group3-code

April 21, 2025

1 IMPORT LIBRARY/MODULE

[ ]:
```
import pandas as pd
import numpy as np
from sklearn.feature_selection import RFE
import statsmodels.api as sm
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import MinMaxScaler
import matplotlib.pyplot as plt
import seaborn as sns
import statsmodels.api as sm
from scipy.stats import zscore
from tqdm import tqdm


from statsmodels.stats.diagnostic import linear_reset #Ramsey RESET test
from statsmodels.stats.stattools import jarque_bera # Residual Normality test
from statsmodels.stats.outliers_influence import variance_inflation_factor #␣
↪VIF for multicollinearity
from statsmodels.stats.diagnostic import het_breuschpagan #BP test for␣
↪heteroskedasticity
```

2 READ FILE AND PREPROCESS DATA

2.1 Read file

[ ]:
```
df = pd.read_excel("group3_data.xlsx")
```

[ ]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 900 entries, 0 to 899
Data columns (total 14 columns):
 #   Column                                 Non-Null Count  Dtype
---  ------                                 --------------  -----
 0   Country Name                           900 non-null    object
```

```
 1   Country Code                             900 non-null    object
 2   Year                                     900 non-null    int64
 3   Suicide mortality rate                   900 non-null    float64
 4   Prevalence of bipolar disorder           900 non-null    float64
 5   Prevalence of anxiety disorder           900 non-null    float64
 6   Prevalence of depression                 900 non-null    float64
 7   Alcohol use disorders                    900 non-null    float64
 8   Prevalence of eating disorders           900 non-null    float64
 9   Continent                                900 non-null    object
 10  Current health expenditure per capita    875 non-null    float64
 11  GDP per capita                           669 non-null    float64
 12  Inflation                                669 non-null    float64
 13  Unemployment                             669 non-null    float64
dtypes: float64(10), int64(1), object(3)
memory usage: 98.6+ KB
```

The DataFrame df has 900 rows and 15 columns, with some columns such as `Current health expenditure per capita`, `GDP per capita`, `Inflation`, and `Unemployment` having missing values.

2.2 Drop null

[ ]:
```
df.columns = [col.strip().split(' (')[0].replace(" ", "_").lower() for col in df.columns]
df.dropna(inplace=True)
```

[ ]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 666 entries, 2 to 899
Data columns (total 14 columns):
 #   Column                                   Non-Null Count  Dtype
---  ------                                   --------------  -----
 0   country_name                             666 non-null    object
 1   country_code                             666 non-null    object
 2   year                                     666 non-null    int64
 3   suicide_mortality_rate                   666 non-null    float64
 4   prevalence_of_bipolar_disorder           666 non-null    float64
 5   prevalence_of_anxiety_disorder           666 non-null    float64
 6   prevalence_of_depression                 666 non-null    float64
 7   alcohol_use_disorders                    666 non-null    float64
 8   prevalence_of_eating_disorders           666 non-null    float64
 9   continent                                666 non-null    object
 10  current_health_expenditure_per_capita    666 non-null    float64
 11  gdp_per_capita                           666 non-null    float64
 12  inflation                                666 non-null    float64
 13  unemployment                             666 non-null    float64
dtypes: float64(10), int64(1), object(3)
```

```
memory usage: 78.0+ KB
```

- After cleaning, the DataFrame has 666 rows and 15 columns, with all columns being complete (no missing values).
- Column names have been renamed to lowercase with underscores.

2.3 Normalize

[ ]:
```
variables_to_normalize = [
    'prevalence_of_bipolar_disorder',
    'prevalence_of_anxiety_disorder',
    'prevalence_of_eating_disorders',
    'prevalence_of_depression',
    'alcohol_use_disorders'
]

for var in variables_to_normalize:
    df[var] = df[var] * 1000
```

Since the suicide rate is measured per 100,000 people, this step will help normalize the prevalence of mental health disorders into prevalence per 100,000 people.

[ ]:
```
df.head()
```

[ ]:
```
  country_name country_code  year  suicide_mortality_rate  \
2  Afghanistan          AFG  2008                     4.6
3  Afghanistan          AFG  2012                     4.0
4  Afghanistan          AFG  2016                     4.0
5      Albania          ALB  2000                     4.9
6      Albania          ALB  2004                     4.8

   prevalence_of_bipolar_disorder  prevalence_of_anxiety_disorder  \
2                      720.775512                     4825.200606
3                      722.916001                     4897.918983
4                      725.151614                     4990.617027
5                      574.251922                     3904.539352
6                      575.931987                     3922.757146

   prevalence_of_depression  alcohol_use_disorders  \
2                  4129.656                659.501
3                  4132.485                662.372
4                  4135.694                661.850
5                  2195.285               1654.338
6                  2222.116               1701.433

   prevalence_of_eating_disorders continent  \
2                      104.483898      Asia
3                      113.536770      Asia
```

```
4                       121.569505      Asia
5                       107.727638    Europe
6                       116.205704    Europe

   current_health_expenditure_per_capita  gdp_per_capita  inflation  \
2                             137.703508      381.733238  26.418664
3                             155.383026      651.417134   6.441213
4                             239.188747      522.082216   4.383892
5                             222.817466     1126.683340   0.050018
6                             316.166335     2373.581292   2.280019

   unemployment
2         7.878
3         7.856
4        10.133
5        19.023
6        16.306
```

[ ]:
```python
df.describe()
```

[ ]:
```
              year  suicide_mortality_rate  prevalence_of_bipolar_disorder  \
count   666.000000              666.000000                      666.000000
mean   2008.318318               11.013514                      697.575885
std       5.602969                8.956871                      253.503491
min    2000.000000                0.500000                      192.200622
25%    2004.000000                5.200000                      547.863715
50%    2008.000000                8.600000                      597.839467
75%    2012.000000               13.675000                      920.025620
max    2016.000000               87.000000                     1668.155916

       prevalence_of_anxiety_disorder  prevalence_of_depression  \
count                      666.000000                666.000000
mean                      4377.212424               3521.814610
std                       1305.789040                656.897127
min                       1954.403153               2194.091000
25%                       3516.263746               3094.153000
50%                       4019.297834               3508.550500
75%                       5088.015005               3939.231250
max                       8835.455678               5739.526000

       alcohol_use_disorders  prevalence_of_eating_disorders  \
count             666.000000                      666.000000
mean             1587.723796                      219.626924
std               895.656970                      165.843149
min               449.900000                       59.860895
25%               987.957500                       99.085360
50%              1460.450000                      156.225493
```

```
75%               1813.768750                        291.954483
max               5467.508000                       1107.007406

       current_health_expenditure_per_capita  gdp_per_capita   inflation  \
count                             666.000000      666.000000  666.000000
mean                             1107.799287    12481.886221    7.092488
std                              1460.973061    18910.796118   16.056752
min                                19.000000      122.269203   -5.355400
25%                               139.568929     1121.035085    2.002124
50%                               478.700905     3998.474565    4.064958
75%                              1470.479015    14924.741200    8.327062
max                              9599.891046   120422.137934  324.996872

       unemployment
count    666.000000
mean       7.378351
std        5.343756
min        0.150000
25%        3.720000
50%        5.923500
75%        9.929500
max       29.770000
```

Statistical summary of all numeric columns: Count, Mean, Standard deviation, Min, Max, 25th, 50th, and 75th percentiles.

[ ]:
```python
excluded_vars = ['suicide_mortality_rate', 'unemployment', 'inflation', 'year']

df_log = df.copy()
log_vars = [col for col in df_log.select_dtypes(include=[np.number]).columns if␣
 ↪col not in excluded_vars]

for col in log_vars:
    df_log[col] = np.log(df_log[col])


df_log.rename(columns={col: f"log_{col}" for col in log_vars}, inplace=True)


df_log.replace([np.inf, -np.inf], np.nan, inplace=True)
df_log.dropna(inplace=True)


y = df_log['suicide_mortality_rate']
X = df_log.drop(columns=['suicide_mortality_rate', 'country_name',␣
 ↪'country_code', 'year', 'continent'], errors='ignore')
```

```
X = X.astype(float)
y = y.astype(float)
```

[ ]:
```
df_log.describe()
```

[ ]:
```
              year  suicide_mortality_rate  \
count   666.000000              666.000000
mean   2008.318318               11.013514
std       5.602969                8.956871
min    2000.000000                0.500000
25%    2004.000000                5.200000
50%    2008.000000                8.600000
75%    2012.000000               13.675000
max    2016.000000               87.000000

       log_prevalence_of_bipolar_disorder  log_prevalence_of_anxiety_disorder  \
count                          666.000000                          666.000000
mean                             6.476322                            8.343564
std                              0.390990                            0.281131
min                              5.258540                            7.577840
25%                              6.306027                            8.165154
50%                              6.393322                            8.298862
75%                              6.824401                            8.534642
max                              7.419474                            9.086528

       log_prevalence_of_depression  log_alcohol_use_disorders  \
count                    666.000000                 666.000000
mean                       8.149376                   7.243348
std                        0.187299                   0.493257
min                        7.693523                   6.109025
25%                        8.037269                   6.895639
50%                        8.162958                   7.286500
75%                        8.278741                   7.503162
max                        8.655132                   8.606578

       log_prevalence_of_eating_disorders  \
count                          666.000000
mean                             5.159224
std                              0.659076
min                              4.092023
25%                              4.595982
50%                              5.051300
75%                              5.676567
max                              7.009416

       log_current_health_expenditure_per_capita  log_gdp_per_capita  \
count                                 666.000000          666.000000
```

```
mean                                        6.135302          8.325331
std                                         1.419871          1.589284
min                                         2.944439          4.806225
25%                                         4.938389          7.022003
50%                                         6.171076          8.293665
75%                                         7.293343          9.610769
max                                         9.169507         11.698759

        inflation  unemployment
count  666.000000    666.000000
mean     7.092488      7.378351
std     16.056752      5.343756
min     -5.355400      0.150000
25%      2.002124      3.720000
50%      4.064958      5.923500
75%      8.327062      9.929500
max    324.996872     29.770000
```

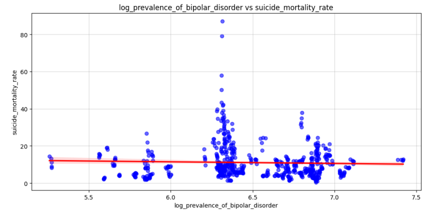
2.4 Scatter plot

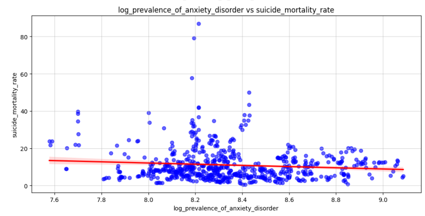
[ ]:
```python
num_col = ['log_prevalence_of_bipolar_disorder',
           'log_prevalence_of_anxiety_disorder',
           'log_prevalence_of_depression',
           'log_alcohol_use_disorders',
           'log_prevalence_of_eating_disorders',
           'log_current_health_expenditure_per_capita',
           'log_gdp_per_capita',
           'inflation',
           'unemployment']
```

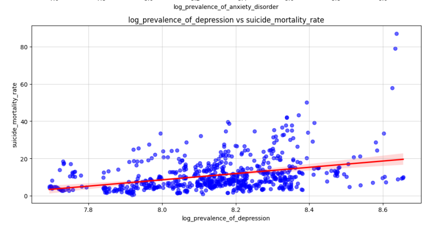
[ ]:
```python
target = 'suicide_mortality_rate'
```

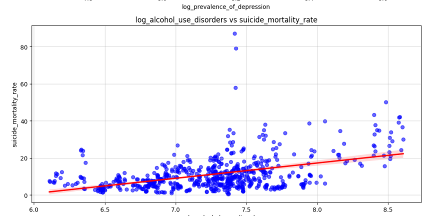
[ ]:
```python
plt.figure(figsize=(10, 5 * len(num_col)))

num_rows = len(num_col)
for i, num_var in enumerate(num_col, 1):
    plt.subplot(num_rows, 1, i)

    # Scatter plot
    plt.scatter(df_log[num_var], df_log[target], color="blue", alpha=0.6)

    # Regression line (red)
    sns.regplot(x=df_log[num_var],
                y=df_log[target],
                scatter=False,
                color="red")
```

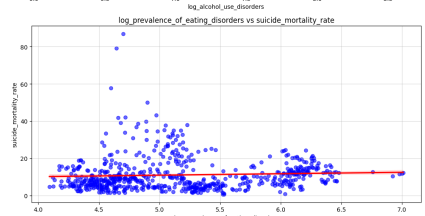
```python
    # Title and labels
    plt.title(f"{num_var} vs {target}")
    plt.xlabel(num_var)
    plt.ylabel(target)
    plt.grid(alpha=0.5)

# Adjust subplot spacing
plt.subplots_adjust(wspace=0, hspace=5)

# Tight layout for better spacing
plt.tight_layout()
plt.show()
```

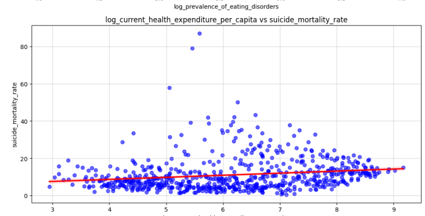

log_prevalence_of_bipolar_disorder vs suicide_mortality_rate
suicide_mortality_rate
log_prevalence_of_bipolar_disorder

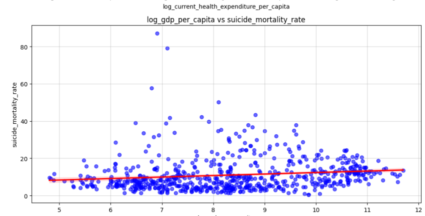

log_prevalence_of_anxiety_disorder vs suicide_mortality_rate
suicide_mortality_rate
log_prevalence_of_anxiety_disorder

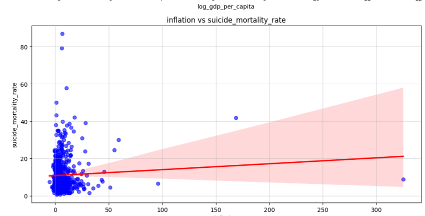

log_prevalence_of_depression vs suicide_mortality_rate
suicide_mortality_rate
log_prevalence_of_depression

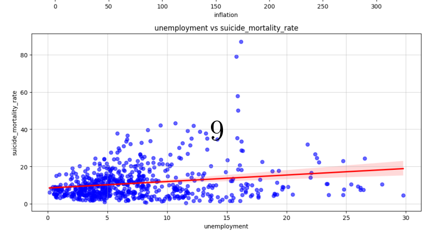

log_alcohol_use_disorders vs suicide_mortality_rate
suicide_mortality_rate
log_alcohol_use_disorders


log_prevalence_of_eating_disorders vs suicide_mortality_rate
suicide_mortality_rate
log_prevalence_of_eating_disorders


log_current_health_expenditure_per_capita vs suicide_mortality_rate
suicide_mortality_rate
log_current_health_expenditure_per_capita

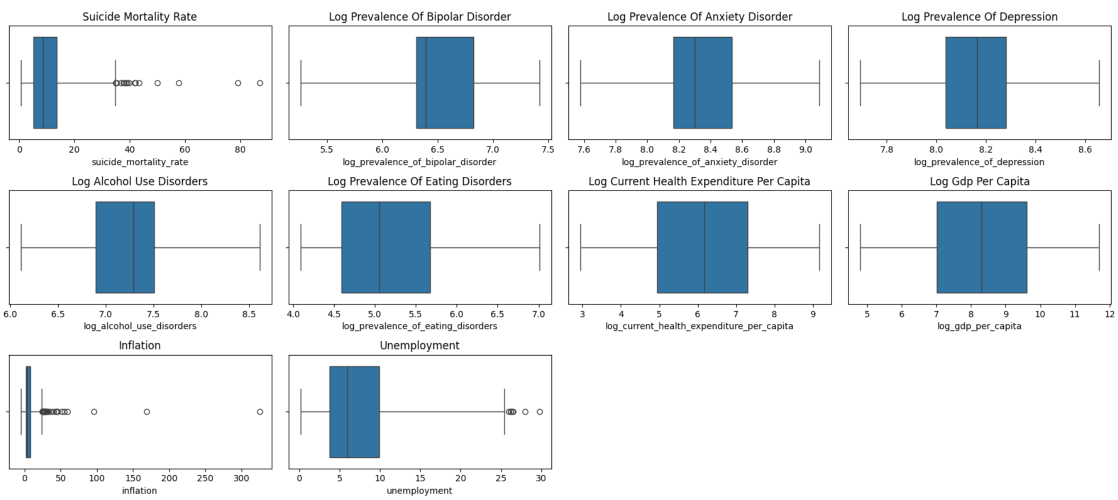

log_gdp_per_capita vs suicide_mortality_rate
suicide_mortality_rate
log_gdp_per_capita


inflation vs suicide_mortality_rate
suicide_mortality_rate
inflation


unemployment vs suicide_mortality_rate
suicide_mortality_rate
unemployment

2.5 Boxplot for detecting and treating outliers

[ ]:
```
df_plot = df_log.drop(columns=['year', 'country_name', 'country_code',␣
↪'continent'])
columns = df_plot.columns.tolist()

fig, axs = plt.subplots(3, 4, figsize=(18, 8))

for i, col in enumerate(columns):
    row = i // 4
    col_pos = i % 4
    sns.boxplot(x=df_plot[col], ax=axs[row, col_pos], whis=2.5)
    axs[row, col_pos].set_title(col.replace('_', ' ').title())

for j in range(len(columns), 12):
    axs[j // 4, j % 4].axis('off')

plt.tight_layout()
plt.show()
```

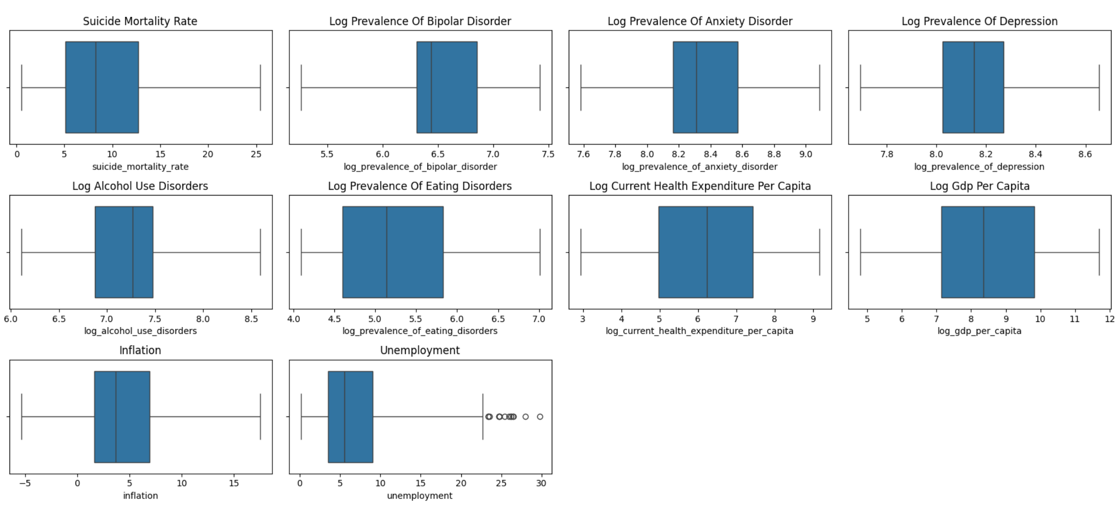


[ ]:
```
Q1 = df_log['suicide_mortality_rate'].quantile(0.25)
Q3 = df_log['suicide_mortality_rate'].quantile(0.75)
IQR = Q3 - Q1

lower_bound = Q1 - 1.5*IQR
upper_bound = Q3 + 1.5*IQR
```

```python
df_log = df_log[(df_log['suicide_mortality_rate'] >= lower_bound) & (df_log['suicide_mortality_rate'] <= upper_bound)]
```

[ ]:
```python
Q1 = df_log['inflation'].quantile(0.25)
Q3 = df_log['inflation'].quantile(0.75)
IQR = Q3 - Q1

lower_bound = Q1 - 1.5*IQR
upper_bound = Q3 + 1.5*IQR

df_log = df_log[(df_log['inflation'] >= lower_bound) & (df_log['inflation'] <= upper_bound)]
```

[ ]:
```python
df_plot = df_log.drop(columns=['year', 'country_name', 'country_code', 'continent'])
columns = df_plot.columns.tolist()

fig, axs = plt.subplots(3, 4, figsize=(18, 8))

for i, col in enumerate(columns):
    row = i // 4
    col_pos = i % 4
    sns.boxplot(x=df_plot[col], ax=axs[row, col_pos], whis=2.5)
    axs[row, col_pos].set_title(col.replace('_', ' ').title())

for j in range(len(columns), 12):
    axs[j // 4, j % 4].axis('off')

plt.tight_layout()
plt.show()
```



[ ]:
```
df_log.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 588 entries, 3 to 899
Data columns (total 14 columns):
 #   Column                                Non-Null Count  Dtype
---  ------                                --------------  -----
 0   country_name                          588 non-null    object
 1   country_code                          588 non-null    object
 2   year                                  588 non-null    int64
 3   suicide_mortality_rate                588 non-null    float64
 4   log_prevalence_of_bipolar_disorder    588 non-null    float64
 5   log_prevalence_of_anxiety_disorder    588 non-null    float64
 6   log_prevalence_of_depression          588 non-null    float64
 7   log_alcohol_use_disorders             588 non-null    float64
 8   log_prevalence_of_eating_disorders    588 non-null    float64
 9   continent                             588 non-null    object
 10  log_current_health_expenditure_per_capita 588 non-null    float64
 11  log_gdp_per_capita                    588 non-null    float64
 12  inflation                             588 non-null    float64
 13  unemployment                          588 non-null    float64
dtypes: float64(10), int64(1), object(3)
memory usage: 68.9+ KB
```

[ ]:
```
df_log.describe()
```

[ ]:
```
              year  suicide_mortality_rate  \
count   588.000000              588.000000
mean   2008.578231                9.456803
std       5.619903                5.585093
min    2000.000000                0.500000
25%    2004.000000                5.075000
50%    2008.000000                8.250000
75%    2012.000000               12.700000
max    2016.000000               25.400000

       log_prevalence_of_bipolar_disorder  log_prevalence_of_anxiety_disorder  \
count                          588.000000                          588.000000
mean                             6.488233                            8.360441
std                              0.406574                            0.285906
min                              5.258540                            7.582434
25%                              6.303451                            8.164478
50%                              6.439094                            8.309910
75%                              6.852073                            8.571461
max                              7.419474                            9.086528

       log_prevalence_of_depression  log_alcohol_use_disorders  \
```

```
count                    588.000000                 588.000000
mean                       8.139512                   7.192856
std                        0.185579                   0.457026
min                        7.693523                   6.109025
25%                        8.023982                   6.873360
50%                        8.151592                   7.269026
75%                        8.270801                   7.473190
max                        8.655132                   8.590631

       log_prevalence_of_eating_disorders  \
count                          588.000000
mean                             5.205924
std                              0.678614
min                              4.092023
25%                              4.597677
50%                              5.134231
75%                              5.825012
max                              7.009416

       log_current_health_expenditure_per_capita  log_gdp_per_capita  \
count                                 588.000000          588.000000
mean                                    6.213467            8.438998
std                                     1.444915            1.598133
min                                     2.944439            4.806225
25%                                     4.978605            7.138408
50%                                     6.236404            8.349048
75%                                     7.438060            9.815937
max                                     9.169507           11.698759

        inflation  unemployment
count  588.000000    588.000000
mean     4.691502      7.103080
std      4.120504      5.320859
min     -5.355400      0.150000
25%      1.665195      3.537250
50%      3.682269      5.534000
75%      6.936934      9.074000
max     17.489449     29.770000
```

3 MODEL 1

[ ]:
```
y = df_log['suicide_mortality_rate']
X = df_log.drop(columns=['suicide_mortality_rate', 'country_name',␣
↪'country_code', 'year', 'continent'], errors='ignore')

X = X.astype(float)
```

```python
y = y.astype(float)

X = sm.add_constant(X)

model_1 = sm.OLS(y, X).fit()
residuals = model_1.resid

print(model_1.summary())
```

```
                                OLS Regression Results
================================================================================
==
Dep. Variable:     suicide_mortality_rate   R-squared:
0.396
Model:                                OLS   Adj. R-squared:
0.387
Method:                     Least Squares   F-statistic:
42.09
Date:                    Sun, 20 Apr 2025   Prob (F-statistic):
7.81e-58
Time:                            15:44:31   Log-Likelihood:
-1697.1
No. Observations:                     588   AIC:
3414.
Df Residuals:                         578   BIC:
3458.
Df Model:                               9
Covariance Type:                nonrobust
================================================================================
=============================
                                                coef    std err          t
P>|t|      [0.025      0.975]
--------------------------------------------------------------------------------
-----------------------------
const                                       -51.6153     10.540     -4.897
0.000     -72.318     -30.913
log_prevalence_of_bipolar_disorder           -4.2680      0.892     -4.785
0.000      -6.020      -2.516
log_prevalence_of_anxiety_disorder           -5.8411      0.879     -6.646
0.000      -7.567      -4.115
log_prevalence_of_depression                 10.1844      1.061      9.603
0.000       8.101      12.267
log_alcohol_use_disorders                     4.9359      0.428     11.522
0.000       4.095       5.777
log_prevalence_of_eating_disorders            1.5204      1.123      1.354
0.176      -0.685       3.726
log_current_health_expenditure_per_capita     0.5409      0.499      1.083
```

```
0.279      -0.440      1.521
log_gdp_per_capita                          0.9337      0.473      1.974
0.049       0.005      1.863
inflation                                  -0.0468      0.047     -0.988
0.324      -0.140      0.046
unemployment                                0.0371      0.036      1.034
0.302      -0.033      0.108
==============================================================================
Omnibus:                       49.330   Durbin-Watson:                   0.660
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               59.814
Skew:                           0.734   Prob(JB):                     1.03e-13
Kurtosis:                       3.534   Cond. No.                     1.24e+03
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
[2] The condition number is large, 1.24e+03. This might indicate that there are
strong multicollinearity or other numerical problems.
```

3.1 Ramsey's test for linear model

[ ]:
```
reset_test = linear_reset(model_1, power=2, use_f=True)
print("Ramsey RESET Test:", reset_test)
```

```
Ramsey RESET Test: <F test: F=0.7281858320893587, p=0.39382596316335305,
df_denom=577, df_num=1>
```

[ ]:
```
if reset_test.pvalue > 0.05:
  print("No error, no omitted variable", "The model is satisfied linearity␣
↪assumption", sep = '\n')
else:
  print("There's an error and one or many omitted variables", "Our model is not␣
↪linearly validated", sep = '\n')
```

```
No error, no omitted variable
The model is satisfied linearity assumption
```

3.2 t-test for zero mean

[ ]:
```
from scipy.stats import ttest_1samp
residuals = model_1.resid
t_stat, p_value = ttest_1samp(residuals, 0)
print(f"T-Statistic: {t_stat}, P-Value: {p_value}")
```

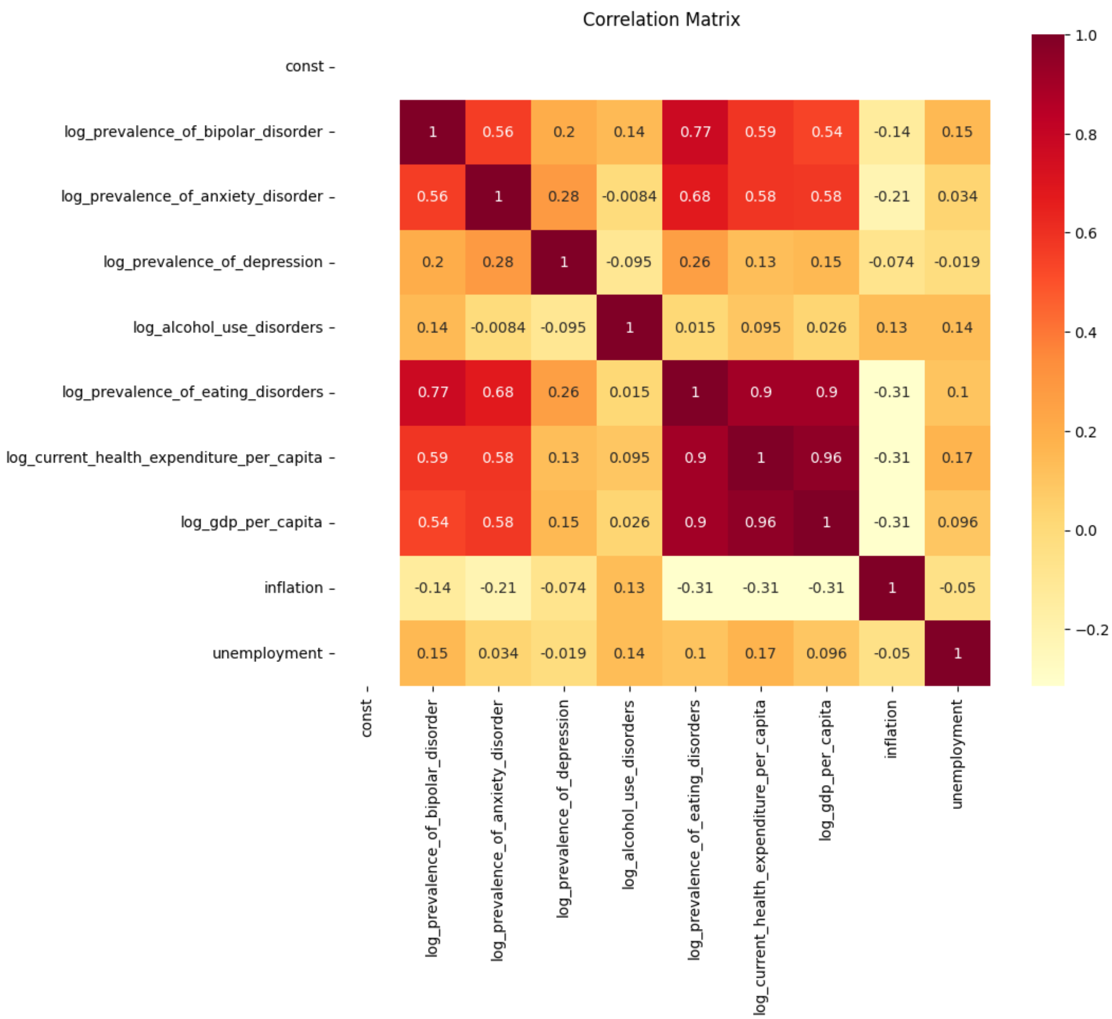
```
T-Statistic: 1.706496139849354e-13, P-Value: 0.9999999999998639
```

3.3 Correlation matrix and VIF for multicollinear

[ ]:
```
vif_data = pd.DataFrame()
vif_data["feature"] = X.columns
vif_data["VIF"] = [variance_inflation_factor(X.values, i) for i in range(X.
 ↪shape[1])]
print(vif_data)
```

```
                                 feature          VIF
0                                  const  3413.808579
1         log_prevalence_of_bipolar_disorder     4.033381
2         log_prevalence_of_anxiety_disorder     1.936941
3               log_prevalence_of_depression     1.188261
4                  log_alcohol_use_disorders     1.175772
5         log_prevalence_of_eating_disorders    17.818187
6   log_current_health_expenditure_per_capita    15.964839
7                         log_gdp_per_capita    17.521044
8                                  inflation     1.167006
9                               unemployment     1.119337
```

[ ]:
```
corr_matrix = X.corr()

plt.figure(figsize=(10, 8))
sns.heatmap(corr_matrix, annot=True, cmap='YlOrRd')
plt.title('Correlation Matrix')
plt.show()
```



4 MODEL 2: REMOVE `log_gdp_per_capita`

4.1 OLS Regression

[ ]:
```
y = df_log['suicide_mortality_rate']
X = df_log.drop(columns=['suicide_mortality_rate', 'country_name',␣
↪'country_code', 'year', 'continent', 'log_gdp_per_capita'], errors='ignore')

X = X.astype(float)
y = y.astype(float)

X = sm.add_constant(X)

model_2 = sm.OLS(y, X).fit()
residuals = model_2.resid
```

```python
print(model_2.summary())
```

```
                                OLS Regression Results
================================================================================
==
Dep. Variable:     suicide_mortality_rate   R-squared:
0.392
Model:                                OLS   Adj. R-squared:
0.383
Method:                     Least Squares   F-statistic:
46.64
Date:                    Sun, 20 Apr 2025   Prob (F-statistic):
7.44e-58
Time:                            15:44:33   Log-Likelihood:
-1699.0
No. Observations:                     588   AIC:
3416.
Df Residuals:                         579   BIC:
3455.
Df Model:                               8
Covariance Type:                nonrobust
================================================================================
=============================
                                                coef    std err          t
P>|t|      [0.025      0.975]
--------------------------------------------------------------------------------
-----------------------------
const                                       -46.8013     10.280     -4.553
0.000     -66.992     -26.610
log_prevalence_of_bipolar_disorder           -4.9595      0.822     -6.031
0.000      -6.575      -3.344
log_prevalence_of_anxiety_disorder           -5.9346      0.880     -6.745
0.000      -7.663      -4.207
log_prevalence_of_depression                 10.0788      1.062      9.492
0.000       7.993      12.164
log_alcohol_use_disorders                     4.8824      0.429     11.392
0.000       4.041       5.724
log_prevalence_of_eating_disorders            2.5347      1.001      2.532
0.012       0.568       4.501
log_current_health_expenditure_per_capita     1.2402      0.353      3.516
0.000       0.547       1.933
inflation                                    -0.0413      0.047     -0.872
0.383      -0.134       0.052
unemployment                                  0.0268      0.036      0.752
0.453      -0.043       0.097
```

```
==============================================================================
Omnibus:                       50.846   Durbin-Watson:                   0.661
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               62.039
Skew:                           0.749   Prob(JB):                     3.38e-14
Kurtosis:                       3.539   Cond. No.                     1.11e+03
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
[2] The condition number is large, 1.11e+03. This might indicate that there are
strong multicollinearity or other numerical problems.
```

4.2 Ramsey's test for linear model

```
[ ]: reset_test = linear_reset(model_2, power=2, use_f=True)
     print("Ramsey RESET Test:", reset_test)
```

```
Ramsey RESET Test: <F test: F=1.113683691403977, p=0.2917235008962452,
df_denom=578, df_num=1>
```

```
[ ]: if reset_test.pvalue > 0.05:
       print("No error, no omitted variable", "The model is satisfied linearity␣
      ↪assumption", sep = '\n')
     else:
       print("There's an error and one or many omitted variables", "Our model is not␣
      ↪linearly validated", sep = '\n')
```

```
No error, no omitted variable
The model is satisfied linearity assumption
```

4.3 t-test for zero mean

```
[ ]: from scipy.stats import ttest_1samp
     residuals = model_2.resid
     t_stat, p_value = ttest_1samp(residuals, 0)
     print(f"T-Statistic: {t_stat}, P-Value: {p_value}")
```

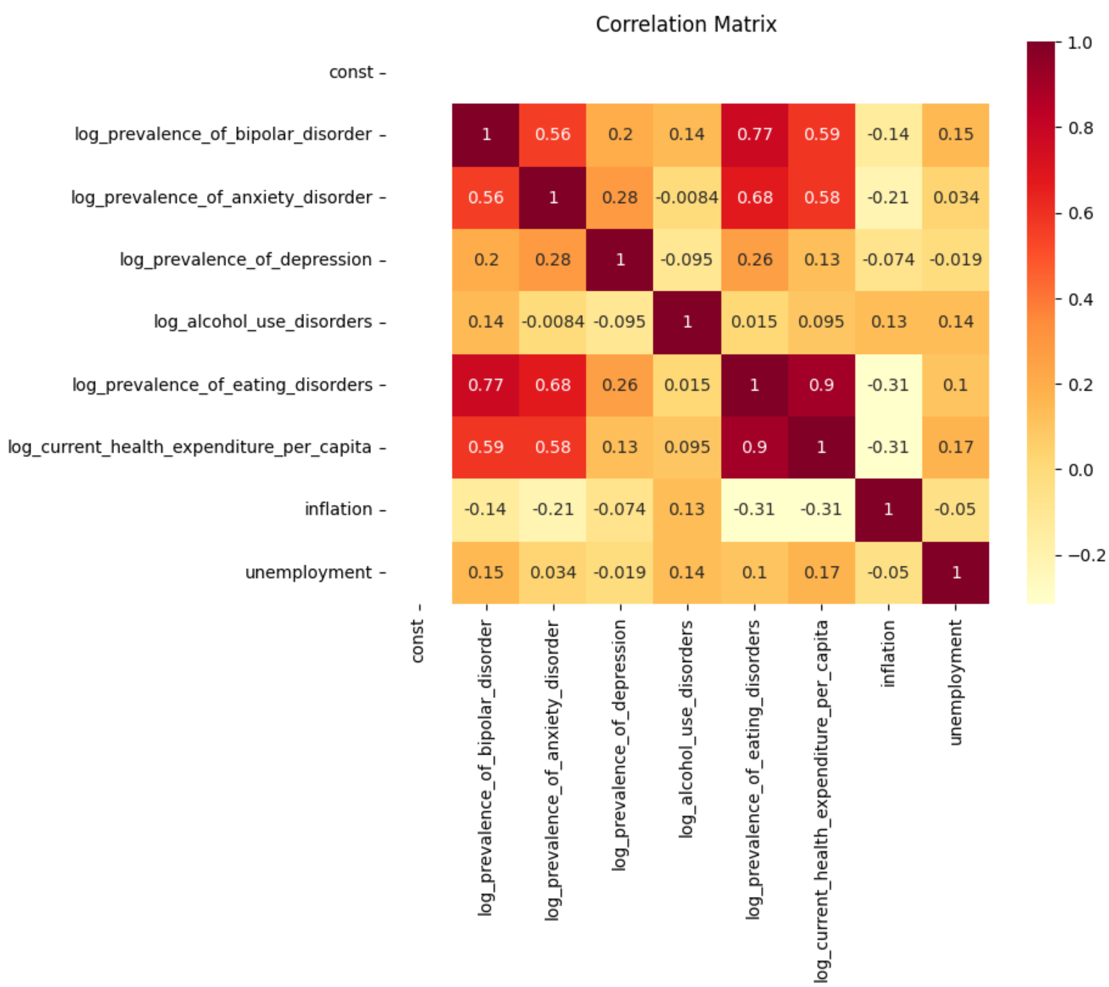
```
T-Statistic: 2.0519584532507295e-13, P-Value: 0.9999999999998364
```

4.4 Correlation matrix and VIF for multicollinear

```
[ ]: vif_data = pd.DataFrame()
     vif_data["feature"] = X.columns
     vif_data["VIF"] = [variance_inflation_factor(X.values, i) for i in range(X.
      ↪shape[1])]
     print(vif_data)
```

```
                                    feature          VIF
0                                     const  3231.134275
1           log_prevalence_of_bipolar_disorder     3.411409
2           log_prevalence_of_anxiety_disorder     1.931323
3                 log_prevalence_of_depression     1.185238
4                    log_alcohol_use_disorders     1.171068
5           log_prevalence_of_eating_disorders    14.089693
6  log_current_health_expenditure_per_capita     7.930084
7                                 inflation     1.163050
8                              unemployment     1.095427
```

[ ]:
```
corr_matrix = X.corr()

plt.figure(figsize=(8, 6))
sns.heatmap(corr_matrix, annot=True, cmap='YlOrRd')
plt.title('Correlation Matrix')
plt.show()
```



5 MODEL 3: REMOVE log_prevalence_of_eating_disorders

5.1 OLS Regression

```
[ ]: y = df_log['suicide_mortality_rate']
     X = df_log.drop(columns=['suicide_mortality_rate', 'country_name',␣
      ↪'country_code', 'year', 'continent', 'log_gdp_per_capita',␣
      ↪'log_prevalence_of_eating_disorders'], errors='ignore')

     X = X.astype(float)
     y = y.astype(float)

     X = sm.add_constant(X)

     model_3 = sm.OLS(y, X).fit()
     residuals = model_3.resid

     print(model_3.summary())
```

```
                                OLS Regression Results
================================================================================
==
Dep. Variable:     suicide_mortality_rate   R-squared:
0.385
Model:                                OLS   Adj. R-squared:
0.378
Method:                     Least Squares   F-statistic:
51.90
Date:                    Sun, 20 Apr 2025   Prob (F-statistic):
2.28e-57
Time:                            15:44:35   Log-Likelihood:
-1702.3
No. Observations:                     588   AIC:
3421.
Df Residuals:                         580   BIC:
3456.
Df Model:                               7
Covariance Type:                nonrobust
================================================================================
=============================
                                             coef    std err          t
P>|t|      [0.025      0.975]
--------------------------------------------------------------------------------
-----------------------------
const                                    -55.1756      9.779     -5.642
0.000     -74.382     -35.970
log_prevalence_of_bipolar_disorder        -3.5215      0.597     -5.895
0.000      -4.695      -2.348
```

```
log_prevalence_of_anxiety_disorder           -5.4350      0.861     -6.309
0.000      -7.127      -3.743
log_prevalence_of_depression                 10.7547      1.033     10.416
0.000       8.727      12.783
log_alcohol_use_disorders                     4.5938      0.415     11.067
0.000       3.779       5.409
log_current_health_expenditure_per_capita     2.0155      0.176     11.459
0.000       1.670       2.361
inflation                                    -0.0551      0.047     -1.166
0.244      -0.148       0.038
unemployment                                  0.0104      0.035      0.294
0.769      -0.059       0.079
==============================================================================
Omnibus:                       51.217   Durbin-Watson:                   0.684
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               62.593
Skew:                           0.752   Prob(JB):                     2.56e-14
Kurtosis:                       3.542   Cond. No.                     1.01e+03
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
[2] The condition number is large, 1.01e+03. This might indicate that there are
strong multicollinearity or other numerical problems.
```

Ramsey's test for linear model

[ ]:
```
reset_test = linear_reset(model_3, power=2, use_f=True)
print("Ramsey RESET Test:", reset_test)
```

```
Ramsey RESET Test: <F test: F=0.015245417955895837, p=0.9017759272935483,
df_denom=579, df_num=1>
```

[ ]:
```
if reset_test.pvalue > 0.05:
  print("No error, no omitted variable", "The model is satisfied linearity␣
 ↪assumption", sep = '\n')
else:
  print("There's an error and one or many omitted variables", "Our model is not␣
 ↪linearly validated", sep = '\n')
```

```
No error, no omitted variable
The model is satisfied linearity assumption
```

5.2 t-test for zero mean

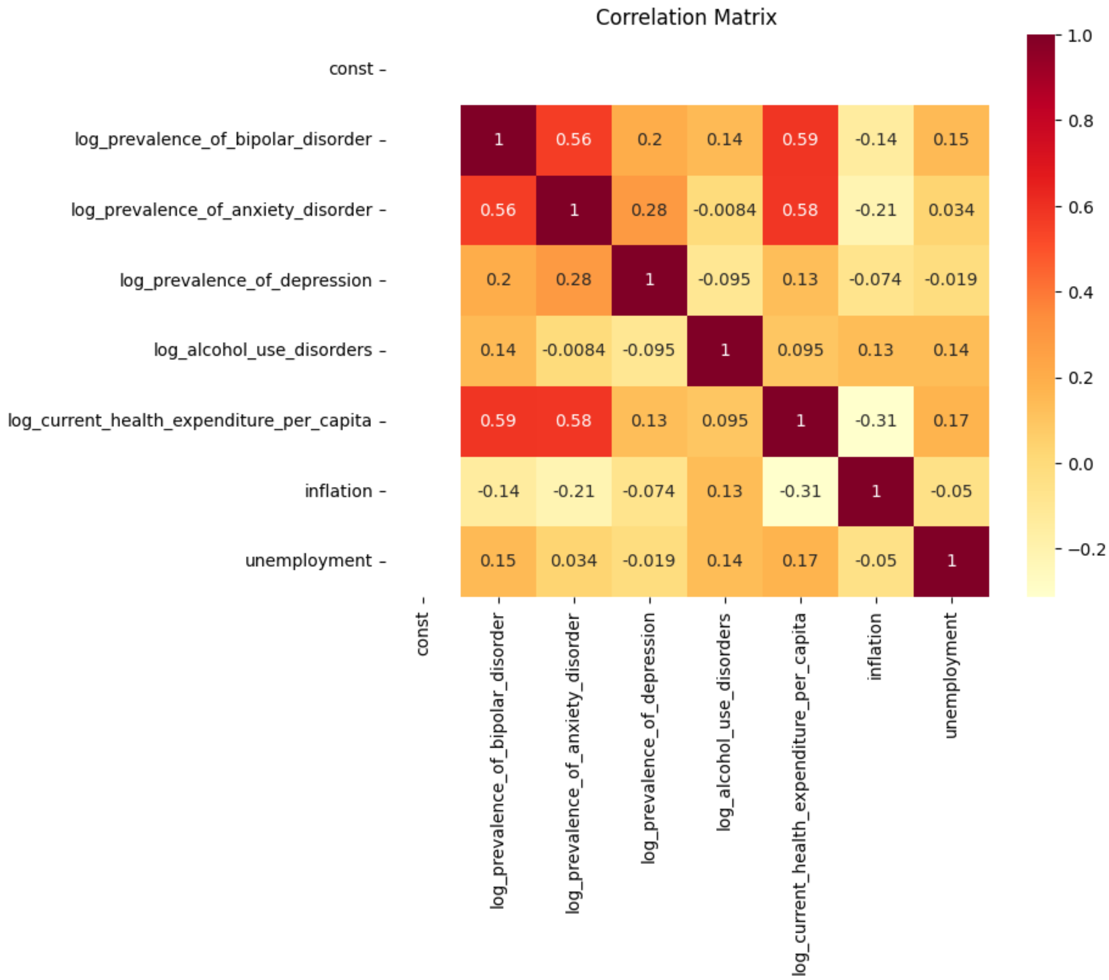
[ ]:
```
from scipy.stats import ttest_1samp
residuals = model_3.resid
t_stat, p_value = ttest_1samp(residuals, 0)
```

```python
print(f"T-Statistic: {t_stat}, P-Value: {p_value}")
```

```
T-Statistic: 2.5585628505742805e-13, P-Value: 0.9999999999997959
```

5.3 Correlation matrix and VIF for multicollinear

[ ]:
```python
corr_matrix = X.corr()

plt.figure(figsize=(8, 6))
sns.heatmap(corr_matrix, annot=True, cmap='YlOrRd')
plt.title('Correlation Matrix')
plt.show()
```



[ ]:
```python
vif_data = pd.DataFrame()
vif_data["feature"] = X.columns
vif_data["VIF"] = [variance_inflation_factor(X.values, i) for i in range(X.
 ↪shape[1])]
```

```
print(vif_data)
```

```
                                    feature          VIF
0                                     const  2896.598764
1         log_prevalence_of_bipolar_disorder     1.783510
2         log_prevalence_of_anxiety_disorder     1.834156
3               log_prevalence_of_depression     1.110304
4                  log_alcohol_use_disorders     1.088232
5  log_current_health_expenditure_per_capita     1.953321
6                                 inflation     1.147681
7                              unemployment     1.059134
```

5.4 5. BP-test for Heteroskedasticity

[ ]:
```
y_pred = model_3.predict(X)
```

[ ]:
```
bp_test = het_breuschpagan(residuals, X)

bp_results = {
    "Lagrange multiplier statistic": bp_test[0],
    "p-value": bp_test[1],
    "F-statistic": bp_test[2],
    "F p-value": bp_test[3]
}

print("Breusch-Pagan Test Results:")
for key, value in bp_results.items():
    print(f"{key}: {value:.4e}")
```

```
Breusch-Pagan Test Results:
Lagrange multiplier statistic: 3.4713e+01
p-value: 1.2663e-05
F-statistic: 5.1984e+00
F p-value: 9.2271e-06
```

[ ]:
```
pval = bp_test[1]
if pval < 0.05:
    print("Homoskedasticity is violated.")
else:
    print("Homoskedasticity assumption is not violated.")
print(pval)
```

```
Homoskedasticity is violated.
1.2662710917134855e-05
```

5.5 6. Durbin-Watson test for autocorrelation

[ ]:
```python
from statsmodels.stats.stattools import durbin_watson

independent_vars = df_log.drop(columns=['suicide_mortality_rate',␣
↪'country_name', 'country_code', 'year',
                                        'continent', 'log_gdp_per_capita',␣
↪'log_prevalence_of_eating_disorders'], errors='ignore')
y = df_log['suicide_mortality_rate']

print("\nDurbin-Watson test for autocorrelation:")
for var in independent_vars:
    X_dw = sm.add_constant(df_log[[var]])
    model = sm.OLS(y, X_dw).fit()
    dw_stat = durbin_watson(model.resid)
    print(f"\nIndependent variable: {var}")
    print(f"Durbin-Watson statistic: {dw_stat:.4f}")
```

```
Durbin-Watson test for autocorrelation:

Independent variable: log_prevalence_of_bipolar_disorder
Durbin-Watson statistic: 0.5775

Independent variable: log_prevalence_of_anxiety_disorder
Durbin-Watson statistic: 0.5721

Independent variable: log_prevalence_of_depression
Durbin-Watson statistic: 0.5916

Independent variable: log_alcohol_use_disorders
Durbin-Watson statistic: 0.5488

Independent variable: log_current_health_expenditure_per_capita
Durbin-Watson statistic: 0.6355

Independent variable: inflation
Durbin-Watson statistic: 0.5998

Independent variable: unemployment
Durbin-Watson statistic: 0.5733
```

5.6 7. Normality of residual

n = 588

6 GLS regression

6.1 Construct Model

[ ]:
```
X = df_log[['log_prevalence_of_bipolar_disorder',
            'log_prevalence_of_anxiety_disorder',
            'log_prevalence_of_depression',
            'log_alcohol_use_disorders',
            'log_current_health_expenditure_per_capita',
            'inflation',
            'unemployment']].astype(float)

y = df_log['suicide_mortality_rate'].astype(float)

X = sm.add_constant(X)

# Fit the OLS model to get residuals
ols_model = sm.OLS(y, X).fit()
residuals = ols_model.resid
y_cap = ols_model.predict(X)

# Binning forecast values
bins = list(range(int(y_cap.min()), int(y_cap.max()) + 1))
ycap_binned = pd.cut(y_cap, bins=bins)

# Calculate variance of residuals within each bin
bin_variances = []
bin_midpoints = []

for interval in ycap_binned.cat.categories:
    idx = ycap_binned == interval
    if idx.sum() > 1:
        var = residuals[idx].var()
        midpoint = (interval.left + interval.right) / 2
        bin_variances.append(var)
        bin_midpoints.append(midpoint)

# Regress log-variance ~ midpoint
log_variances = np.log(bin_variances)
df_var = pd.DataFrame({
    'midpoint': bin_midpoints,
    'log_variance': log_variances
})
var_model = sm.OLS(df_var['log_variance'], sm.add_constant(df_var['midpoint'])).
 ↪fit()

# Predict the variance for the entire data set
fitted_log_var = var_model.predict(sm.add_constant(y_cap))
```

```python
fitted_var = np.exp(fitted_log_var)

# Create Sigma matrix for GLS
sigma_matrix = np.diag(fitted_var)

# Fit the GLS model
gls_model = sm.GLS(y, X, sigma=sigma_matrix)
gls_pre_results = gls_model.fit()

# GLS + HAC (robust với heteroskedasticity + autocorrelation)
gls_results = gls_pre_results.get_robustcov_results(cov_type='HAC', maxlags = 1)
print("\nGLS + HAC:")
print(gls_results.summary())
```

```
GLS + HAC:
                              GLS Regression Results
================================================================================
==
Dep. Variable:     suicide_mortality_rate   R-squared:
0.344
Model:                                GLS   Adj. R-squared:
0.336
Method:                     Least Squares   F-statistic:
29.46
Date:                    Sun, 20 Apr 2025   Prob (F-statistic):
7.63e-35
Time:                            15:44:37   Log-Likelihood:
-1687.6
No. Observations:                     588   AIC:
3391.
Df Residuals:                         580   BIC:
3426.
Df Model:                               7
Covariance Type:                      HAC
================================================================================
=============================
                                               coef    std err          t
P>|t|      [0.025      0.975]
--------------------------------------------------------------------------------
-----------------------------
const                                      -65.8462     13.985     -4.708
0.000     -93.313     -38.380
log_prevalence_of_bipolar_disorder          -3.7441      0.796     -4.703
0.000      -5.308      -2.180
log_prevalence_of_anxiety_disorder          -4.5182      1.266     -3.569
0.000      -7.005      -2.032
```

```
log_prevalence_of_depression                    11.3976      1.113     10.242
0.000       9.212      13.583
log_alcohol_use_disorders                        4.7672      0.500      9.543
0.000       3.786       5.748
log_current_health_expenditure_per_capita        1.7376      0.257      6.772
0.000       1.234       2.241
inflation                                       -0.0859      0.047     -1.832
0.067      -0.178       0.006
unemployment                                    -0.0182      0.036     -0.506
0.613      -0.089       0.053
==============================================================================
Omnibus:                       65.530   Durbin-Watson:                   0.673
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               85.635
Skew:                           0.865   Prob(JB):                     2.54e-19
Kurtosis:                       3.710   Cond. No.                     1.07e+03
==============================================================================

Notes:
[1] Standard Errors are heteroscedasticity and autocorrelation robust (HAC)
using 1 lags and without small sample correction
[2] The condition number is large, 1.07e+03. This might indicate that there are
strong multicollinearity or other numerical problems.
```

[ ]:
```python
# Comparison of coefficient estimates between OLS and GLS
coefficients_comparison = pd.DataFrame({
    'OLS': model_3.params,
    'GLS': gls_results.params
})
print("=== Coefficients Comparison ===")
print(coefficients_comparison)

# Comparison of standard errors between OLS and GLS
std_errors_comparison = pd.DataFrame({
    'OLS': model_3.bse,
    'GLS': gls_results.bse
})
print("\n=== Standard Errors Comparison ===")
print(std_errors_comparison)
```

```
=== Coefficients Comparison ===
                                                 OLS        GLS
const                                     -55.175643 -65.846213
log_prevalence_of_bipolar_disorder         -3.521470  -3.744095
log_prevalence_of_anxiety_disorder         -5.434959  -4.518159
log_prevalence_of_depression               10.754737  11.397565
log_alcohol_use_disorders                   4.593810   4.767157
log_current_health_expenditure_per_capita   2.015511   1.737558
inflation                                  -0.055114  -0.085853
```

```
unemployment                                0.010355  -0.018246

=== Standard Errors Comparison ===
                                                OLS        GLS
const                                      9.778762  13.984621
log_prevalence_of_bipolar_disorder         0.597321   0.796156
log_prevalence_of_anxiety_disorder         0.861398   1.265976
log_prevalence_of_depression               1.032529   1.112830
log_alcohol_use_disorders                  0.415078   0.499562
log_current_health_expenditure_per_capita  0.175895   0.256578
inflation                                  0.047279   0.046868
unemployment                               0.035172   0.036067
```

6.2 Detect and corect residual and influence points

[ ]:
```python
residuals_gls = gls_results.resid
standardized_gls_resid = zscore(residuals_gls)
```

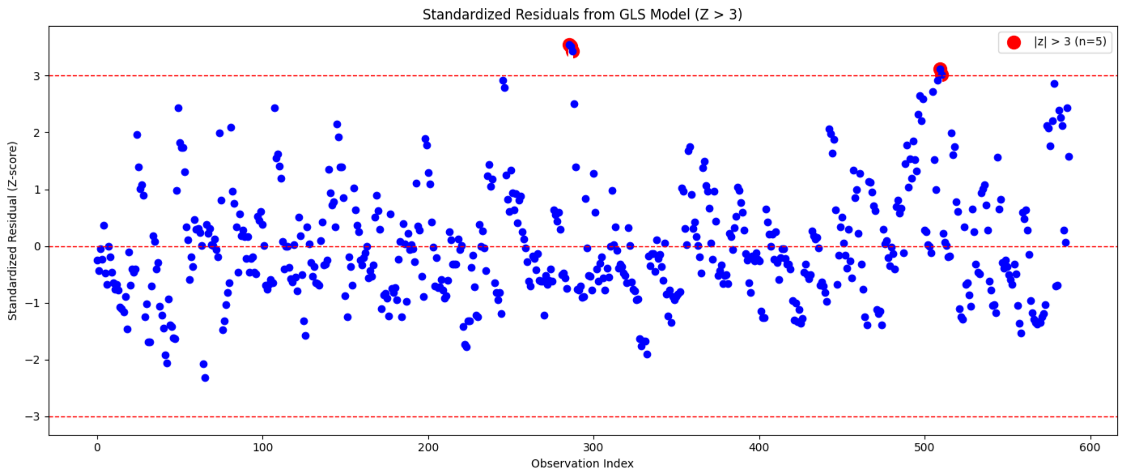
[ ]:
```python
from scipy.stats import zscore
residuals_gls = gls_results.resid
standardized_gls_resid = zscore(residuals_gls)
df_log["gls_outlier_z>3"] = np.abs(standardized_gls_resid) > 3
```

[ ]:
```python
residuals_gls = gls_results.resid
standardized_gls_resid = zscore(residuals_gls)
standardized_gls_resid_np = np.asarray(standardized_gls_resid)

# Flag |z| > 3
outlier_mask_z3 = np.abs(standardized_gls_resid_np) > 3
outlier_indices_z3 = np.where(outlier_mask_z3)[0]

df_log["gls_outlier_z>3"] = False
df_log.loc[df_log.index[outlier_indices_z3], "gls_outlier_z>3"] = True

gls_outliers_z3 = df_log[df_log["gls_outlier_z>3"] == True]
print(gls_outliers_z3)
```

```
    country_name country_code  year  suicide_mortality_rate  \
410        Japan          JPN  2000                    23.9
411        Japan          JPN  2004                    24.1
412        Japan          JPN  2008                    24.4
767     Suriname          SUR  2008                    24.5
768     Suriname          SUR  2012                    25.2

     log_prevalence_of_bipolar_disorder  log_prevalence_of_anxiety_disorder  \
410                            6.548598                            7.951466
411                            6.578126                            7.955562
412                            6.563019                            7.916878
```

```
767                         6.836148                            8.351886
768                         6.837994                            8.358631

     log_prevalence_of_depression  log_alcohol_use_disorders  \
410                      8.002684                   6.344804
411                      8.025117                   6.330934
412                      8.040751                   6.334142
767                      8.294871                   7.476257
768                      8.301034                   7.479025

     log_prevalence_of_eating_disorders continent  \
410                            6.010288      Asia
411                            6.024970      Asia
412                            6.037059      Asia
767                            5.376777  Americas
768                            5.428374  Americas

     log_current_health_expenditure_per_capita  log_gdp_per_capita  inflation  \
410                                   7.560080           10.575650  -0.676579
411                                   7.751475           10.553179  -0.008573
412                                   7.956126           10.593538   1.380079
767                                   6.427354            8.791195  14.667143
768                                   6.554318            9.088313   5.006863

     unemployment  gls_outlier_z>3
410         4.748             True
411         4.734             True
412         4.002             True
767         8.831             True
768         8.100             True
```

[ ]:
```python
plt.figure(figsize=(14, 6))
plt.stem(standardized_gls_resid_np, linefmt='white', markerfmt='bo', basefmt='␣
↪')
plt.scatter(outlier_indices_z3, standardized_gls_resid_np[outlier_indices_z3],
            color='red', edgecolors='red', s=120, linewidth=1.5,
            label=f'|z| > 3 (n={len(outlier_indices_z3)})')
plt.axhline(3, color='red', linestyle='--', linewidth=1)
plt.axhline(-3, color='red', linestyle='--', linewidth=1)
plt.axhline(0, color='red', linestyle='--', linewidth=1)
plt.title("Standardized Residuals from GLS Model (Z > 3)")
plt.xlabel("Observation Index")
plt.ylabel("Standardized Residual (Z-score)")
plt.legend()
plt.tight_layout()
plt.show()
```



[ ]:
```
original_params = gls_results.params
influential_scores = []

# Get the actual indices of df
df_indices = df_log.index.to_list()

# Loop through the valid indices to avoid out-of-bounds errors
for idx in tqdm(df_indices):
    try:
        # Remove the row based on the index
        X_loo = X.drop(index=idx)
        y_loo = y.drop(index=idx)
        sigma_loo = np.delete(np.delete(sigma_matrix, df_log.index.
 ↪get_loc(idx), axis=0), df_log.index.get_loc(idx), axis=1)

        # Fit GLS without that observation
        model_loo = sm.GLS(y_loo, X_loo, sigma=sigma_loo).fit()

        # Measure the deviation between the new and original coefficients
        score = np.linalg.norm(model_loo.params.values - original_params)
        influential_scores.append(score)
    except:
        influential_scores.append(np.nan)  # fallback in case of error

# Add the influence_score column to df_log
df_log["gls_influence_score"] = influential_scores

# Display the top influential observations
top_influential = df_log.sort_values("gls_influence_score", ascending=False).
 ↪head(10)
```

```
100%|          | 588/588 [00:32<00:00, 18.04it/s]
```

[ ]:
```
top_influential[['country_name', 'year', 'suicide_mortality_rate',␣
↪'gls_outlier_z>3', 'gls_influence_score']]
```

[ ]:
```
                country_name  year  suicide_mortality_rate  gls_outlier_z>3  \
410                    Japan  2000                    23.9             True
411                    Japan  2004                    24.1             True
565                  Myanmar  2000                     4.7            False
412                    Japan  2008                    24.4             True
155  Central African Republic  2000                    19.0            False
566                  Myanmar  2004                     4.2            False
413                    Japan  2012                    21.6            False
157  Central African Republic  2008                    15.1            False
111                   Brazil  2004                     4.4            False
156  Central African Republic  2004                    15.7            False

     gls_influence_score
410             3.679987
411             3.357421
565             3.172833
412             2.945951
155             2.568986
566             1.873788
413             1.777083
157             1.609349
111             1.493545
156             1.483687
```

Remove 5 points including outliners and influential obs

[ ]:
```
# Calculate z-scores and identify outliers with |z| > 3
residuals_gls = gls_results.resid
standardized_gls_resid = zscore(residuals_gls)
standardized_gls_resid_np = np.asarray(standardized_gls_resid)
outlier_z3_indices = np.where(np.abs(standardized_gls_resid_np) > 3)[0]

# Get indices of influential observations with influence score > 3.0
influential_indices = df_log[df_log["gls_influence_score"] > 3.0].index

# Combine indices of outliers and influential points
outlier_indices_df = df_log.index[outlier_z3_indices]
combined_indices = sorted(set(outlier_indices_df).
↪union(set(influential_indices)))

# Create cleaned dataset by removing identified indices
X_combined_cleaned = X.drop(index=combined_indices)
y_combined_cleaned = y.drop(index=combined_indices)
```

```
# Remove corresponding rows and columns from the sigma matrix
rows_to_remove = [df_log.index.get_loc(i) for i in combined_indices]
sigma_matrix_combined_cleaned = np.delete(np.delete(sigma_matrix,␣
↪rows_to_remove, axis=0), rows_to_remove, axis=1)

# Fit GLS model on the cleaned dataset
gls_model_cleaned = sm.GLS(y_combined_cleaned, X_combined_cleaned,␣
↪sigma=sigma_matrix_combined_cleaned)
gls_results_cleaned = gls_model_cleaned.fit()

# GLS + HAC (robust với heteroskedasticity + autocorrelation)
gls_hac = gls_results_cleaned.get_robustcov_results(cov_type='HAC', maxlags = 1)
print("\nGLS + HAC:")
print(gls_hac.summary())
```

```
GLS + HAC:
                              GLS Regression Results
================================================================================
==
Dep. Variable:     suicide_mortality_rate   R-squared:
0.363
Model:                                GLS   Adj. R-squared:
0.356
Method:                     Least Squares   F-statistic:
39.26
Date:                    Sun, 20 Apr 2025   Prob (F-statistic):
4.44e-45
Time:                            15:45:10   Log-Likelihood:
-1642.0
No. Observations:                     582   AIC:
3300.
Df Residuals:                         574   BIC:
3335.
Df Model:                               7
Covariance Type:                      HAC
================================================================================
=============================
                                          coef    std err          t
P>|t|      [0.025      0.975]
--------------------------------------------------------------------------------
-----------------------------
const                                 -77.5773     11.765     -6.594
0.000    -100.686     -54.469
log_prevalence_of_bipolar_disorder     -3.9368      0.777     -5.067
0.000      -5.463      -2.411
```

```
log_prevalence_of_anxiety_disorder          -3.6492      1.047     -3.486
0.001      -5.706      -1.593
log_prevalence_of_depression                11.8557      0.985     12.031
0.000       9.920      13.791
log_alcohol_use_disorders                    5.0860      0.454     11.204
0.000       4.194       5.978
log_current_health_expenditure_per_capita    1.6412      0.224      7.336
0.000       1.202       2.081
inflation                                   -0.0691      0.044     -1.562
0.119      -0.156       0.018
unemployment                                -0.0044      0.035     -0.128
0.898      -0.072       0.064
==============================================================================
Omnibus:                       41.966   Durbin-Watson:                   0.706
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               49.203
Skew:                           0.701   Prob(JB):                     2.07e-11
Kurtosis:                       3.248   Cond. No.                     1.09e+03
==============================================================================

Notes:
[1] Standard Errors are heteroscedasticity and autocorrelation robust (HAC)
using 1 lags and without small sample correction
[2] The condition number is large, 1.09e+03. This might indicate that there are
strong multicollinearity or other numerical problems.
```

6.3 Recheck assumptions

Ramsey Test

[ ]:
```python
import statsmodels.api as sm
from scipy.stats import chi2

def ramsey_reset_gls_df2_hac(X, y, sigma_matrix, maxlags=1):
    """
    Ramsey RESET test for GLS model with df = 2 (ŷ², ŷ³) using HAC standard␣
 ↪errors.

    Parameters:
      - X: DataFrame of independent variables (with constant already added)
      - y: Dependent variable
      - sigma_matrix: Variance-covariance matrix (diagonal)
      - maxlags: Number of lags for HAC (default is 1)

    Returns:
    - Dictionary với LR-statistic, df, p-value and conclusion
    """
    # Fit GLS original
    gls_model = sm.GLS(y, X, sigma=sigma_matrix).fit()
```

```
    y_hat = gls_model.predict(X)

    # ŷ² và ŷ³
    X_reset = X.copy()
    X_reset['y_hat_sq'] = y_hat ** 2
    X_reset['y_hat_cu'] = y_hat ** 3

    # Fit GLS reset model
    gls_reset = sm.GLS(y, X_reset, sigma=sigma_matrix).fit()

    # Calculate LR
    lr_stat = 2 * (gls_reset.llf - gls_model.llf)
    df_diff = X_reset.shape[1] - X.shape[1]

    # Calculate robust standard errors using HAC
    gls_hac = gls_reset.get_robustcov_results(cov_type='HAC', maxlags = 1)

    # Calculate p-value for the LR statistic using the robust covariance matrix
    robust_p_value = chi2.sf(lr_stat, df_diff)

    conclusion = "Wrong function" if robust_p_value < 0.05 else "Right function"

    return {
        'GLS_LR_statistic': lr_stat,
        'df': df_diff,
        'p_value': robust_p_value,
        'conclusion': conclusion,
        'robust_standard_errors': gls_hac.bse
    }
```

[ ]:
```
reset_result_df2 = ramsey_reset_gls_df2_hac(X_combined_cleaned,␣
 ↪y_combined_cleaned, sigma_matrix_combined_cleaned)
for key, value in reset_result_df2.items():
    print(f"{key}: {value}")
```

```
GLS_LR_statistic: 3.4279297443490577
df: 2
p_value: 0.18015010261590234
conclusion: Right function
robust_standard_errors: [4.09023539e+01 2.28777274e+00 1.75283713e+00
5.62366846e+00
 2.63291125e+00 8.32942852e-01 5.78682008e-02 3.43369434e-02
 5.55845823e-02 1.82120338e-03]
```

Zero Mean Test

[ ]:
```
# Fit GLS
```

```python
gls_model = sm.GLS(y_combined_cleaned, X_combined_cleaned,␣
    ↪sigma=sigma_matrix_combined_cleaned).fit()
gls_hac = gls_model.get_robustcov_results(cov_type='HAC', maxlags = 1)
resid_gls = y - gls_hac.predict(X)

# Calculate mean
mean_resid_gls = resid_gls.mean()
print("Mean of residuals (GLS):", mean_resid_gls)

# Calculate t_stat, p_value
from scipy.stats import ttest_1samp

t_stat, p_val = ttest_1samp(resid_gls, 0)
print(f"T-statistic: {t_stat:.3f}, p-value: {p_val:.3f}")
```

```
Mean of residuals (GLS): 0.16183716521616884
T-statistic: 0.888, p-value: 0.375
```