In [3]:

```
import numpy as np
import pandas as pd
import scipy.stats
import warnings
warnings.filterwarnings("ignore")

import os
os.chdir("/Users/Lenovo/Desktop/EBAC")

data = pd.read_csv("Advertising.csv")
Y = np.array(data["Sales"]).reshape(-1, 1)
X = np.array(data[["TV", "Radio", "Newspaper"]])

# intercepto
X = np.hstack([np.ones((X.shape[0], 1)), X])

# Estimación de betas (MCO)
XT_X = np.matmul(X.T, X)
XT_X_inv = np.linalg.inv(XT_X)
XT_Y = np.matmul(X.T, Y)
betas = np.matmul(XT_X_inv, XT_Y)

# Predicciones y residuos
Y_pred = np.matmul(X, betas)
Resid = Y - Y_pred

RSS = np.matmul(Resid.T, Resid)
TSS = np.matmul(Y.T, Y) - len(Y)*(Y.mean()**2)
R2 = float(1 - RSS/TSS)

n, k = X.shape
s_cuad = RSS / (n - k)
s = float(np.sqrt(s_cuad))

print("Coeficientes beta:")
print(betas)
print("R²:", R2)

# Intervalo de confianza 90%
# para TV=100, Radio=50, Newspaper=70

f = np.array([[1, 100, 50, 70]])
Pron_puntual = float(np.matmul(f, betas))

gl = n - k
Confianza = 0.90
q = (1 - Confianza) / 2
t_critico = abs(scipy.stats.t.ppf(q, df=gl))

Margen_error = t_critico * (s * (float(np.matmul(np.matmul(f, XT_X_inv), f.T))**0.5))
Lim_inferior = Pron_puntual - Margen_error
Lim_superior = Pron_puntual + Margen_error

print("Predicción puntual:", Pron_puntual)
print("Intervalo de confianza 90%: (", Lim_inferior, ",", Lim_superior, ")")


# Validación de supuestos
# Normalidad de residuos (Jarque-Bera)
skew = float(scipy.stats.skew(Resid, bias=True))
kurtosis = float(scipy.stats.kurtosis(Resid, fisher=False))
Jarque_bera = (n/6) * (skew**2 + (kurtosis-3)**2 / 4)

print("Asimetría:", skew)
print("Curtosis:", kurtosis)
print("Estadístico Jarque-Bera:", Jarque_bera)

# Autocorrelación (Durbin-Watson)
from statsmodels.formula.api import ols
from statsmodels.stats.stattools import durbin_watson

df = pd.DataFrame(data={"Y": Y.flatten(), "TV": data["TV"], "Radio": data["Radio"], "Newspaper": data["Newspape
model = ols("Y ~ TV + Radio + Newspaper", data=df).fit()

dw = durbin_watson(model.resid)
print("Durbin-Watson:", dw)

# Heterocedasticidad (Prueba de White)
from statsmodels.stats.diagnostic import het_white
white_test = het_white(model.resid, model.model.exog)

print("Estadístico White:", white_test[0])
```

```
print("Valor p White:", white_test[1])
```

```
Coeficientes beta:
[[4.62512408e+00]
 [5.44457803e-02]
 [1.07001228e-01]
 [3.35657922e-04]]
R²: 0.9025912899684556
Predicción puntual: 15.44325957906403
Intervalo de confianza 90%: ( 14.954628615233734 , 15.931890542894326 )
Asimetría: -0.4310967996702898
Curtosis: 4.604756333677674
Estadístico Jarque-Bera: 27.65517244352062
Durbin-Watson: 2.250551836079356
Estadístico White: 19.95314047160226
Valor p White: 0.018203803931681445
```

Jarque - Bera: Dado que el estadístico de Jarque-Bera es mayor al valor crítico, rechazamos la hipótesis nula y concluimos que los residuales no siguen una distribución normal.

Durbin-Watson: Dado que DW = 2.25, que es mayor que 2, se evidencia una ligera autocorrelación negativa entre los residuales.

Estadistico de White: Dado que el valor p de la prueba de White es menor que 0.05, rechazamos la hipótesis nula y concluimos que existe heterocedasticidad en los residuales (varianzas desiguales).

Multicolinealidad: Todas las variables independientes tienen VIF cercanos a 1, lo que indica que no existe multicolinealidad significativa en el modelo.

In [9]:
```
from sklearn.model_selection import train_test_split

# Variables explicativas
X = data[["TV", "Radio", "Newspaper"]]
y = data["Sales"]

# División 70% entrenamiento, 30% prueba (semilla = 1)
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.3, random_state=1
)
```

In [11]:
```
import statsmodels.api as sm

# intercepto
X_train_const = sm.add_constant(X_train)

# Ajuste del modelo
model = sm.OLS(y_train, X_train_const).fit()
print(model.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  Sales   R-squared:                       0.899
Model:                            OLS   Adj. R-squared:                  0.897
Method:                 Least Squares   F-statistic:                     405.2
Date:                Tue, 09 Sep 2025   Prob (F-statistic):           1.36e-67
Time:                        18:13:17   Log-Likelihood:                -272.35
No. Observations:                 140   AIC:                             552.7
Df Residuals:                     136   BIC:                             564.5
Df Model:                           3
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          4.6614      0.368     12.650      0.000       3.933       5.390
TV             0.0550      0.002     32.747      0.000       0.052       0.058
Radio          0.1025      0.010      9.801      0.000       0.082       0.123
Newspaper     -0.0015      0.007     -0.203      0.839      -0.016       0.013
==============================================================================
Omnibus:                       15.370   Durbin-Watson:                   2.041
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               25.876
Skew:                          -0.526   Prob(JB):                     2.41e-06
Kurtosis:                       4.824   Cond. No.                         432.
==============================================================================

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

In [13]:
```
X_train_sig = X_train[["TV", "Radio"]]
X_train_sig = sm.add_constant(X_train_sig)

model_sig = sm.OLS(y_train, X_train_sig).fit()
print(model_sig.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  Sales   R-squared:                       0.899
Model:                            OLS   Adj. R-squared:                  0.898
Method:                 Least Squares   F-statistic:                     612.0
Date:                Tue, 09 Sep 2025   Prob (F-statistic):           4.95e-69
Time:                        18:13:32   Log-Likelihood:                -272.37
No. Observations:                 140   AIC:                             550.7
Df Residuals:                     137   BIC:                             559.6
Df Model:                           2
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          4.6390      0.350     13.240      0.000       3.946       5.332
TV             0.0550      0.002     32.903      0.000       0.052       0.058
Radio          0.1016      0.010     10.677      0.000       0.083       0.120
==============================================================================
Omnibus:                       14.841   Durbin-Watson:                   2.038
Prob(Omnibus):                  0.001   Jarque-Bera (JB):               24.663
Skew:                          -0.512   Prob(JB):                     4.41e-06
Kurtosis:                       4.783   Cond. No.                         407.
==============================================================================

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

In [15]:
```python
from sklearn.metrics import r2_score

# Preparar X_test con solo las variables significativas
X_test_sig = sm.add_constant(X_test[["TV", "Radio"]])

# Predicciones
y_pred = model_sig.predict(X_test_sig)

# R² en prueba
r2_test = r2_score(y_test, y_pred)
print("R² en conjunto de prueba:", r2_test)
```

```
R² en conjunto de prueba: 0.9073562242286408
```

In [19]:
```python
import scipy.stats
from statsmodels.formula.api import ols
from statsmodels.stats.stattools import durbin_watson
from statsmodels.stats.diagnostic import het_white
import pandas as pd

data = pd.read_csv("Advertising.csv")
df = pd.DataFrame({
    "Y": data["Sales"],
    "TV": data["TV"],
    "Radio": data["Radio"],
    "Newspaper": data["Newspaper"]
})

# Ajustar modelo de regresión
model = ols("Y ~ TV + Radio + Newspaper", data=df).fit()
resid = model.resid

# Sesgo
skew_lib = float(scipy.stats.skew(resid, bias=True))

# Curtosis
kurtosis_lib = float(scipy.stats.kurtosis(resid, fisher=False))

# Durbin-Watson
dw_lib = durbin_watson(resid)

# Heterocedasticidad (White)
white_test = het_white(resid, model.model.exog)
white_stat_lib = white_test[0]
white_pvalue_lib = white_test[1]

print("Sesgo (skew) librería:", skew_lib)
print("Curtosis librería:", kurtosis_lib)
print("Durbin-Watson librería:", dw_lib)
print("White test estadístico:", white_stat_lib)
print("White test valor p:", white_pvalue_lib)
```

```
Sesgo (skew) librería: -0.4310967996702909
Curtosis librería: 4.604756333677675
Durbin-Watson librería: 2.250551836079356
White test estadístico: 19.95314047160226
White test valor p: 0.018203803931681445
```

Conclusion

Despues de hacer la comparativa de los calculos de matrices manuales VS las librerias, podemos concluir que los calculos manuales fueron exitosos, ya que llegamos a los mismos valores.

In [22]:
```
from statsmodels.stats.outliers_influence import variance_inflation_factor
import pandas as pd

X = data[["TV", "Radio", "Newspaper"]]
X = sm.add_constant(X)

vif_data = pd.DataFrame()
vif_data["Variable"] = X.columns
vif_data["VIF"] = [variance_inflation_factor(X.values, i) for i in range(X.shape[1])]

print(vif_data)
```

```
    Variable       VIF
0      const  6.848900
1         TV  1.004611
2      Radio  1.144952
3  Newspaper  1.145187
```

In [ ]:
```

```