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

data = pd.read_csv('Country-data.csv')
data_dictionary = pd.read_csv('data-dictionary.csv')

print("Primeras filas del dataset principal:")
print(data.head())
print("\nDescripción del diccionario de datos:")
print(data_dictionary.head())
```

```
Primeras filas del dataset principal:
              country  child_mort  exports  health  imports
income  \
0         Afghanistan        90.2     10.0    7.58     44.9    1610

1             Albania        16.6     28.0    6.55     48.6    9930

2             Algeria        27.3     38.4    4.17     31.4   12900

3              Angola       119.0     62.3    2.85     42.9    5900

4  Antigua and Barbuda        10.3     45.5    6.03     58.9   19100


   inflation  life_expec  total_fer   gdpp
0       9.44        56.2       5.82    553
1       4.49        76.3       1.65   4090
2      16.10        76.5       2.89   4460
3      22.40        60.1       6.16   3530
4       1.44        76.8       2.13  12200

Descripción del diccionario de datos:
  Column Name                                          Description
0     country                                Name of the country
1  child_mort  Death of children under 5 years of age per 100...
2     exports  Exports of goods and services. Given as %age o...
3      health        Total health spending as %age of Total GDP
4     imports  Imports of goods and services. Given as %age o...
```

```python
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from statsmodels.stats.outliers_influence import
variance_inflation_factor
import statsmodels.api as sm

# Eliminar filas con valores nulos en X o y
data = data.dropna(subset=X.columns.tolist() + ['gdpp'])
X = data.drop(columns=['gdpp'])
y = data['gdpp']
```

```python
# Eliminar la columna `country` de las variables predictoras
X = X.drop(columns=['country'])

# Modelo
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.3, random_state=42)
X_train_sm = sm.add_constant(X_train)
model = sm.OLS(y_train, X_train_sm).fit()

print(model.summary())
```

```
                            OLS Regression Results

==============================================================================
========
Dep. Variable:                    gdpp   R-squared:
0.865
Model:                            OLS   Adj. R-squared:
0.854
Method:                 Least Squares   F-statistic:
85.38
Date:                Sun, 27 Oct 2024   Prob (F-statistic):
6.54e-43
Time:                        15:34:05   Log-Likelihood:
-1196.7
No. Observations:                 116   AIC:
2411.
Df Residuals:                     107   BIC:
2436.
Df Model:                           8

Covariance Type:            nonrobust

==============================================================================
========
                 coef    std err          t      P>|t|      [0.025
0.975]
------------------------------------------------------------------------------
--------
const      -5.005e+04   1.44e+04     -3.474      0.001   -7.86e+04   -
2.15e+04
child_mort    86.3265     46.718      1.848      0.067      -6.287
178.940
exports       72.0341     58.342      1.235      0.220     -43.621
187.690
health      1795.2826    311.317      5.767      0.000    1178.133
2412.432
imports      -25.5729     59.197     -0.432      0.667    -142.924
91.778
income         0.7712      0.055     13.949      0.000       0.662
```

```
0.881
inflation     -86.6508     65.556     -1.322      0.189    -216.607
43.306
life_expec    451.9926    187.088      2.416      0.017      81.113
822.873
total_fer     593.8797    860.548      0.690      0.492   -1112.056
2299.815
==============================================================================
========
Omnibus:                       30.855   Durbin-Watson:
1.986
Prob(Omnibus):                  0.000   Jarque-Bera (JB):
95.094
Skew:                           0.899   Prob(JB):
2.24e-21
Kurtosis:                       7.055   Cond. No.
5.48e+05
==============================================================================
========

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is
correctly specified.
[2] The condition number is large, 5.48e+05. This might indicate that
there are
strong multicollinearity or other numerical problems.
```

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

print("\nFactor de Inflación de Varianza (VIF):")
print(vif_data)
```

```
Factor de Inflación de Varianza (VIF):
     Variable        VIF
0  child_mort   8.060881
1     exports  16.008933
2      health   9.832297
3     imports  17.078102
4      income   4.282023
5   inflation   1.942327
6  life_expec  20.033757
7   total_fer  17.652689
```

Interpretación

Con un R-cuadrado de 0.865 y un R-cuadrado ajustado de 0.854, el modelo muestra un buen ajuste, explicando aproximadamente el 86% de la variabilidad en gdpp a partir de las variables predictoras. Sin embargo, varios p-valores de las variables como exports, imports, inflation y total_fer son mayores a 0.05, esto indica que no son significativas en el modelo, de todos modos, las variables health, income y life_expec presentan p-valores bajos, sugiriendo que son predictoras significativas de gdpp.

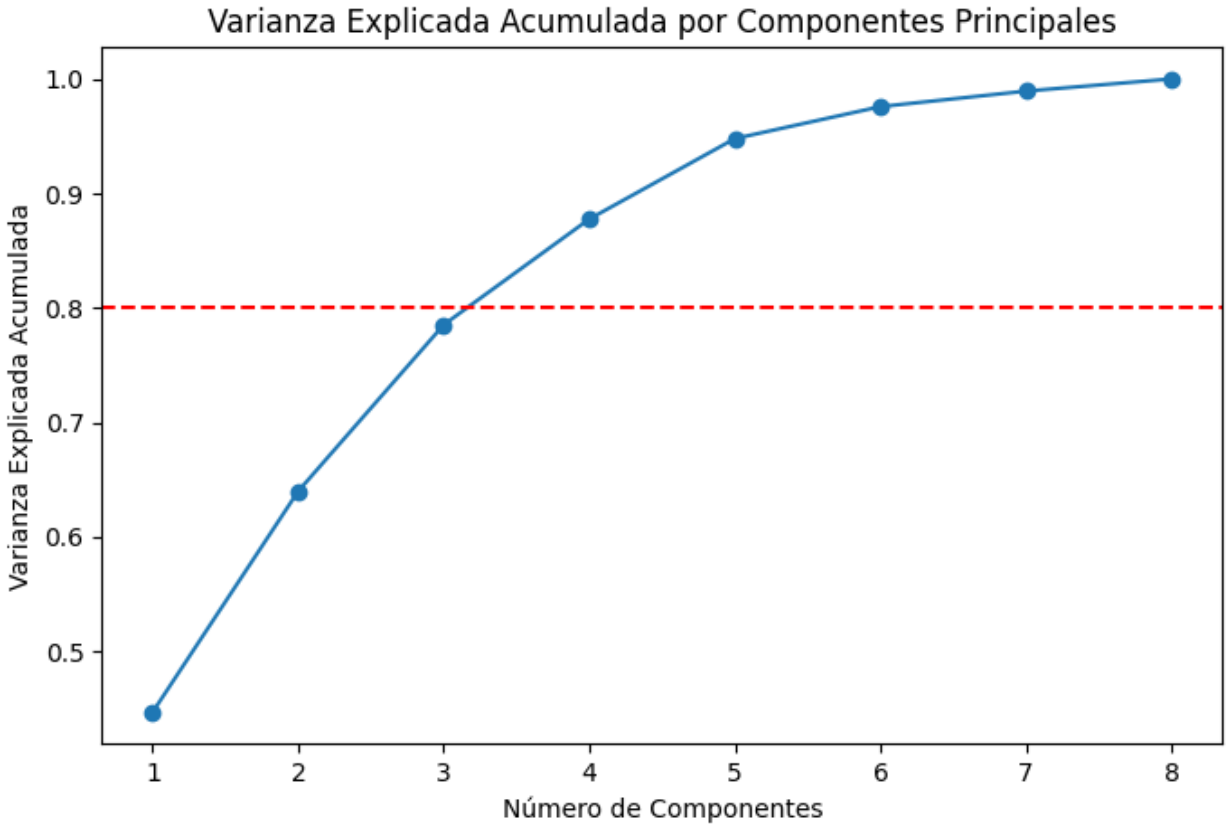
Los altos valores del VIF, especialmente en life_expec (20.03), imports (17.08), total_fer (17.65) y exports (16.01), indican una fuerte multicolinealidad entre las variables, lo que puede afectar la fiabilidad del modelo. También, los supuestos del modelo podrían no cumplirse completamente debido a esta multicolinealidad y a posibles desviaciones en los residuales.

En resumen, aunque el modelo funciona en general, la presencia de variables no significativas y alta multicolinealidad sugiere que se debe considerar la reducción de variables o aplicar técnicas como el análisis de componentes principales para mejorar la interpretabilidad y fiabilidad del modelo.

```
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
import matplotlib.pyplot as plt

scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
pca = PCA()
X_pca = pca.fit_transform(X_scaled)
explained_variance = np.cumsum(pca.explained_variance_ratio_)

print("Varianza explicada acumulada por cada componente principal:")
for i, variance in enumerate(explained_variance):
    print(f"Componente {i + 1}: {variance:.2f}")

plt.figure(figsize=(8, 5))
plt.plot(range(1, len(explained_variance) + 1), explained_variance,
marker='o')
plt.axhline(y=0.8, color='r', linestyle='--')
plt.xlabel('Número de Componentes')
plt.ylabel('Varianza Explicada Acumulada')
plt.title('Varianza Explicada Acumulada por Componentes Principales')
plt.show()
```

```
Varianza explicada acumulada por cada componente principal:
Componente 1: 0.45
Componente 2: 0.64
Componente 3: 0.79
Componente 4: 0.88
Componente 5: 0.95
Componente 6: 0.98
```

```
Componente 7: 0.99
Componente 8: 1.00
```




```
# Valores propios (varianza de cada componente principal)
print("\nValores propios de cada componente principal:")
print(pca.explained_variance_)

# Vectores propios (dirección de cada componente principal)
print("\nVectores propios (carga de cada variable en cada componente
principal):")
pca_components = pd.DataFrame(pca.components_, columns=X.columns)
print(pca_components)


Valores propios de cada componente principal:
[3.59615705 1.55324696 1.17038164 0.74324181 0.56558787 0.22483356
 0.10919005 0.08555385]

Vectores propios (carga de cada variable en cada componente
principal):
   child_mort   exports    health   imports    income  inflation
life_expec  \
0    0.472880 -0.308396 -0.144568 -0.194640 -0.386787   0.220475   -
0.464191
```

```
1    0.214124  0.608374 -0.241608  0.661131  0.031207   0.005771   -
0.237343
2    0.099988 -0.146037  0.647403  0.285257 -0.247776  -0.615777   -
0.158082
3    0.115187  0.101508  0.680156  0.056361  0.315029   0.621292
0.003857
4    0.297170  0.057511 -0.058959 -0.315368  0.728256  -0.417865   -
0.091366
5   -0.203321  0.053447 -0.013921  0.036543 -0.178963  -0.063577
0.600435
6    0.135133  0.696419  0.182673 -0.569245 -0.351358  -0.086150
0.020344
7    0.747904 -0.109448 -0.044089  0.125062 -0.054303   0.009900
0.577846

   total_fer
0   0.456952
1   0.176702
2   0.051085
3   0.159304
4   0.303536
5   0.746781
6  -0.089684
7  -0.272258
```

El análisis de Componentes Principales (PCA) buscaba reducir la dimensionalidad de las variables predictoras para mejorar el modelo y reducir problemas de multicolinealidad. El PCA reestructura los datos en nuevas variables no correlacionadas llamadas componentes principales, ordenadas según la cantidad de varianza que explican. En este caso, los primeros cuatro componentes principales explican el 88% de la varianza total, superando el umbral del 80%, por eso decidí trabajar con ellos para el proceso anterior.

Los valores propios de cada componente muestran cuánta varianza aporta cada uno de ellos, el primer componente tiene un valor propio de 3.60, indicando que por sí solo explica el 45% de la varianza total. La carga de cada variable en los vectores propios ayuda a interpretar la dirección de cada componente. En el primer componente (PC1), destacan las altas cargas de child_mort, income, life_expec, y total_fer. El segundo componente (PC2) tiene fuertes cargas en exports e imports. Los componentes adicionales (PC3 y PC4) muestran altas cargas en health, inflation, y income. En cuanto a la agrupación de los datos, los componentes principales obtenidos muestran que existen patrones que distinguen a los países en función de estas dimensiones, aunque no son categorías rígidas sino que describen distintas dimensiones que contribuyen al gdpp.

En conclusión, el PCA logró eficientar el modelo al reducir las variables predictoras a cuatro componentes principales que conservan la mayoría de la varianza. Esto hace mas fácil la interpretación del modelo.

```
n_components = 4
X_pca_selected = X_pca[:, :n_components]
```

```
component_names = ['Wellbeing', 'Trade', 'Health', 'Economics']
X_pca_df = pd.DataFrame(X_pca_selected, columns=component_names)
X_pca_df['gdpp'] = y.values

X_train_pca, X_test_pca, y_train_pca, y_test_pca = train_test_split(
    X_pca_df.drop(columns=['gdpp']), X_pca_df['gdpp'], test_size=0.3,
random_state=42)

X_train_pca_sm = sm.add_constant(X_train_pca)
model_pca = sm.OLS(y_train_pca, X_train_pca_sm).fit()

print(model_pca.summary())
```

```
                            OLS Regression Results

==============================================================================
========
Dep. Variable:                   gdpp   R-squared:
0.618
Model:                            OLS   Adj. R-squared:
0.604
Method:                 Least Squares   F-statistic:
44.90
Date:                Sun, 27 Oct 2024   Prob (F-statistic):
2.25e-22
Time:                        17:04:04   Log-Likelihood:
-1256.8
No. Observations:                 116   AIC:
2524.
Df Residuals:                     111   BIC:
2537.
Df Model:                           4

Covariance Type:            nonrobust

==============================================================================
========
                 coef    std err          t      P>|t|      [0.025
0.975]
------------------------------------------------------------------------------
--------
const       1.407e+04   1177.197     11.951      0.000    1.17e+04
1.64e+04
Wellbeing  -7422.0067    627.480    -11.828      0.000   -8665.399   -
6178.614
Trade        877.5664   1050.378      0.835      0.405   -1203.827
2958.960
Health      -291.6240   1013.925     -0.288      0.774   -2300.784
1717.536
Economics   7496.8194   1327.916      5.646      0.000    4865.465
```

```
1.01e+04
==============================================================================
========
Omnibus:                       23.467   Durbin-Watson:
2.175
Prob(Omnibus):                  0.000   Jarque-Bera (JB):
36.114
Skew:                           0.954   Prob(JB):
1.44e-08
Kurtosis:                       4.958   Cond. No.
2.26
==============================================================================
========

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

Ecuaciones de transformación lineal:

Componente Principal 1 (Wellbeing) Este componente está fuertemente influenciado por child_mort, income, life_expec y total_fer.

Ecuación:

Wellbeing

$0.473 \cdot Z$ child_mort $- 0.387 \cdot Z$ income $- 0.464 \cdot Z$ life_expec

- $0.457 \cdot Z$ total_fer Wellbeing=0.473·Z child_mort −0.387·Z income −0.464·Z life_expec +0.457·Z total_fer

Componente Principal 2 (Trade) Este componente está dominado por las variables exports y imports.

Ecuación:

Trade

$0.608 \cdot Z$ exports

- $0.661 \cdot Z$ imports Trade=0.608·Z exports +0.661·Z imports

Componente Principal 3 (Health) Las variables más influyentes en este componente son health e inflation.

Ecuación:

Health

$0.647 \cdot Z$ health $- 0.616 \cdot Z$ inflation Health=0.647·Z health −0.616·Z inflation

Componente Principal 4 (Economics) Este componente está influenciado por health, inflation e income.

Ecuación:

Economics

$0.680 \cdot Z$ health

- $0.621 \cdot Z$ inflation
- $0.315 \cdot Z$ income Economics=0.680·Z health +0.621·Z inflation +0.315·Z income

El modelo de regresión con los cuatro componentes principales tiene un R-cuadrado de 0.618 y un R-cuadrado ajustado de 0.604, lo que significa que estos componentes explican alrededor del 61.8% de la variabilidad en gdpp. Aunque esto es menor que el modelo inicial, la reducción de variables aporta un modelo más simple y menos susceptible a problemas de multicolinealidad. La constante y los coeficientes de Wellbeing y Economics son significativos ($p < 0.05$), indicando que estos componentes principales están estrechamente relacionados con el gdpp.

Wellbeing tiene un coeficiente negativo significativo, lo que sugiere que los factores asociados a este componente (como child_mort, income, life_expec, y total_fer) tienen una relación inversa con el gdpp. Economics, en cambio, muestra un coeficiente positivo significativo, lo que indica una relación directa con el gdpp.

En general, este modelo con PCA proporciona una interpretación más sencilla del gdpp, sacrificando algo de precisión a cambio de simplificación y estabilidad en el modelo.

Comparación de ambos modelos

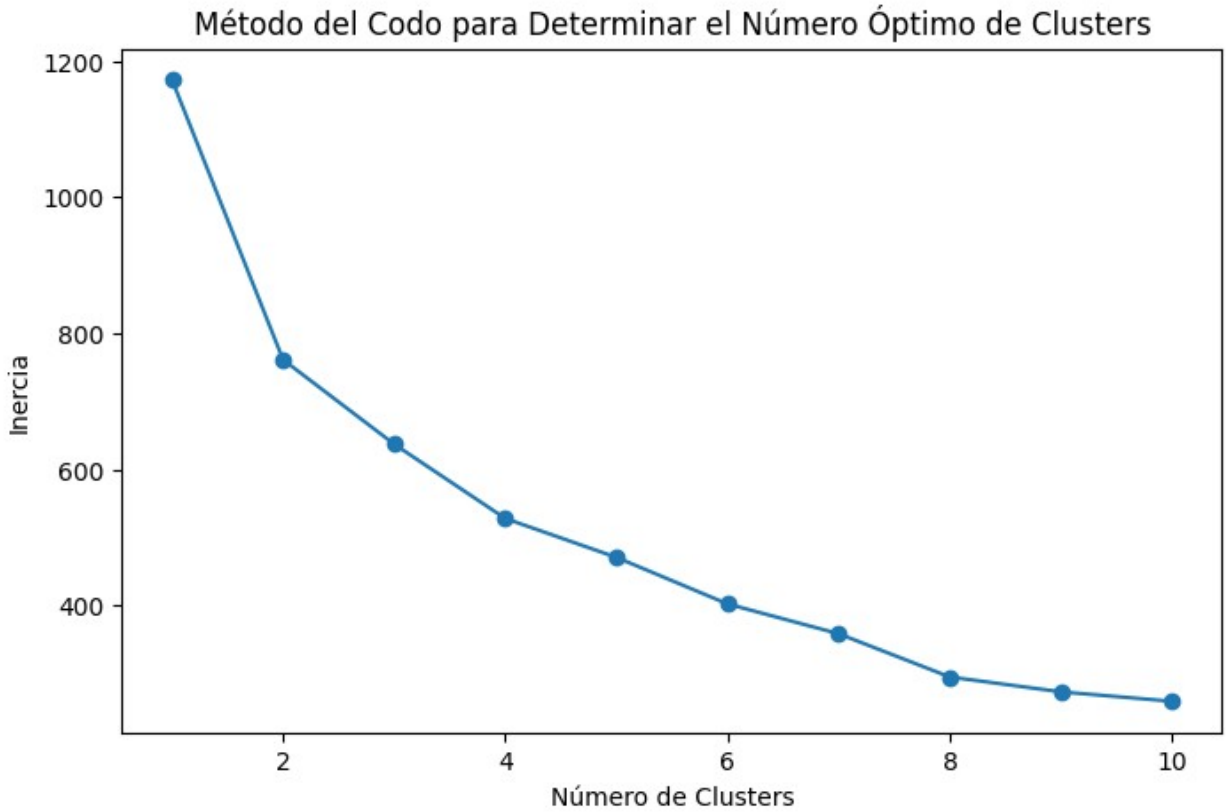
Al comparar ambos modelos de regresión, se nota que el modelo original, que incluye todas las variables predictoras, tiene un alto poder explicativo con un R-cuadrado de 0.865 y un R-cuadrado ajustado de 0.854, indicando que explica aproximadamente el 86.5% de la variabilidad en gdpp.

Sin embargo, presenta problemas significativos de multicolinealidad, como lo demuestran los altos valores del VIF en variables como life_expec, imports, total_fer y exports. Esta multicolinealidad puede afectar la estabilidad y confiabilidad del modelo, dificultando la interpretación precisa de la influencia individual de cada variable.

En contraste, el modelo que aplica el Análisis de Componentes Principales (PCA) reduce la complejidad al condensar las variables en cuatro componentes principales que explican el 88% de la varianza total. Aunque su R-cuadrado es menor (0.618), ofrece un modelo más simple y conciso que elimina la multicolinealidad. Los componentes denominados "Wellbeing" y "Economics" resultan significativos, facilitando una interpretación más clara y global de las

dimensiones que afectan al gdpp. Aunque se sacrifica cierta capacidad explicativa en comparación con el modelo original, el modelo con PCA proporciona una solución más robusta y generalizable, abordando eficazmente los problemas de multicolinealidad y mejorando la interpretabilidad general del modelo.

```
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt

inertia = []
X_cluster = X_pca_df.drop(columns=['gdpp']).values  # Usar los
componentes principales seleccionados

for i in range(1, 11):
    kmeans = KMeans(n_clusters=i, random_state=42)
    kmeans.fit(X_cluster)
    inertia.append(kmeans.inertia_)

plt.figure(figsize=(8, 5))
plt.plot(range(1, 11), inertia, marker='o')
plt.xlabel('Número de Clusters')
plt.ylabel('Inercia')
plt.title('Método del Codo para Determinar el Número Óptimo de
Clusters')
plt.show()
```



```
n_clusters = 2  # Basándome en el procedimiento anterior

kmeans = KMeans(n_clusters=n_clusters, random_state=42)
X_pca_df['Cluster'] = kmeans.fit_predict(X_cluster)

print(X_pca_df[['Cluster']].head())
```

```
   Cluster
0        0
1        1
2        1
3        0
4        1
```

```
import seaborn as sns

plt.figure(figsize=(10, 6))
sns.scatterplot(x=X_pca_df['PC1'], y=X_pca_df['PC2'],
hue=X_pca_df['Cluster'], palette='viridis', s=100, alpha=0.7)
plt.title('Visualización de Clusters en los Primeros Dos Componentes
Principales')
plt.xlabel('Componente Principal 1')
plt.ylabel('Componente Principal 2')
plt.legend(title='Cluster')
plt.show()
```