# RegresiónLineal

March 24, 2026

Tarea 1 - Inciso B

Dara Eugenia Gama Sandoval

Maestría en Cómputo Estadístico | Modelos Estadisticos

Objetivo: documentar, con rigor y claridad, el proceso de modelado y sus hallazgos.

## 1 Instrucciones

Para cada inciso se debe: * Ajustar un modelo de regresión lineal * Realizar un análisis del ajuste del modelo. * Obtener las pruebas de hipótesis * Los estadísticos * Gráficas vistas en clase. * Nota: Para validar correlación, es suficiente usar dos pruebas. * Buscar outliers * Transformar los datos si es necesario (por ejemplo, transformación de Box-Cox) * Ver si las variables son significativas y la distancia de Cook * Analizar los residuales. * Comentar brevemente los hallazgos clave y su interpretación. * Escribir el modelo final, y hacer una predicción con un nuevo valor, así como su intervalo de confianza.

Además, priorizar la reproducibilidad y la interpretación práctica de resultados.

[1]:
```
# Correr este comando para instalar las dependencias necesarias
! pip install pandas kagglehub scikit-learn matplotlib seaborn statsmodels
```

```
Requirement already satisfied: pandas in /Users/daragama/Documents/Master25/2_Mo
delosEstadisticos/modelos_estadisticos/lib/python3.12/site-packages (3.0.0)
Requirement already satisfied: kagglehub in /Users/daragama/Documents/Master25/2
_ModelosEstadisticos/modelos_estadisticos/lib/python3.12/site-packages (0.4.2)
Requirement already satisfied: scikit-learn in /Users/daragama/Documents/Master2
5/2_ModelosEstadisticos/modelos_estadisticos/lib/python3.12/site-packages
(1.8.0)
Requirement already satisfied: matplotlib in /Users/daragama/Documents/Master25/
2_ModelosEstadisticos/modelos_estadisticos/lib/python3.12/site-packages (3.10.8)
Requirement already satisfied: seaborn in /Users/daragama/Documents/Master25/2_M
odelosEstadisticos/modelos_estadisticos/lib/python3.12/site-packages (0.13.2)
Requirement already satisfied: statsmodels in /Users/daragama/Documents/Master25
/2_ModelosEstadisticos/modelos_estadisticos/lib/python3.12/site-packages
(0.14.6)
Requirement already satisfied: numpy>=1.26.0 in /Users/daragama/Documents/Master
25/2_ModelosEstadisticos/modelos_estadisticos/lib/python3.12/site-packages (from
pandas) (2.4.2)
```

```
Requirement already satisfied: python-dateutil>=2.8.2 in /Users/daragama/Documen
ts/Master25/2_ModelosEstadisticos/modelos_estadisticos/lib/python3.12/site-
packages (from pandas) (2.9.0.post0)
Requirement already satisfied: kagglesdk<1.0,>=0.1.14 in /Users/daragama/Documen
ts/Master25/2_ModelosEstadisticos/modelos_estadisticos/lib/python3.12/site-
packages (from kagglehub) (0.1.15)
Requirement already satisfied: packaging in /Users/daragama/Documents/Master25/2
_ModelosEstadisticos/modelos_estadisticos/lib/python3.12/site-packages (from
kagglehub) (26.0)
Requirement already satisfied: pyyaml in /Users/daragama/Documents/Master25/2_Mo
delosEstadisticos/modelos_estadisticos/lib/python3.12/site-packages (from
kagglehub) (6.0.3)
Requirement already satisfied: requests in /Users/daragama/Documents/Master25/2_
ModelosEstadisticos/modelos_estadisticos/lib/python3.12/site-packages (from
kagglehub) (2.32.5)
Requirement already satisfied: tqdm in /Users/daragama/Documents/Master25/2_Mode
losEstadisticos/modelos_estadisticos/lib/python3.12/site-packages (from
kagglehub) (4.67.2)
Requirement already satisfied: protobuf in /Users/daragama/Documents/Master25/2_
ModelosEstadisticos/modelos_estadisticos/lib/python3.12/site-packages (from
kagglesdk<1.0,>=0.1.14->kagglehub) (6.33.5)
Requirement already satisfied: scipy>=1.10.0 in /Users/daragama/Documents/Master
25/2_ModelosEstadisticos/modelos_estadisticos/lib/python3.12/site-packages (from
scikit-learn) (1.17.0)
Requirement already satisfied: joblib>=1.3.0 in /Users/daragama/Documents/Master
25/2_ModelosEstadisticos/modelos_estadisticos/lib/python3.12/site-packages (from
scikit-learn) (1.5.3)
Requirement already satisfied: threadpoolctl>=3.2.0 in /Users/daragama/Documents
/Master25/2_ModelosEstadisticos/modelos_estadisticos/lib/python3.12/site-
packages (from scikit-learn) (3.6.0)
Requirement already satisfied: contourpy>=1.0.1 in /Users/daragama/Documents/Mas
ter25/2_ModelosEstadisticos/modelos_estadisticos/lib/python3.12/site-packages
(from matplotlib) (1.3.3)
Requirement already satisfied: cycler>=0.10 in /Users/daragama/Documents/Master2
5/2_ModelosEstadisticos/modelos_estadisticos/lib/python3.12/site-packages (from
matplotlib) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /Users/daragama/Documents/Ma
ster25/2_ModelosEstadisticos/modelos_estadisticos/lib/python3.12/site-packages
(from matplotlib) (4.61.1)
Requirement already satisfied: kiwisolver>=1.3.1 in /Users/daragama/Documents/Ma
ster25/2_ModelosEstadisticos/modelos_estadisticos/lib/python3.12/site-packages
(from matplotlib) (1.4.9)
Requirement already satisfied: pillow>=8 in /Users/daragama/Documents/Master25/2
_ModelosEstadisticos/modelos_estadisticos/lib/python3.12/site-packages (from
matplotlib) (12.1.0)
Requirement already satisfied: pyparsing>=3 in /Users/daragama/Documents/Master2
5/2_ModelosEstadisticos/modelos_estadisticos/lib/python3.12/site-packages (from
matplotlib) (3.3.2)
```

```
Requirement already satisfied: patsy>=0.5.6 in /Users/daragama/Documents/Master2
5/2_ModelosEstadisticos/modelos_estadisticos/lib/python3.12/site-packages (from
statsmodels) (1.0.2)
Requirement already satisfied: six>=1.5 in /Users/daragama/Documents/Master25/2_
ModelosEstadisticos/modelos_estadisticos/lib/python3.12/site-packages (from
python-dateutil>=2.8.2->pandas) (1.17.0)
Requirement already satisfied: charset_normalizer<4,>=2 in /Users/daragama/Docum
ents/Master25/2_ModelosEstadisticos/modelos_estadisticos/lib/python3.12/site-
packages (from requests->kagglehub) (3.4.4)
Requirement already satisfied: idna<4,>=2.5 in /Users/daragama/Documents/Master2
5/2_ModelosEstadisticos/modelos_estadisticos/lib/python3.12/site-packages (from
requests->kagglehub) (3.11)
Requirement already satisfied: urllib3<3,>=1.21.1 in /Users/daragama/Documents/M
aster25/2_ModelosEstadisticos/modelos_estadisticos/lib/python3.12/site-packages
(from requests->kagglehub) (2.6.3)
Requirement already satisfied: certifi>=2017.4.17 in /Users/daragama/Documents/M
aster25/2_ModelosEstadisticos/modelos_estadisticos/lib/python3.12/site-packages
(from requests->kagglehub) (2026.1.4)

[notice] A new release of pip is
available: 25.3 -> 26.0.1
[notice] To update, run:
pip install --upgrade pip
```

# 2 B) Multivariada

Buscar datos donde la variable de respuesta tome valores en los reales, y haya al menos 2 variables predictoras, tomando al menos dos niveles.

## 2.1 Descargar datos

Usaremos la misma base de datos que en el caso univariado agregando más variables categóricas.

[2]:
```python
import kagglehub

# Download latest version
path = kagglehub.dataset_download("frederickfelix/diabetes-mexico-data-set")

print("Path to dataset files:", path)
```

```
/Users/daragama/Documents/Master25/2_ModelosEstadisticos/modelos_estadisticos/li
b/python3.12/site-packages/tqdm/auto.py:21: TqdmWarning: IProgress not found.
Please update jupyter and ipywidgets. See
https://ipywidgets.readthedocs.io/en/stable/user_install.html
  from .autonotebook import tqdm as notebook_tqdm

Path to dataset files:
/Users/daragama/.cache/kagglehub/datasets/frederickfelix/diabetes-mexico-data-
set/versions/2
```

[3]:
```
! ls /Users/daragama/.cache/kagglehub/datasets/frederickfelix/
  ↪diabetes-mexico-data-set/versions/2
```

```
Diabetes_Mexico.csv
```

[4]:
```
# Read CSV file
import pandas as pd

path = "/Users/daragama/.cache/kagglehub/datasets/frederickfelix/
  ↪diabetes-mexico-data-set/versions/2"
df_ = pd.read_csv(path + "/Diabetes_Mexico.csv")
print(df_.head())
print(df_["sexo"].value_counts())
```

```
         folio_i         folio_int    sexo  edad          Ciudad   Peso  \
0  2024_01001010  2024_01001010_01   Mujer  89.0  AGUASCALIENTES    NaN
1  2024_01001012  2024_01001012_01   Mujer  71.0  AGUASCALIENTES    NaN
2  2024_01001016  2024_01001016_02   Mujer  57.0  AGUASCALIENTES  80,25
3  2024_01001016  2024_01001016_05   Mujer  20.0  AGUASCALIENTES   68,6
4  2024_01001031  2024_01001031_01  Hombre  74.0  AGUASCALIENTES    NaN

  Estatura   imc  muestra_suero ac_urico  … creat glu_suero  insulina  \
0      NaN   NaN            1.0      4,1  …  0,62     166.0       4,7
1      NaN  32.0            1.0      4,3  …  0,64      98.0       9,6
2    170,6   NaN            1.0      4,9  …   0,6     128.0       4,7
3    166,2   NaN            1.0      4,9  …  0,63     112.0      10,1
4      NaN   NaN            1.0      6,4  …  1,09      97.0      10,2

    trig hb1ac  ponde_venosa estrato  est_sel   ponde_hemo riesgo_diabetes_cat
0  107.0   NaN   2245,917738     3.0     13.0  2888,530238                   2
1  132.0   NaN   2245,917738     3.0     13.0  2888,530238                   1
2  263.0   6.0   4047,445276     3.0     13.0          NaN                   2
3  238.0   NaN   13079,14232     3.0     13.0  13776,80526                   2
4  135.0   NaN   5292,158204     3.0     13.0  5824,779445                   0

[5 rows x 24 columns]
sexo
Mujer     1547
Hombre    1002
Name: count, dtype: int64
```

### 2.2 Limpieza

Se unifican nombres de columnas a snake_case y se eliminan registros con valores nulos en variables clave, garantizando consistencia mínima para el modelado posterior.

[5]:
```
df = df_.rename(columns=lambda x: x.lower().replace(" ", "_")) # to snake_case
```

```
# Remover Valores nulos
df = df[df["sexo"].notna()] # No se encontraron datos nulos en esta columna
df = df[df["peso"].notna()] # Se encontraron datos nulos en esta columna
df = df[df["riesgo_diabetes_cat"].notna()] # No se encontraron datos nulos en␣
 ↪esta columna
df = df[df["estrato"].notna()] # No se encontraron datos nulos en esta columna
df = df[df["edad"].notna()] # Algunos datos nulos en esta columna
df = df[df["estatura"].notna()] # No se ecnontraron datos nulos en esta columna

# Convertir sexo a variable binaria
# Convertir sexo a variable binaria
df["sexo"] = df["sexo"].map({"Hombre": 1, "Mujer": 0}).astype("int")

# Convertir peso a numérico
df["peso"] = df["peso"].str.replace(",", ".") # Reemplazar comas por puntos
df["peso"] = pd.to_numeric(df["peso"])

# Convertir edad en variable categórica
df["edad"] = pd.to_numeric(df["edad"], errors="coerce")
df["edad_4cat"] = pd.qcut(df["edad"], q=4, labels=[1, 2, 3, 4]).astype("int")

# Convertir edad en variable categórica
df["estatura"] = df["estatura"].str.replace(",", ".") # Reemplazar comas por␣
 ↪puntos
df["estatura"] = pd.to_numeric(df["estatura"], errors="coerce")
df["estatura_4cat"] = pd.qcut(df["estatura"], q=4, labels=[1, 2, 3, 4]).
 ↪astype("int")

print(df["sexo"].value_counts(), "\n")
print(df["riesgo_diabetes_cat"].value_counts(), "\n")
print(df["estrato"].value_counts(), "\n")
# Bordes numéricos de los bins para edad
print(df["edad_4cat"].value_counts(), "\n")
_, bins = pd.qcut(df["edad"], q=4, retbins=True, duplicates="drop")
print("Bordes (edad):", bins)
# Bordes numéricos de los bins para estatura
print(df["estatura_4cat"].value_counts(), "\n")
_, bins = pd.qcut(df["estatura"], q=4, retbins=True, duplicates="drop")
print("Bordes (estatura):", bins)
```

```
sexo
0    1122
1     693
Name: count, dtype: int64

riesgo_diabetes_cat
0    1314
```

```
2     492
1       9
Name: count, dtype: int64

estrato
3.0    721
2.0    551
1.0    543
Name: count, dtype: int64

edad_4cat
1    503
3    473
2    433
4    406
Name: count, dtype: int64

Bordes (edad): [-7975.    31.    40.    50.    60.]
estatura_4cat
1    457
2    457
4    454
3    447
Name: count, dtype: int64

Bordes (estatura): [126.5  152.5  158.9  165.65 222.2 ]
```

Se transformaron las variables `edad` y `estatura` en nuevas variables categóricas `edad_4cat` y `estatura_4cat` utilizando el método `pd.qcut`, que divide la distribución en 4 cuartiles. Los valores de esta nueva variable se asignaron como categorías numeradas del 1 al 4.

Estas transformaciones permitieron cumplir con el requisito de incluir al menos dos variables predictoras categóricas con al menos dos niveles, asegurando además la limpieza y consistencia de los datos para el análisis posterior.

Los datos están aproximadamente igual distribuidos excepto por `riesgo_de_diabetes_cat`pues una de sus categorías solo tiene 9 datos. Veremos que se puede hacer.

### 2.3 Objetivo

Predecir su peso:

$$Y = \beta_0 + \beta_i x_i + \epsilon$$

donde: - $Y$ = Variable de respuesta: `peso` - $x_i$ = Variables predictoras: - `sexo` (codificada como 0 para Mujer, 1 para Hombre) - `riesgo_diabetes_cat` - `estrato` - `edad_4cat` - `estatura_4cat` - $\beta_0$ = intercepto - $\beta_i$ = coeficientes - $\epsilon$ = error

## 2.4 Ajuste de Modelo

En esta ocasión usaremos directamente la librería `statsmodels` para ajustar el modelo de regresión lineal y obtener los residuales.

[6]:
```
import numpy as np
import statsmodels.api as sm
import statsmodels.formula.api as smf


model_formula = smf.ols("peso ~ C(sexo) + C(riesgo_diabetes_cat) + C(estrato) +␣
 ↪C(edad_4cat) + C(estatura_4cat)", data=df)
results = model_formula.fit()

y = df["peso"].values
y_pred = results.predict(df)




print(results.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                   peso   R-squared:                       0.231
Model:                            OLS   Adj. R-squared:                  0.226
Method:                 Least Squares   F-statistic:                     49.12
Date:                Wed, 04 Mar 2026   Prob (F-statistic):           1.25e-94
Time:                        20:10:56   Log-Likelihood:                -7869.4
No. Observations:                1815   AIC:                         1.576e+04
Df Residuals:                    1803   BIC:                         1.583e+04
Df Model:                          11
Covariance Type:            nonrobust
================================================================================
===============
                                   coef    std err          t      P>|t|
[0.025      0.975]
--------------------------------------------------------------------------------
---------------
Intercept                       58.8330      1.292     45.524      0.000
56.298      61.368
C(sexo)[T.1]                    -3.9414      1.220     -3.231      0.001
-6.334      -1.549
C(riesgo_diabetes_cat)[T.1]     -0.0679      6.213     -0.011      0.991
-12.253      12.117
C(riesgo_diabetes_cat)[T.2]      6.1768      1.022      6.042      0.000
4.172       8.182
C(estrato)[T.2.0]                3.1415      1.127      2.788      0.005
```

```
0.931       5.351
C(estrato)[T.3.0]              1.6784      1.079      1.556      0.120
-0.437       3.794
C(edad_4cat)[T.2]              4.5792      1.225      3.738      0.000
2.176       6.982
C(edad_4cat)[T.3]              4.1446      1.214      3.414      0.001
1.764       6.526
C(edad_4cat)[T.4]              3.6171      1.303      2.777      0.006
1.062       6.172
C(estatura_4cat)[T.2]          8.6442      1.258      6.873      0.000
6.178      11.111
C(estatura_4cat)[T.3]         15.1095      1.382     10.933      0.000
12.399      17.820
C(estatura_4cat)[T.4]         28.8748      1.647     17.527      0.000
25.644      32.106
==============================================================================
Omnibus:                     1090.199   Durbin-Watson:                   1.945
Prob(Omnibus):                  0.000   Jarque-Bera (JB):            17342.211
Skew:                           2.516   Prob(JB):                         0.00
Kurtosis:                      17.282   Cond. No.                         19.7
==============================================================================

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

En este caso el intercepto $\beta_0$ representa el peso esperado cuando - `sexo`: Mujer - `riesgo_diabetes_cat`: 0 - `estrato`: 1 - `edad_4cat`: 1 - `estatura_4cat`: 1

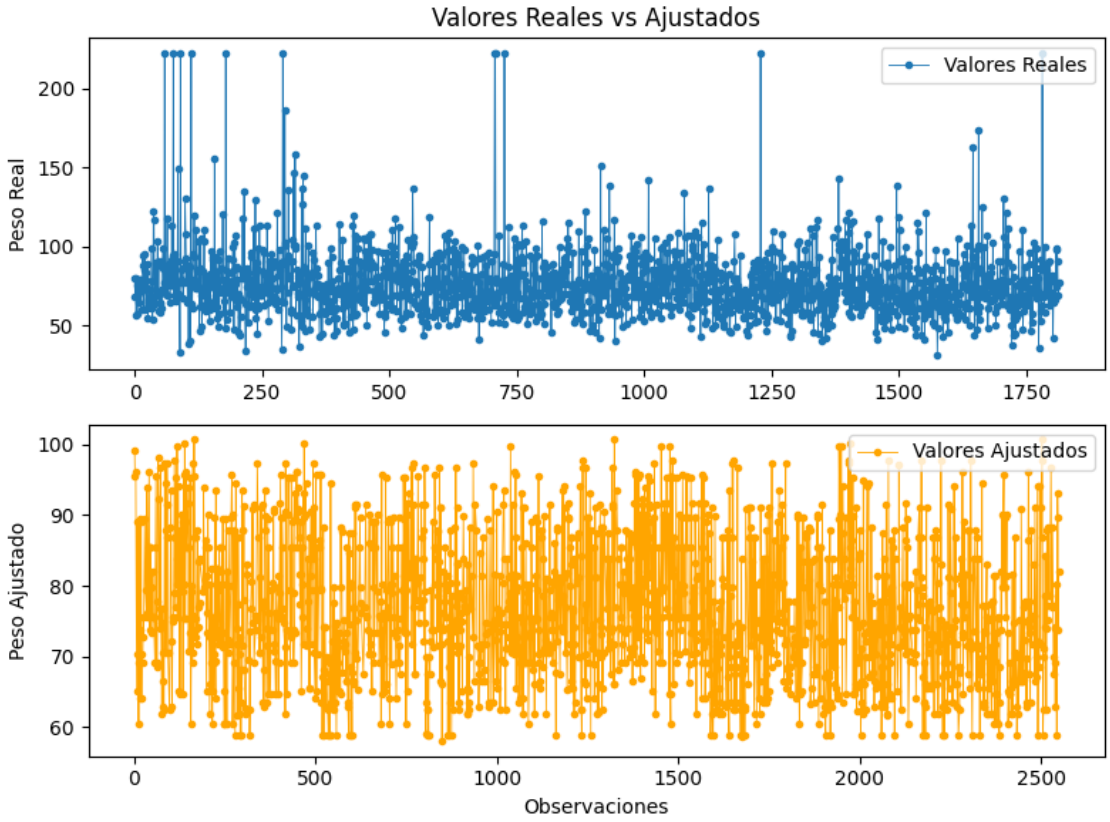
Por ejemplo, $\beta_3$ `C(riesgo_diabetes_cat)[T.2]` representa: - `sexo`: Mujer - `riesgo_diabetes_cat`: 2 - `estrato`: 1 - `edad_4cat`: 1 - `estatura_4cat`: 1

Algunas observaciones: - Como es de esperarse los coeficientes correspondientes a la estatura tienen magnitudes bastante altas - Tenemos un $R^2 = 0.231$, es decir que logramos capturar el 23.1% de la estructura de los datos - Los intervalos de confianza de `rieso_diabetes_cat`(T2), `estrato`(T3) están capturando el cero. - Tenemos un F muy "alto", lo que indica que estamos sacando información real del peso y no solo ruido.

**Gráfica Valores Reales vs Ajustados**

[7]:
```
import matplotlib.pyplot as plt

plt.figure(figsize=(8, 6))
plt.subplot(2, 1, 1)
plt.plot(y, 'o-', label='Valores Reales', linewidth=0.6, markersize=3)
plt.ylabel('Peso Real')
plt.title('Valores Reales vs Ajustados')
plt.legend()

plt.subplot(2, 1, 2)
```

```
plt.plot(y_pred, 'o-', color='orange', label='Valores Ajustados', linewidth=0.
  ↪6, markersize=3)
plt.ylabel('Peso Ajustado')
plt.xlabel('Observaciones')
plt.legend()

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




Vemos que la mayoría de los valores ajustados se encuentran entre 60-100, y a simple vista es algo complicado notar si dentro de ese rango hay similaridad con los valores reales.

## 2.5 Diagnóstico del Modelo

Aunque el poder predictivo es limitado con una sola variable binaria, realizamos el diagnóstico para examinar supuestos, detectar influencias y reforzar la interpretación de los resultados.

### 2.5.1 Residuales

Calculamos los residuales y su varianza por medio de:

$$e_i = Y_i - \hat{Y}_i$$

y

$$\hat{\sigma}^2 = \frac{\sum_{i=1}^{n} e_i^2}{n - p - 1}$$

donde: - $e_i$ es el residual para la observación $i$ - $Y_i$ es el valor observado - $\hat{Y}_i$ es el valor calculado - $n$ es el número total de observaciones - $p$ es el número de variables predictoras

El denominador refleja los grados de libertad residuales. Estos cálculos se interpretan bajo los supuestos clásicos de linealidad, independencia y varianza constante.

[8]:
```
residuals = y - y_pred

# Varianza de los residuales
residual_sum_squares = np.sum(residuals**2) # RSS
n = len(y)
p = df.shape[1]
sigma_squared = residual_sum_squares / (n - p - 1)
sigma = np.sqrt(sigma_squared)

print(f"Error Estándar Residual: {sigma:.4f}")
print(f"Varianza residual: {sigma_squared:.4f}")
print("\n")
```

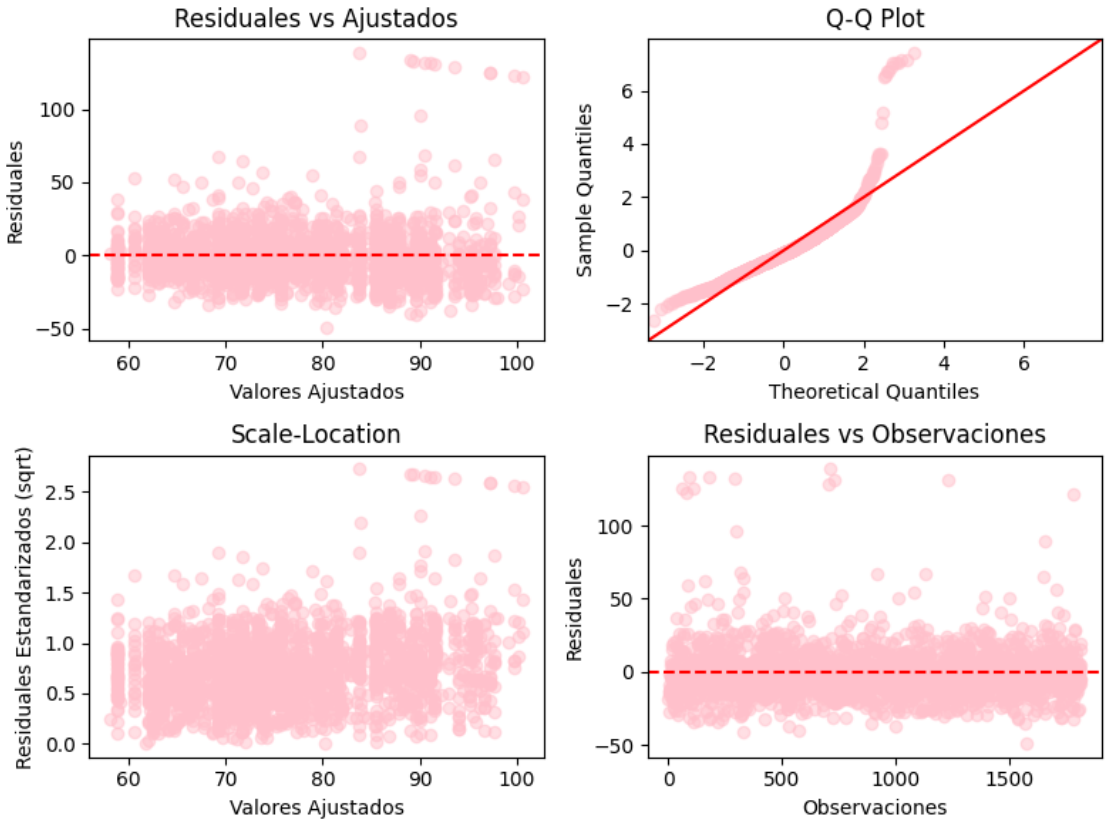
```
Error Estándar Residual: 18.6213
Varianza residual: 346.7515
```

El error no es tan alto, pero la varianza es considerablemente alta.

### 2.5.2 Gráficas de Diagnóstico de Residuales

[9]:
```
from statsmodels.graphics.gofplots import ProbPlot

fig, axes = plt.subplots(2, 2, figsize=(8, 6))
color = 'pink'
# Residuales vs Ajustados
axes[0, 0].scatter(y_pred, residuals, alpha=0.5, color=color)
axes[0, 0].axhline(y=0, color='r', linestyle='--')
axes[0, 0].set_xlabel('Valores Ajustados')
axes[0, 0].set_ylabel('Residuales')
axes[0, 0].set_title('Residuales vs Ajustados')

standardized_residuals = residuals / sigma
```

```
# Q-Q plot
ProbPlot(standardized_residuals).qqplot(ax=axes[0, 1], line='45', alpha=0.5,␣
 ↪markerfacecolor=color, markeredgecolor=color)
axes[0, 1].set_title('Q-Q Plot')

# Residuales Estandarizados vs Ajustados
axes[1, 0].scatter(y_pred, np.sqrt(np.abs(standardized_residuals)), alpha=0.5,␣
 ↪color=color)
axes[1, 0].set_xlabel('Valores Ajustados')
axes[1, 0].set_ylabel('Residuales Estandarizados (sqrt)')
axes[1, 0].set_title('Scale-Location')

# Residuales vs Leverage
axes[1, 1].scatter(range(n), residuals, alpha=0.5, color=color)
axes[1, 1].axhline(y=0, color='r', linestyle='--')
axes[1, 1].set_xlabel('Observaciones')
axes[1, 1].set_ylabel('Residuales')
axes[1, 1].set_title('Residuales vs Observaciones')

plt.tight_layout()
```



En ambas gráficas de valores ajustados contra residuales se pueden observar claramente los patrones correspondientes a las variables categóricas.

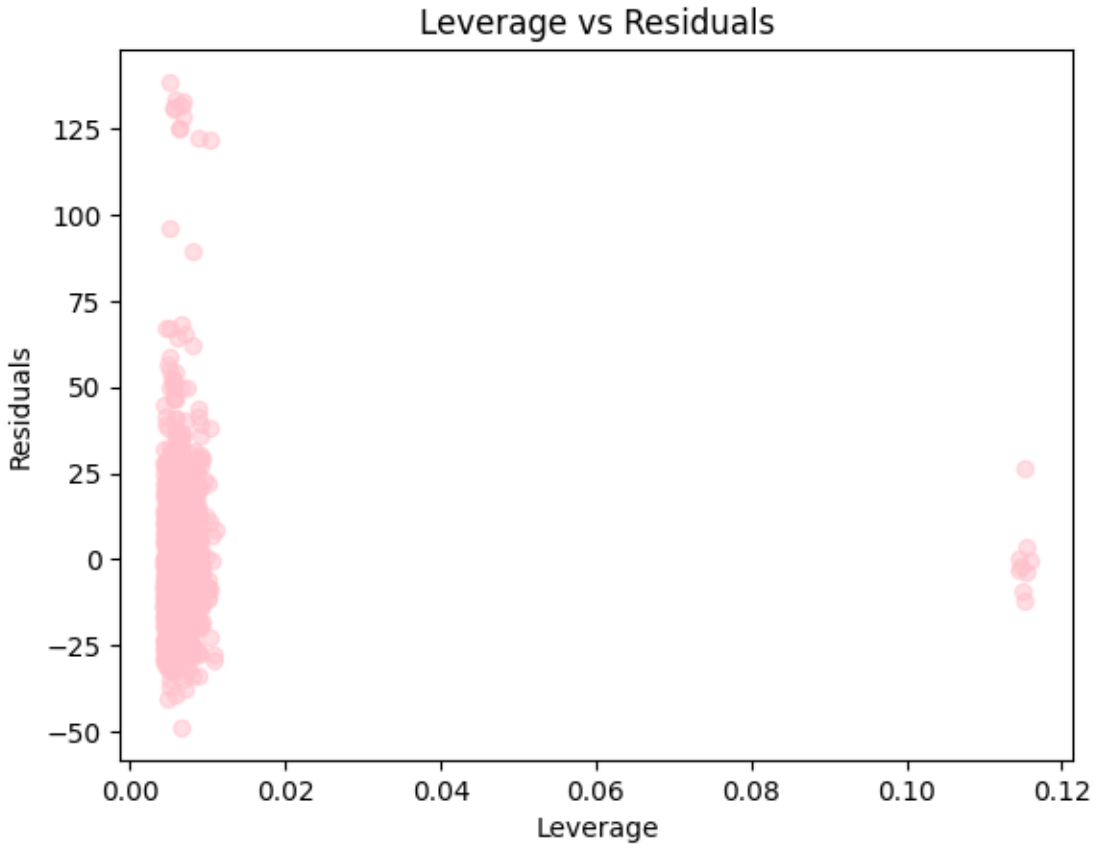
La QQ Plot se desvía mucho de la recta sobre todo en las colas. Notamos además la posible presencia de outliers. Vamos a removerlos y además aplicar una transformación.

la gráfia de residuales contra observaciones nos deja ver la posible presencia de algunos outliers.

## 2.6 Análisis de Influencia

### 2.6.1 Detección de Outliers

[10]:
```
# Leverage Analysis
influence = results.get_influence()
leverage = influence.hat_matrix_diag
plt.scatter(leverage, residuals, alpha=0.5, color=color)
plt.title('Leverage vs Residuals')
plt.xlabel('Leverage')
plt.ylabel('Residuals')
plt.show()
```




Notemos los puntos del lado derecho (leverage alto, residuos entre -20 y 30) son posiblemente un

grupo que esta influenciando en y. Vamos a analizar la distancia para confirmar.

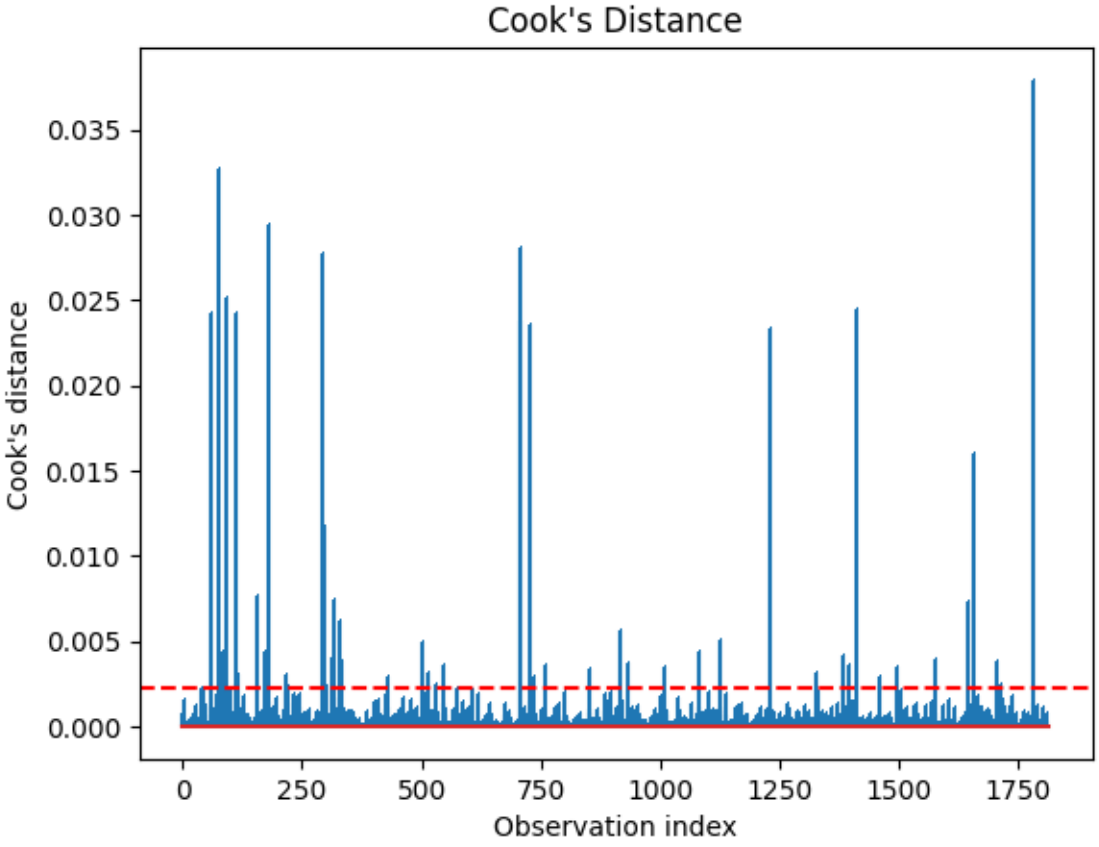
### 2.6.2 Distancia de Cook

$$D_i = \frac{e_i^2}{p \cdot \text{MSE}} \cdot \frac{h_{ii}}{1 - h_{ii}}$$

donde: - $e_i$ es el residual para la observación $i$ - $p$ es el número de parámetros en el modelo - MSE es el error cuadrático medio (varianza residual) - $h_{ii}$ es el valor de leverage para la observación $i$

Umbrales prácticos comunes incluyen $4/n$ o valores cercanos a 1 para señalar posibles influencias. Aquí utilizamos $4/n$ como referencia.

[11]:
```
cooks_d = influence.cooks_distance[0]

plt.stem(np.arange(len(cooks_d)), cooks_d, markerfmt=",")
plt.axhline(4/len(cooks_d), color='red', linestyle='--')
plt.xlabel("Observation index")
plt.ylabel("Cook's distance")
plt.title("Cook's Distance")
plt.show()
```



Se obtienen varios valores influyentes en el resultado. Que son aquellos que sobrepasan el limite establecido (linea roja). Vamos a remover esos datos y antes de volver a ajustar el modelo, aplicaremos una transformación.

### 2.6.3 Remover Outliers

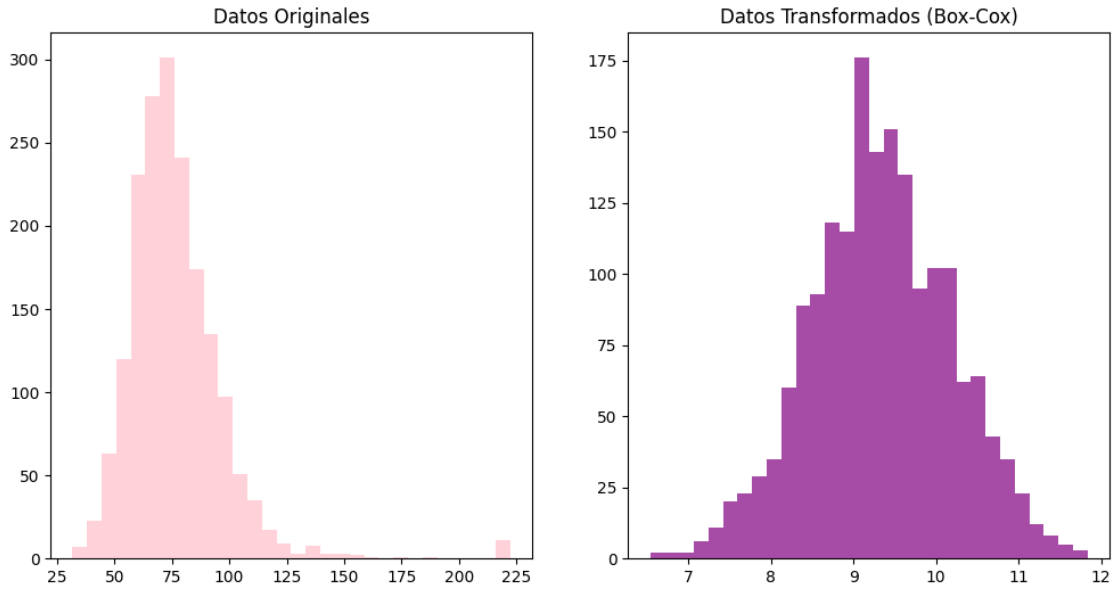
[12]:
```
outlier_threshold = 4 / n
non_outlier_mask = cooks_d < outlier_threshold
df_no_outliers = df[non_outlier_mask]
```

## 2.7 Transformación Box-Cox

[13]:
```
from scipy import stats

transformed_data, lambda_value = stats.boxcox(df_no_outliers['peso'])
df_no_outliers['peso_boxcox'] = transformed_data
print(f"Peso escala original -  Min: {df['peso'].min()}, Max: {df['peso'].
 ↪max()}")
print(f"Lambda value for Box-Cox transformation: {lambda_value}")
```

```
Peso escala original -  Min: 31.7, Max: 222.22
Lambda value for Box-Cox transformation: 0.3234576108677771
```

[14]:
```
plt.figure(figsize=(12, 6))

plt.subplot(1, 2, 1)
plt.hist(y, bins=30, color=color, alpha=0.7)
plt.title('Datos Originales')

color = 'purple'
plt.subplot(1, 2, 2)
plt.hist(transformed_data, bins=30, color='purple', alpha=0.7)
plt.title('Datos Transformados (Box-Cox)')

plt.show()
```




Mismo procedimiento que el inciso anterior, mismo buen resultado.

### 2.8 Segundo ajuste de modelo sin outliers y con T. Box-Cox

[15]:
```
import statsmodels.formula.api as smf
from scipy.special import inv_boxcox

model_formula_bc = smf.ols("peso_boxcox ~ C(sexo) + C(riesgo_diabetes_cat) +␣
↪C(estrato) + C(edad_4cat) + C(estatura_4cat)", data=df_no_outliers)
results_bc = model_formula_bc.fit()

y_no_outliers = df_no_outliers["peso"].values
y_predicted_bc_t = results_bc.predict(df_no_outliers)

# Convertir a escala de peso original
y_pred_bc_original_scale = inv_boxcox(y_predicted_bc_t, lambda_value)

# Print the summary of the model and the first few predictions
print(results_bc.summary())
print("\nFirst 5 predictions on the original scale:")
print(y_pred_bc_original_scale[:5])
```

OLS Regression Results

| | | | |
|---|---|---|---|
| Dep. Variable: | peso_boxcox | R-squared: | 0.275 |
| Model: | OLS | Adj. R-squared: | 0.270 |
| Method: | Least Squares | F-statistic: | 60.33 |
| Date: | Wed, 04 Mar 2026 | Prob (F-statistic): | 6.49e-114 |

```
Time:                        20:10:57   Log-Likelihood:                -1937.1
No. Observations:                1764   AIC:                             3898.
Df Residuals:                    1752   BIC:                             3964.
Df Model:                          11
Covariance Type:            nonrobust
==============================================================================
                                 coef    std err          t      P>|t|
[0.025      0.975]
------------------------------------------------------------------------------
Intercept                      8.4992      0.051    165.872      0.000
8.399       8.600
C(sexo)[T.1]                  -0.1494      0.049     -3.045      0.002
-0.246      -0.053
C(riesgo_diabetes_cat)[T.1]    0.0732      0.298      0.245      0.806
-0.512       0.659
C(riesgo_diabetes_cat)[T.2]    0.3045      0.041      7.462      0.000
0.224       0.385
C(estrato)[T.2.0]              0.1021      0.045      2.281      0.023
0.014       0.190
C(estrato)[T.3.0]              0.0638      0.043      1.488      0.137
-0.020       0.148
C(edad_4cat)[T.2]              0.2179      0.049      4.466      0.000
0.122       0.314
C(edad_4cat)[T.3]              0.1937      0.048      4.017      0.000
0.099       0.288
C(edad_4cat)[T.4]              0.1388      0.052      2.671      0.008
0.037       0.241
C(estatura_4cat)[T.2]          0.4707      0.050      9.454      0.000
0.373       0.568
C(estatura_4cat)[T.3]          0.8002      0.055     14.606      0.000
0.693       0.908
C(estatura_4cat)[T.4]          1.2522      0.067     18.801      0.000
1.122       1.383
==============================================================================
Omnibus:                        1.215   Durbin-Watson:                   1.972
Prob(Omnibus):                  0.545   Jarque-Bera (JB):                1.224
Skew:                           0.003   Prob(JB):                        0.542
Kurtosis:                       2.871   Cond. No.                         23.7
==============================================================================

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

First 5 predictions on the original scale:
2     92.061960
```

```
3     89.135838
5     90.572497
8     64.061773
10    82.724096
dtype: float64
```

Notamos: - Los coeficientes de estatura se disminuyeron para la estatura, pero esto se debe a que ahora están en otra escala. - $R^2$ de 23.1% a 27.2% - Estadístico F: De 41.12 a 59.4 - Ahora solo 3 coeficientes incluyen al cero en sus intervalos de confianza

### 2.9 Segundo análisis de residuales

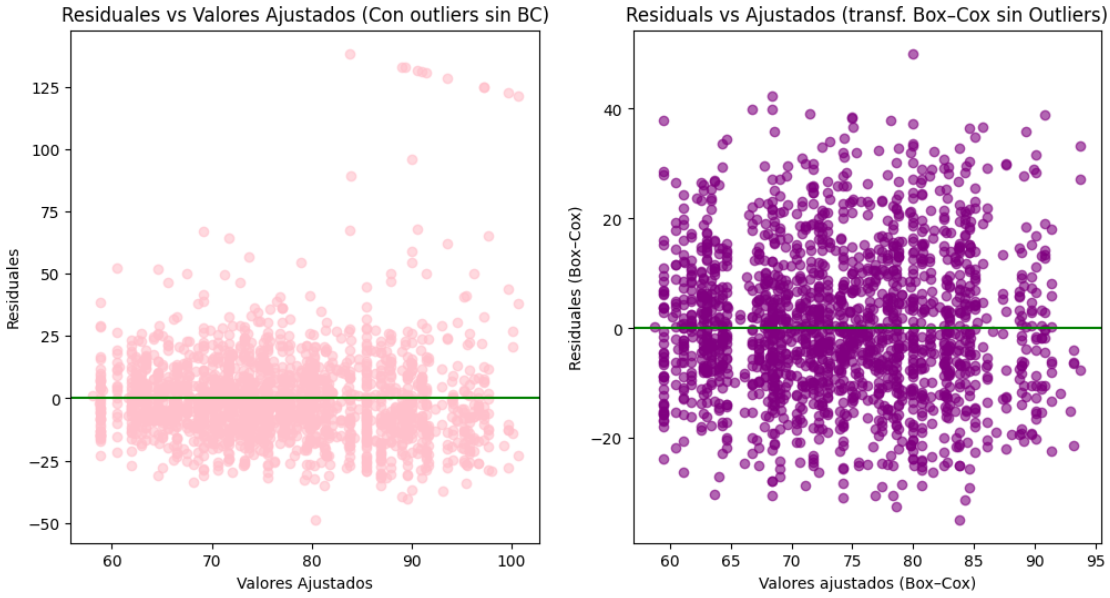
[16]:
```python
residuals_bc = y_no_outliers - y_pred_bc_original_scale

# Varianza de los residuales
residual_sum_squares = np.sum(residuals_bc**2) # RSS
n = len(y_no_outliers)
p = X_sel_no_outliers.shape[1]
sigma_squared = residual_sum_squares / (n - p - 1)
sigma = np.sqrt(sigma_squared)

print(f"Error Estándar Residual: {sigma:.4f}")
print(f"Varianza residual: {sigma_squared:.4f}")
print("\n")
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[16], line 6
      4 residual_sum_squares = np.sum(residuals_bc**2) # RSS
      5 n = len(y_no_outliers)
----> 6 p = X_sel_no_outliers.shape[1]
      7 sigma_squared = residual_sum_squares / (n - p - 1)
      8 sigma = np.sqrt(sigma_squared)

NameError: name 'X_sel_no_outliers' is not defined
```

Vemos una ligera mejora en el error estandar valor anterior 18.62 y casa una disminución del 50% la varianza.

[ ]:
```python
plt.figure(figsize=(12, 6))


# Residuals vs Fitted
plt.subplot(1, 2, 1)
plt.scatter(y_pred, residuals, alpha=0.6, color='pink')
plt.axhline(y=0, color='g')
plt.xlabel('Valores Ajustados')
```

```
plt.ylabel('Residuales')
plt.title('Residuales vs Valores Ajustados (Con outliers sin BC)')



plt.subplot(1, 2, 2)
plt.scatter(y_pred_bc_original_scale, residuals_bc, alpha=0.6, color=color)
plt.axhline(0, color='g')
plt.xlabel("Valores ajustados (Box-Cox)")
plt.ylabel("Residuales (Box-Cox)")
plt.title("Residuals vs Ajustados (transf. Box-Cox sin Outliers)")
plt.show()
```

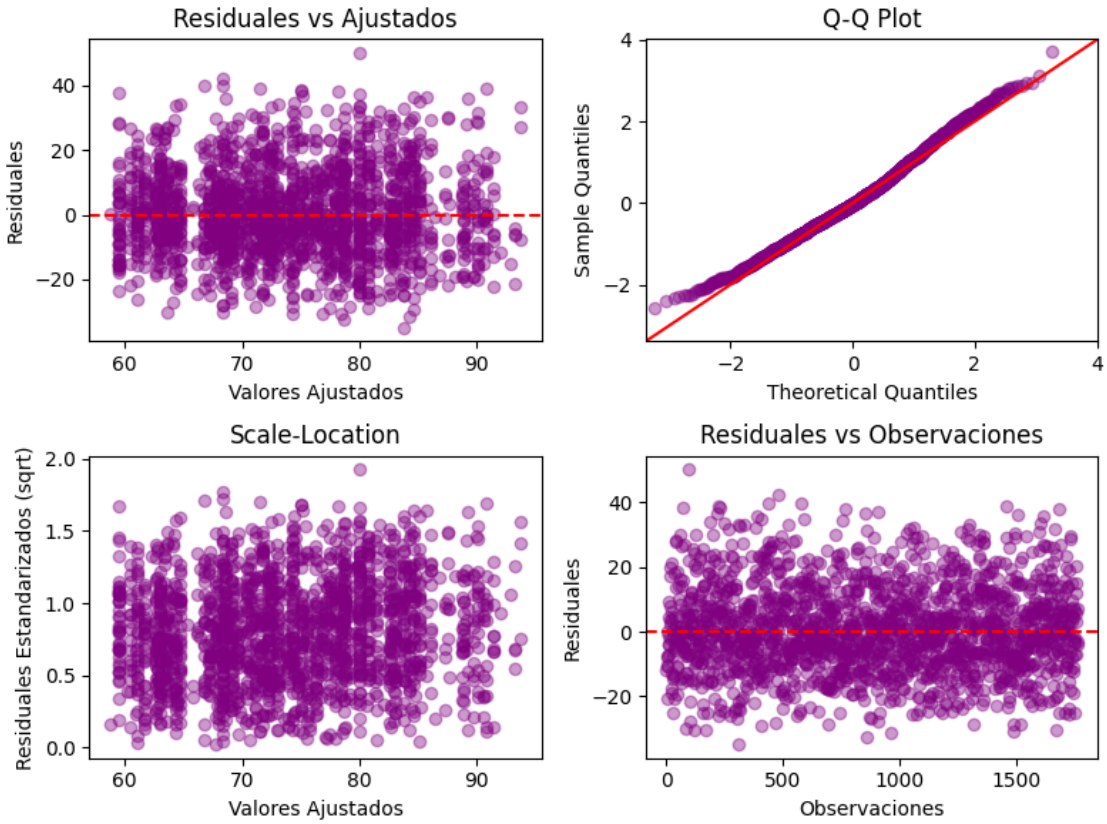



Notamos valores más dispersos fuera de los patrones categóricos

### 2.9.1 Gráficas de Diagnóstico de Residuales

[ ]:
```
from statsmodels.graphics.gofplots import ProbPlot

fig, axes = plt.subplots(2, 2, figsize=(8, 6))

# Residuales vs Ajustados
axes[0, 0].scatter(y_pred_bc_original_scale, residuals_bc, alpha=0.4,␣
 ↪color=color)
axes[0, 0].axhline(y=0, color='r', linestyle='--')
axes[0, 0].set_xlabel('Valores Ajustados')
axes[0, 0].set_ylabel('Residuales')
```

```
axes[0, 0].set_title('Residuales vs Ajustados')

standardized_residuals_bc = residuals_bc / sigma

# Q-Q plot
ProbPlot(standardized_residuals_bc).qqplot(ax=axes[0, 1], line='45', alpha=0.4,␣
 ↪markerfacecolor=color, markeredgecolor=color)
axes[0, 1].set_title('Q-Q Plot')

# Residuales Estandarizados vs Ajustados
axes[1, 0].scatter(y_pred_bc_original_scale, np.sqrt(np.
 ↪abs(standardized_residuals_bc)), alpha=0.4, color=color)
axes[1, 0].set_xlabel('Valores Ajustados')
axes[1, 0].set_ylabel('Residuales Estandarizados (sqrt)')
axes[1, 0].set_title('Scale-Location')

# Residuales vs Leverage
axes[1, 1].scatter(range(n), residuals_bc, alpha=0.4, color=color)
axes[1, 1].axhline(y=0, color='r', linestyle='--')
axes[1, 1].set_xlabel('Observaciones')
axes[1, 1].set_ylabel('Residuales')
axes[1, 1].set_title('Residuales vs Observaciones')

plt.tight_layout()
```




- Se notan aún los patrones de las variables categóricas en las 2 gráficas de la izquierda
- Como es de esperarse, ahora los datos se acercan mucho a la linea normal.
- En la gráfica de residuales vs observacionesya no hay patrones lo cuál no da fuerte evidencia de la homocedasticidad.

## 2.10 Multicolinealidad - VIF

El Factor de Inflación de la Varianza (VIF) se define como:

$$\text{VIF}_i = \frac{1}{1 - R_i^2}$$

donde $R_i^2$ es el coeficiente de determinación al regresar ese predictor contra los demás (no contra la respuesta).

[ ]:
```
from statsmodels.stats.outliers_influence import variance_inflation_factor

# Select predictors for VIF calculation
predictors = ["sexo", "riesgo_diabetes_cat", "estrato", "edad_4cat",␣
  ↪"estatura_4cat"]
X_vif = pd.get_dummies(df[predictors], drop_first=True)
```

```
# Compute VIF for each predictor
vif_data = pd.DataFrame()
vif_data["Variable"] = X_vif.columns
vif_data["VIF"] = [variance_inflation_factor(X_vif.values, i) for i in␣
 ↪range(X_vif.shape[1])]

print("Variance Inflation Factor (VIF):")
print(vif_data)
```

```
Variance Inflation Factor (VIF):
              Variable       VIF
0                 sexo  2.771090
1  riesgo_diabetes_cat  1.494356
2              estrato  5.904314
3            edad_4cat  4.166611
4        estatura_4cat  8.124789
```

Notamos un alto valor en estatura, intuitivamente puede estar correlacionada con sexo y edad.

### 2.11 Tercer ajuste de modelo

Esta vez Hagamos la prueba, quitando `sexo` y `edad_4cat` ambas, y una por una.

[ ]:
```
import statsmodels.formula.api as smf
from scipy.special import inv_boxcox

model_formula_3 = smf.ols("peso_boxcox ~ C(riesgo_diabetes_cat) + C(estrato) +␣
 ↪+ C(edad_4cat) + C(estatura_4cat)", data=df_no_outliers)
results_bc = model_formula_3.fit()

y_3 = df_no_outliers["peso"].values
y_predicted_3 = results_bc.predict(df_no_outliers)

# Convertir a escala de peso original
y_pred_3_original_scale = inv_boxcox(y_predicted_3, lambda_value)

# Print the summary of the model and the first few predictions
print(results_bc.summary())
print("\nFirst 5 predictions on the original scale:")
print(y_pred_3_original_scale[:5])
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:            peso_boxcox   R-squared:                       0.271
Model:                            OLS   Adj. R-squared:                  0.267
Method:                 Least Squares   F-statistic:                     65.13
Date:                Thu, 05 Feb 2026   Prob (F-statistic):          7.49e-113
Time:                        05:34:48   Log-Likelihood:                -1941.7
```

```
No. Observations:                1764   AIC:                             3905.
Df Residuals:                    1753   BIC:                             3966.
Df Model:                          10
Covariance Type:            nonrobust
==============================================================================
==============
                                 coef    std err          t      P>|t|
[0.025      0.975]
------------------------------------------------------------------------------
--------------
Intercept                      8.4984      0.051    165.469      0.000
8.398       8.599
C(riesgo_diabetes_cat)[T.1]    0.0864      0.299      0.289      0.773
-0.500       0.673
C(riesgo_diabetes_cat)[T.2]    0.3045      0.041      7.444      0.000
0.224       0.385
C(estrato)[T.2.0]              0.1092      0.045      2.436      0.015
0.021       0.197
C(estrato)[T.3.0]              0.0788      0.043      1.846      0.065
-0.005       0.162
C(edad_4cat)[T.2]              0.2148      0.049      4.394      0.000
0.119       0.311
C(edad_4cat)[T.3]              0.1797      0.048      3.735      0.000
0.085       0.274
C(edad_4cat)[T.4]              0.1214      0.052      2.345      0.019
0.020       0.223
C(estatura_4cat)[T.2]          0.4477      0.049      9.076      0.000
0.351       0.544
C(estatura_4cat)[T.3]          0.7286      0.050     14.688      0.000
0.631       0.826
C(estatura_4cat)[T.4]          1.1200      0.051     22.118      0.000
1.021       1.219
==============================================================================
Omnibus:                        1.870   Durbin-Watson:                   1.977
Prob(Omnibus):                  0.393   Jarque-Bera (JB):                1.800
Skew:                           0.021   Prob(JB):                        0.407
Kurtosis:                       2.849   Cond. No.                         22.6
==============================================================================

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

First 5 predictions on the original scale:
2     89.207563
3     86.699075
5     91.173275
8     64.247709
```

```
10    83.004546
dtype: float64
```

[ ]:
```python
residuals_bc = y_no_outliers - y_pred_3_original_scale

# Varianza de los residuales
residual_sum_squares = np.sum(residuals_bc**2) # RSS
n = len(y_no_outliers)
p = df_no_outliers.shape[1]
sigma_squared = residual_sum_squares / (n - p - 1)
sigma = np.sqrt(sigma_squared)

print(f"Error Estándar Residual: {sigma:.4f}")
print(f"Varianza residual: {sigma_squared:.4f}")
print("\n")
```

```
Residual standard error: 13.6446
Varianza residual: 186.1743
```

Cuando quitamos ambas, el estadístico $R^2$ y el F disminuyeron, el error estándar residual y la varianza quedaron practicamente igual. Al quitar solo el sexo, los estadísticos aumentaron al igual que el error residual y su varianza. Así que nos quedaremos con el segundo modelo.

### 2.12 Predicciones

[ ]:
```python
# Definimos unas nuevas variables
new_values = pd.DataFrame()
new_values['sexo'] = [1, 0]  # Hombre, Mujer
new_values['riesgo_diabetes_cat'] = [1, 2]  # Bajo
new_values['estrato'] = [1, 2]  # Estrato 1, Estrato 2
new_values['edad_4cat'] = [2, 3]  # 31-40 y 41-50 años
new_values['estatura_4cat'] = [3, 4]  # 152.5 - 158.9 y 158.91 - 165.65

y_predicted_new = results_bc.predict(new_values)


# Convertir a escala de peso original
new_values['predicted'] = inv_boxcox(y_predicted_new, lambda_value)



print("Predicciones para nuevos valores en escala original:")
print(new_values)
```

```
Predicciones para nuevos valores en escala original:
   sexo  riesgo_diabetes_cat  estrato  edad_4cat  estatura_4cat  ci_95_inf  \
0     1                    1        1          2              3  65.130679
```

```
1     0                   2        2          3              4  90.916649

   ci_95_sup  predicted
0  87.433720  75.727652
1  97.290519  94.067064
```

[ ]: