Gender Inequality Index',
          'Life expectancy at birth (years)',
          'Population age 15-64 years (millions)',
          'HDI Group_Medium', 'HDI Group_High', 'HDI Group_Very high']].dtypes)
```

```
Gender Inequality Index                  float64
Life expectancy at birth (years)         float64
Population age 15-64 years (millions)    float64
HDI Group_Medium                            bool
HDI Group_High                              bool
HDI Group_Very high                         bool
dtype: object
```

```
# Convert the columns to numeric, forcing errors to NaN (which can then be handled)
df['Gender Inequality Index'] = pd.to_numeric(df['Gender Inequality Index'], errors='coerce')
df['Life expectancy at birth (years)'] = pd.to_numeric(df['Life expectancy at birth (years)'], errors='coerce')
df['Population age 15-64 years (millions)'] = pd.to_numeric(df['Population age 15-64 years (millions)'], errors='coerce')
# Convert boolean columns to numeric
df['HDI Group_Medium'] = df['HDI Group_Medium'].astype(int)
df['HDI Group_High'] = df['HDI Group_High'].astype(int)
df['HDI Group_Very high'] = df['HDI Group_Very high'].astype(int)
```

```
# Check the data types to ensure all are numeric
print(df[['Gender Inequality Index',
          'Life expectancy at birth (years)',
          'Population age 15-64 years (millions)',
          'HDI Group_Medium', 'HDI Group_High', 'HDI Group_Very high']].dtypes)
```

```
Gender Inequality Index                  float64
Life expectancy at birth (years)         float64
Population age 15-64 years (millions)    float64
HDI Group_Medium                           int64
HDI Group_High                             int64
HDI Group_Very high                        int64
dtype: object
```

```
# Drop rows with NaN values
df = df.dropna(subset=['Gender Inequality Index',
                       'Life expectancy at birth (years)',
                       'Population age 15-64 years (millions)',
                       'HDI Group_Medium', 'HDI Group_High', 'HDI Group_Very high'])
```

```
# Define your independent variables
X = df[['Gender Inequality Index',
        'Life expectancy at birth (years)',
        'Population age 15-64 years (millions)',
        'HDI Group_Medium', 'HDI Group_High', 'HDI Group_Very high']]

# Add a constant to the model (intercept)
X = sm.add_constant(X)

# Define your dependent variable
y = df['Gross national income (GNI) per capita (2017 PPP $)']

# Fit the model
model = sm.OLS(y, X).fit()

# Print the summary of the model
print(model.summary())
```

```
                              OLS Regression Results
==============================================================================================================
Dep. Variable:     Gross national income (GNI) per capita (2017 PPP $)   R-squared:          0.736
Model:                                   OLS   Adj. R-squared:          0.726
Method:                        Least Squares   F-statistic:              72.09
Date:                       Thu, 15 Aug 2024   Prob (F-statistic):    2.41e-42
Time:                               17:00:43   Log-Likelihood:         -1726.3
No. Observations:                        162   AIC:                      3467.
Df Residuals:                            155   BIC:                      3488.
Df Model:                                  6
Covariance Type:                   nonrobust
==========================================================================================================
                          coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------------------
const                 -5428.5966   1.97e+04     -0.275      0.784   -4.44e+04    3.36e+04
Gender Inequality Index     -3.498e+04   1.05e+04     -3.339      0.001   -5.57e+04   -1.43e+04
Life expectancy at birth (years)     461.6654    254.551      1.814      0.072     -41.171     964.501
Population age 15-64 years (millions)    -0.0110      7.664     -0.001      0.999     -15.150      15.128
HDI Group_Medium         -3306.5352   3199.464     -1.033      0.303   -9626.715    3013.644
HDI Group_High           -3446.7064   3860.354     -0.893      0.373   -1.11e+04    4178.988
HDI Group_Very high       1.558e+04   5214.288      2.988      0.003    5278.007    2.59e+04
==============================================================================
Omnibus:                       60.766   Durbin-Watson:                   1.878
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              230.338
Skew:                           1.390   Prob(JB):                     9.61e-51
```

```
Kurtosis:                      8.138   Cond. No.                    2.99e+03
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 2.99e+03. This might indicate that there are
strong multicollinearity or other numerical problems.
```

```
df = pd.read_csv('dataset.csv', encoding='cp1252')
```

```
# Reorder the HDI Group to make 'Very High' the reference category
df['HDI Group'] = pd.Categorical(df['HDI Group'], categories=['Very high', 'Low', 'Medium', 'High'], ordered=True)

# Create dummy variables again, this time 'Very High' will be the reference category (and thus not included in the dummy variables)
df = pd.get_dummies(df, columns=['HDI Group'], drop_first=True)

# Check to see which columns were created
print(df.columns)
```

Show hidden output

```
# Check the data types of all relevant columns
print(df[['Gender Inequality Index',
        'Life expectancy at birth (years)',
        'Population age 15-64 years (millions)',
        'HDI Group_Low', 'HDI Group_Medium', 'HDI Group_High',
        'Gross national income (GNI) per capita (2017 PPP $)']].dtypes)
```

```
Gender Inequality Index                                object
Life expectancy at birth (years)                       object
Population age 15-64 years (millions)                  object
HDI Group_Low                                          bool
HDI Group_Medium                                       bool
HDI Group_High                                         bool
Gross national income (GNI) per capita (2017 PPP $)    object
dtype: object
```

```
# Convert the columns to numeric, forcing errors to NaN (which can then be handled)
df['Gender Inequality Index'] = pd.to_numeric(df['Gender Inequality Index'], errors='coerce')
df['Life expectancy at birth (years)'] = pd.to_numeric(df['Life expectancy at birth (years)'], errors='coerce')
df['Population age 15-64 years (millions)'] = pd.to_numeric(df['Population age 15-64 years (millions)'], errors='coerce')
# Convert boolean columns to numeric
df['HDI Group_Medium'] = df['HDI Group_Medium'].astype(int)
df['HDI Group_High'] = df['HDI Group_High'].astype(int)
df['HDI Group_Low'] = df['HDI Group_Low'].astype(int)
df['Gross national income (GNI) per capita (2017 PPP $)'] = pd.to_numeric(df['Gross national income (GNI) per capita (2017 PPP $)'], errors='coerce')

# Drop rows with NaN values or handle them appropriately
df = df.dropna(subset=['Gender Inequality Index',
                'Life expectancy at birth (years)',
                'Population age 15-64 years (millions)',
                'HDI Group_Low', 'HDI Group_Medium', 'HDI Group_High',
                'Gross national income (GNI) per capita (2017 PPP $)'])
```

```
# Verify that all columns are now numeric
print(df[['Gender Inequality Index',
        'Life expectancy at birth (years)',
        'Population age 15-64 years (millions)',
        'HDI Group_Low', 'HDI Group_Medium', 'HDI Group_High',
        'Gross national income (GNI) per capita (2017 PPP $)']].dtypes)
```

```
Gender Inequality Index                                float64
Life expectancy at birth (years)                       float64
Population age 15-64 years (millions)                  float64
HDI Group_Low                                          int64
HDI Group_Medium                                       int64
HDI Group_High                                         int64
Gross national income (GNI) per capita (2017 PPP $)    float64
dtype: object
```

```
# Define your independent variables
X = df[['Gender Inequality Index',
        'Life expectancy at birth (years)',
        'Population age 15-64 years (millions)',
        'HDI Group_Low', 'HDI Group_Medium', 'HDI Group_High']]

# Add a constant to the model (intercept)
X = sm.add_constant(X)

# Define your dependent variable
y = df['Gross national income (GNI) per capita (2017 PPP $)']

# Fit the model
model = sm.OLS(y, X).fit()

# Print the summary of the model
print(model.summary())
```

```
                              OLS Regression Results
=====================================================================================================
Dep. Variable:     Gross national income (GNI) per capita (2017 PPP $)   R-squared:           0.736
Model:                                    OLS   Adj. R-squared:           0.726
Method:                         Least Squares   F-statistic:              72.09
Date:                        Thu, 15 Aug 2024   Prob (F-statistic):       2.41e-42
Time:                                17:12:17   Log-Likelihood:           -1726.3
No. Observations:                         162   AIC:                      3467.
Df Residuals:                             155   BIC:                      3488.
Df Model:                                   6
Covariance Type:                    nonrobust
=====================================================================================================
                                          coef    std err          t      P>|t|      [0.025      0.975]
-----------------------------------------------------------------------------------------------------
const                                1.015e+04   2.11e+04      0.481      0.631   -3.15e+04    5.18e+04
Gender Inequality Index             -3.498e+04   1.05e+04     -3.339      0.001   -5.57e+04   -1.43e+04
Life expectancy at birth (years)      461.6654    254.551      1.814      0.072     -41.171     964.501
Population age 15-64 years (millions)  -0.0110      7.664     -0.001      0.999     -15.150      15.128
HDI Group_Low                       -1.558e+04   5214.288     -2.988      0.003   -2.59e+04   -5278.007
HDI Group_Medium                    -1.888e+04   4001.807     -4.719      0.000   -2.68e+04    -1.1e+04
HDI Group_High                      -1.902e+04   2828.801     -6.725      0.000   -2.46e+04   -1.34e+04
==============================================================================
Omnibus:                       60.766   Durbin-Watson:                   1.878
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              230.338
Skew:                           1.390   Prob(JB):                     9.61e-51
Kurtosis:                       8.138   Cond. No.                     3.10e+03
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 3.1e+03. This might indicate that there are
strong multicollinearity or other numerical problems.
```

```
# Create interaction terms between HDI groups and GII
df['Interaction_Low_GII'] = df['HDI Group_Low'] * df['Gender Inequality Index']
df['Interaction_Medium_GII'] = df['HDI Group_Medium'] * df['Gender Inequality Index']
df['Interaction_High_GII'] = df['HDI Group_High'] * df['Gender Inequality Index']
```

```
# Define the independent variables including the interaction terms
X = df[['Gender Inequality Index',
        'Life expectancy at birth (years)',
        'Population age 15-64 years (millions)',
        'HDI Group_Low', 'HDI Group_Medium', 'HDI Group_High',
        'Interaction_Low_GII', 'Interaction_Medium_GII', 'Interaction_High_GII']]

# Add a constant to the model (intercept)
X = sm.add_constant(X)

# Define the dependent variable
y = df['Gross national income (GNI) per capita (2017 PPP $)']

# Fit the model
model = sm.OLS(y, X).fit()

# Print the summary of the model
print(model.summary())
```

```
                            OLS Regression Results
==============================================================================================================
Dep. Variable:     Gross national income (GNI) per capita (2017 PPP $)   R-squared:                0.772
Model:                                   OLS   Adj. R-squared:           0.759
Method:                        Least Squares   F-statistic:              57.34
Date:                       Thu, 15 Aug 2024   Prob (F-statistic):       2.04e-44
Time:                               17:35:01   Log-Likelihood:           -1714.4
No. Observations:                        162   AIC:                      3449.
Df Residuals:                            152   BIC:                      3480.
Df Model:                                  9
Covariance Type:                   nonrobust
=========================================================================================================
                                     coef    std err          t      P>|t|      [0.025      0.975]
-----------------------------------------------------------------------------------------------
const                              2.169e+04   2.04e+04      1.061      0.291   -1.87e+04    6.21e+04
Gender Inequality Index           -8.243e+04   1.39e+04     -5.926      0.000   -1.1e+05    -5.49e+04
Life expectancy at birth (years)    408.8524    245.086      1.668      0.097    -75.362     893.067
Population age 15-64 years (millions)  2.6840    7.288      0.368      0.713    -11.714      17.082
HDI Group_Low                     -4.933e+04   1.55e+04     -3.184      0.002   -7.99e+04   -1.87e+04
HDI Group_Medium                  -4.788e+04   1.17e+04     -4.096      0.000   -7.1e+04    -2.48e+04
HDI Group_High                    -3.845e+04   6541.483     -5.878      0.000   -5.14e+04   -2.55e+04
Interaction_Low_GII                8.987e+04   2.81e+04      3.194      0.002    3.43e+04    1.45e+05
Interaction_Medium_GII              9.03e+04   2.66e+04      3.401      0.001    3.78e+04    1.43e+05
Interaction_High_GII               7.989e+04   2.07e+04      3.869      0.000    3.91e+04    1.21e+05
==============================================================================
Omnibus:                       80.198   Durbin-Watson:                   2.145
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              477.270
Skew:                           1.714   Prob(JB):                    2.30e-104
Kurtosis:                      10.678   Cond. No.                     5.27e+03
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 5.27e+03. This might indicate that there are
```