## Setup

```
import pandas as pd
import numpy as np

import matplotlib.pyplot as plt
from matplotlib import style
import seaborn as sns

# Configuración matplotlib
plt.rcParams['image.cmap'] = "bwr"
#plt.rcParams['figure.dpi'] = "100"
plt.rcParams['savefig.bbox'] = "tight"
style.use('ggplot') or plt.style.use('ggplot')

# Configuración warnings
import warnings
warnings.filterwarnings('ignore')
```

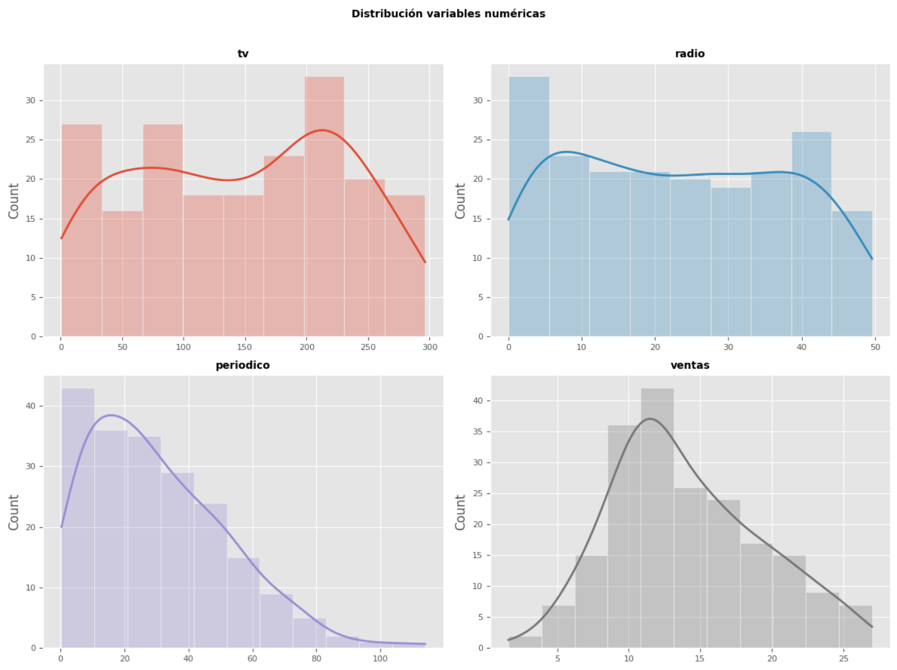
## Datos

```
tv = [230.1, 44.5, 17.2, 151.5, 180.8, 8.7, 57.5, 120.2, 8.6, 199.8, 66.1, 214.7, 23.8, 97.5, 204.1, 195.4, 67.8, 281.4, 69.2, 147.3, 218.4,
228.3, 62.3, 262.9, 142.9, 240.1, 248.8, 70.6, 292.9, 112.9, 97.2, 265.6, 95.7, 290.7, 266.9, 74.7, 43.1, 228.0, 202.5, 177.0, 293.6,
206.9, 25.1, 175.1, 89.7, 239.9, 227.2, 66.9, 199.8, 100.4, 216.4, 182.6, 262.7, 198.9, 7.3, 136.2, 210.8, 210.7, 53.5, 261.3, 239.3, 102.7,
69.0, 31.5, 139.3, 237.4, 216.8, 199.1, 109.8, 26.8, 129.4, 213.4, 16.9, 27.5, 120.5, 5.4, 116.0, 76.4, 239.8, 75.3, 68.4, 213.5, 193.2, 76.3
88.3, 109.8, 134.3, 28.6, 217.7, 250.9, 107.4, 163.3, 197.6, 184.9, 289.7, 135.2, 222.4, 296.4, 280.2, 187.9, 238.2, 137.9, 25.0, 90.4, 13.1,
225.8, 241.7, 175.7, 209.6, 78.2, 75.1, 139.2, 76.4, 125.7, 19.4, 141.3, 18.8, 224.0, 123.1, 229.5, 87.2, 7.8, 80.2, 220.3, 59.6, 0.7, 265.2,
48.3, 25.6, 273.7, 43.0, 184.9, 73.4, 193.7, 220.5, 104.6, 96.2, 140.3, 240.1, 243.2, 38.0, 44.7, 280.7, 121.0, 197.6, 171.3, 187.8, 4.1, 93.
131.7, 172.5, 85.7, 188.4, 163.5, 117.2, 234.5, 17.9, 206.8, 215.4, 284.3, 50.0, 164.5, 19.6, 168.4, 222.4, 276.9, 248.4, 170.2, 276.7, 165.6
287.6, 253.8, 205.0, 139.5, 191.1, 286.0, 18.7, 39.5, 75.5, 17.2, 166.8, 149.7, 38.2, 94.2, 177.0, 283.6, 232.1]

radio = [37.8, 39.3, 45.9, 41.3, 10.8, 48.9, 32.8, 19.6, 2.1, 2.6, 5.8, 24.0, 35.1, 7.6, 32.9, 47.7, 36.6, 39.6, 20.5, 23.9, 27.7, 5.1, 15.9,
29.3, 16.7, 27.1, 16.0, 28.3, 17.4, 1.5, 20.0, 1.4, 4.1, 43.8, 49.4, 26.7, 37.7, 22.3, 33.4, 27.7, 8.4, 25.7, 22.5, 9.9, 41.5, 15.8, 11.7, 3.
41.7, 46.2, 28.8, 49.4, 28.1, 19.2, 49.6, 29.5, 2.0, 42.7, 15.5, 29.6, 42.8, 9.3, 24.6, 14.5, 27.5, 43.9, 30.6, 14.3, 33.0, 5.7, 24.6, 43.7,
29.9, 7.7, 26.7, 4.1, 20.3, 44.5, 43.0, 18.4, 27.5, 40.6, 25.5, 47.8, 4.9, 1.5, 33.5, 36.5, 14.0, 31.6, 3.5, 21.0, 42.3, 41.7, 4.3, 36.3, 10.
34.3, 46.4, 11.0, 0.3, 0.4, 26.9, 8.2, 38.0, 15.4, 20.6, 46.8, 35.0, 14.3, 0.8, 36.9, 16.0, 26.8, 21.7, 2.4, 34.6, 32.3, 11.8, 38.9, 0.0, 49.
39.6, 2.9, 27.2, 33.5, 38.6, 47.0, 39.0, 28.9, 25.9, 43.9, 17.0, 35.4, 33.2, 5.7, 14.8, 1.9, 7.3, 49.0, 40.3, 25.8, 13.9, 8.4, 23.3, 39.7, 21
1.3, 36.9, 18.4, 18.1, 35.8, 18.1, 36.8, 14.7, 3.4, 37.6, 5.2, 23.6, 10.6, 11.6, 20.9, 20.1, 7.1, 3.4, 48.9, 30.2, 7.8, 2.3, 10.0, 2.6, 5.4,
2.1, 28.7, 13.9, 12.1, 41.1, 10.8, 4.1, 42.0, 35.6, 3.7, 4.9, 9.3, 42.0, 8.6]

periodico = [69.2, 45.1, 69.3, 58.5, 58.4, 75.0, 23.5, 11.6, 1.0, 21.2, 24.2, 4.0, 65.9, 7.2, 46.0, 52.9, 114.0, 55.8, 18.3, 19.1, 53.4, 23.5
18.3, 19.5, 12.6, 22.9, 22.9, 40.8, 43.2, 38.6, 30.0, 0.3, 7.4, 8.5, 5.0, 45.7, 35.1, 32.0, 31.6, 38.7, 1.8, 26.4, 43.3, 31.5, 35.7, 18.5, 49
36.8, 34.6, 3.6, 39.6, 58.7, 15.9, 60.0, 41.4, 16.6, 37.7, 9.3, 21.4, 54.7, 27.3, 8.4, 28.9, 0.9, 2.2, 10.2, 11.0, 27.2, 38.7, 31.7, 19.3, 31
89.4, 20.7, 14.2, 9.4, 23.1, 22.3, 36.9, 32.5, 35.6, 33.8, 65.7, 16.0, 63.2, 73.4, 51.4, 9.3, 33.0, 59.0, 72.3, 10.9, 52.9, 5.9, 22.0, 51.2,
100.9, 21.4, 17.9, 5.3, 59.0, 29.7, 23.2, 25.6, 5.5, 56.5, 23.2, 2.4, 10.7, 34.5, 52.7, 25.6, 14.8, 79.2, 22.3, 46.2, 50.4, 15.6, 12.4, 74.2,
9.2, 3.2, 43.1, 8.7, 43.0, 2.1, 45.1, 65.6, 8.5, 9.3, 59.7, 20.5, 1.7, 12.9, 75.6, 37.9, 34.4, 38.9, 9.0, 8.7, 44.3, 11.9, 20.6, 37.0, 48.7,
9.5, 5.7, 50.5, 24.3, 45.2, 34.6, 30.7, 49.3, 25.6, 7.4, 5.4, 84.8, 21.6, 19.4, 57.6, 6.4, 18.4, 47.4, 17.0, 12.8, 13.1, 41.8, 20.3, 35.2, 23
27.4, 29.7, 71.8, 30.0, 19.6, 26.6, 18.2, 3.7, 23.4, 5.8, 6.0, 31.6, 3.6, 6.0, 13.8, 8.1, 6.4, 66.2, 8.7]

ventas = [22.1, 10.4, 9.3, 18.5, 12.9, 7.2, 11.8, 13.2, 4.8, 10.6, 8.6, 17.4, 9.2, 9.7,19.0, 22.4, 12.5, 24.4, 11.3, 14.6, 18.0, 12.5, 5.6, 1
9.7, 12.0, 15.0, 15.9, 18.9, 10.5, 21.4, 11.9, 9.6, 17.4, 9.5, 12.8, 25.4, 14.7, 10.1, 21.5, 16.6, 17.1, 20.7, 12.9, 8.5, 14.9, 10.6, 23.2, 1
22.6, 21.2, 20.2, 23.7, 5.5, 13.2, 23.8, 18.4, 8.1, 24.2, 15.7, 14.0, 18.0, 9.3, 9.5, 13.4, 18.9, 22.3, 18.3, 12.4, 8.8, 11.0, 17.0, 8.7, 6.9
11.8, 12.3, 11.3, 13.6, 21.7, 15.2, 12.0, 16.0, 12.9, 16.7, 11.2, 7.3, 19.4, 22.2, 11.5, 16.9, 11.7, 15.5, 25.4, 17.2, 11.7, 23.8, 14.8, 14.7
5.3, 19.8, 13.4, 21.8, 14.1, 15.9, 14.6, 12.6, 12.2, 9.4, 15.9, 6.6, 15.5, 7.0, 11.6, 15.2, 19.7, 10.6, 6.6, 8.8, 24.7, 9.7, 1.6, 12.7, 5.7,
20.8, 9.6, 20.7, 10.9, 19.2, 20.1, 10.4, 11.4, 10.3, 13.2, 25.4, 10.9, 10.1, 16.1, 11.6, 16.6, 19.0, 15.6, 3.2, 15.3, 10.1, 7.3, 12.9, 14.4,
11.9, 8.0, 12.2, 17.1, 15.0, 8.4, 14.5, 7.6, 11.7, 11.5, 27.0, 20.2, 11.7, 11.8, 12.6, 10.5, 12.2, 8.7, 26.2, 17.6, 22.6, 10.3, 17.3, 15.9, 6
19.6, 17.3, 7.6, 9.7, 12.8, 25.5, 13.4]

datos = pd.DataFrame({'tv':tv, 'radio': radio, 'periodico':periodico,
'ventas': ventas})

display(datos.head(3))
```

```
      tv  radio  periodico  ventas
0  230.1   37.8       69.2    22.1
1   44.5   39.3       45.1    10.4
2   17.2   45.9       69.3     9.3
```

## Analisis Exploratorio

```
fig, axes = plt.subplots(nrows=2, ncols=2, figsize=(12,9))
axes = axes.flat
columnas_numeric = datos.select_dtypes(include=['float64', 'int']).columns

for i, colum in enumerate(columnas_numeric):
    sns.histplot(
        data    = datos,
        x       = colum,
        stat    = "count",
        kde     = True,
        color   = (list(plt.rcParams['axes.prop_cycle'])*2)[i]["color"],
        line_kws= {'linewidth': 2},
        alpha   = 0.3,
        ax      = axes[i]
    )
    axes[i].set_title(colum, fontsize = 10, fontweight = "bold")
    axes[i].tick_params(labelsize = 8)
    axes[i].set_xlabel("")


fig.tight_layout()
plt.subplots_adjust(top = 0.9)
fig.suptitle('Distribución variables numéricas', fontsize = 10, fontweight = "bold");
```



## Ajuste del modelo 70/30

```
from scipy.stats import pearsonr
from sklearn.model_selection import train_test_split
from sklearn.metrics import r2_score
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import mean_squared_error
import statsmodels.api as sm
import statsmodels.formula.api as smf
from statsmodels.stats.anova import anova_lm #*
from scipy import stats
```

```
X = datos[['tv', 'radio', 'periodico']]
y = datos['ventas']

X_train, X_test, y_train, y_test = train_test_split(
                                        X,
                                        y.values.reshape(-1,1),
                                        train_size   = 0.7,
                                        random_state = 1234,
                                        shuffle      = True
                                    )

X_train = sm.add_constant(X_train, prepend=True)
modelo = sm.OLS(endog=y_train, exog=X_train,)
modelo = modelo.fit()
print(modelo.summary())
```

```
                            OLS Regression Results                            
==============================================================================
Dep. Variable:                      y   R-squared:                       0.895
Model:                            OLS   Adj. R-squared:                  0.892
Method:                 Least Squares   F-statistic:                     384.7
Date:                Thu, 27 Apr 2023   Prob (F-statistic):           3.19e-66
Time:                        21:31:56   Log-Likelihood:                -271.78
No. Observations:                 140   AIC:                             551.6
Df Residuals:                     136   BIC:                             563.3
Df Model:                           3                                         
Covariance Type:            nonrobust                                         
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          2.9389      0.389      7.557      0.000       2.170       3.708
tv             0.0461      0.002     26.799      0.000       0.043       0.049
radio          0.1854      0.010     18.304      0.000       0.165       0.205
periodico      0.0016      0.007      0.214      0.831      -0.013       0.016
==============================================================================
Omnibus:                       50.715   Durbin-Watson:                   2.130
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              141.922
Skew:                          -1.416   Prob(JB):                     1.52e-31
Kurtosis:                       7.039   Cond. No.                         456.
==============================================================================

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

```
print("Recta matematica de la regresion \n\n ventas = 0.0461tv + 0.1854radio + 0.0016periodico + 2.9389")
```

```
Recta matematica de la regresion

 ventas = 0.0461tv + 0.1854radio + 0.0016periodico + 2.9389
```

## Eliminando la variable "Periodico"

```
A = datos[['tv', 'radio']]
B = datos['ventas']

A_train, A_test, B_train, B_test = train_test_split(
                                        A,
                                        B.values.reshape(-1,1),
                                        train_size   = 0.7,
                                        random_state = 1234,
                                        shuffle      = True
                                    )

A_train = sm.add_constant(A_train, prepend=True)
modeloB = sm.OLS(endog=B_train, exog=A_train,)
modeloB = modeloB.fit()
print(modeloB.summary())
```

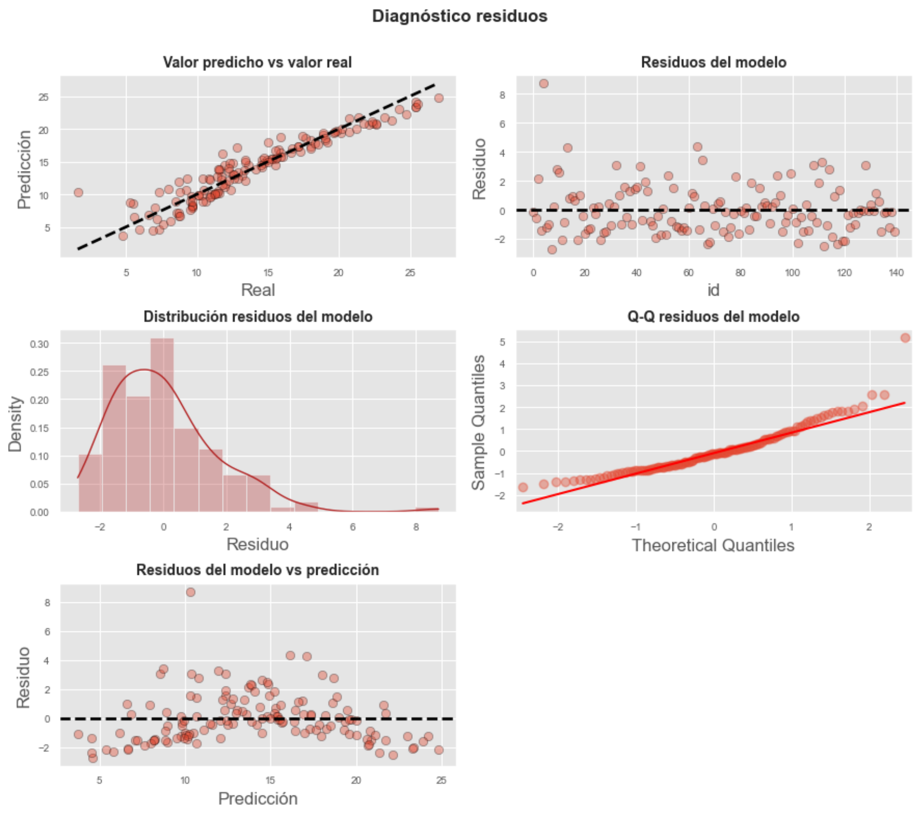
```
                                OLS Regression Results
     ==============================================================================
     Dep. Variable:                      y   R-squared:                       0.895
     Model:                            OLS   Adj. R-squared:                  0.893
     Method:                 Least Squares   F-statistic:                     581.1
     Date:                Thu, 27 Apr 2023   Prob (F-statistic):           1.20e-67
     Time:                        21:32:03   Log-Likelihood:                -271.80
     No. Observations:                 140   AIC:                             549.6
     Df Residuals:                     137   BIC:                             558.4
     Df Model:                           2
     Covariance Type:            nonrobust
     ==============================================================================
                      coef    std err          t      P>|t|      [0.025      0.975]
     ------------------------------------------------------------------------------
     const          2.9722      0.355      8.366      0.000       2.270       3.675
     tv             0.0460      0.002     26.914      0.000       0.043       0.049
     radio          0.1861      0.010     19.568      0.000       0.167       0.205
     ==============================================================================
     Omnibus:                       51.610   Durbin-Watson:                   2.131
     Prob(Omnibus):                  0.000   Jarque-Bera (JB):              147.995
     Skew:                          -1.431   Prob(JB):                     7.30e-33
     Kurtosis:                       7.144   Cond. No.                         413.
     ==============================================================================

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

```
print("Recta matematica de la regresion sin la variable periodico \n\n ventas = 0.046tv + 0.1861radio + 2.9722")
```

```
     Recta matematica de la regresion sin la variable periodico

      ventas = 0.046tv + 0.1861radio + 2.9722
```

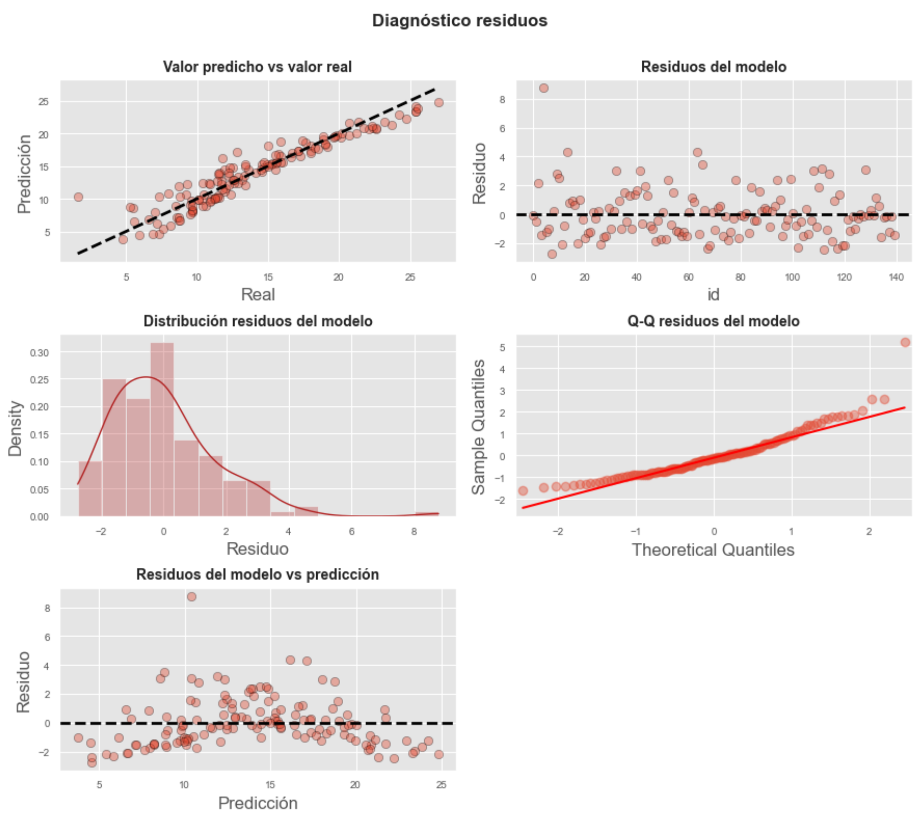
## Diagnostico de los Residuos

```
y_train = y_train.flatten()
prediccion_train = modelo.predict(exog = X_train)
residuos_train   = prediccion_train - y_train

fig, axes = plt.subplots(nrows=3, ncols=2, figsize=(9, 8))

axes[0, 0].scatter(y_train, prediccion_train, edgecolors=(0, 0, 0), alpha = 0.4)
axes[0, 0].plot([y_train.min(), y_train.max()], [y_train.min(), y_train.max()],
                'k--', color = 'black', lw=2)
axes[0, 0].set_title('Valor predicho vs valor real', fontsize = 10, fontweight = "bold")
axes[0, 0].set_xlabel('Real')
axes[0, 0].set_ylabel('Predicción')
axes[0, 0].tick_params(labelsize = 7)

axes[0, 1].scatter(list(range(len(y_train))), residuos_train,
                   edgecolors=(0, 0, 0), alpha = 0.4)
axes[0, 1].axhline(y = 0, linestyle = '--', color = 'black', lw=2)
axes[0, 1].set_title('Residuos del modelo', fontsize = 10, fontweight = "bold")
axes[0, 1].set_xlabel('id')
axes[0, 1].set_ylabel('Residuo')
axes[0, 1].tick_params(labelsize = 7)

sns.histplot(
    data    = residuos_train,
    stat    = "density",
    kde     = True,
    line_kws= {'linewidth': 1},
    color   = "firebrick",
    alpha   = 0.3,
    ax      = axes[1, 0]
)

axes[1, 0].set_title('Distribución residuos del modelo', fontsize = 10,
                     fontweight = "bold")
axes[1, 0].set_xlabel("Residuo")
axes[1, 0].tick_params(labelsize = 7)


sm.qqplot(
    residuos_train,
    fit   = True,
    line  = 'q',
    ax    = axes[1, 1],
    color = 'firebrick',
    alpha = 0.4,
    lw    = 2
)
axes[1, 1].set_title('Q-Q residuos del modelo', fontsize = 10, fontweight = "bold")
axes[1, 1].tick_params(labelsize = 7)

axes[2, 0].scatter(prediccion_train, residuos_train,
                   edgecolors=(0, 0, 0), alpha = 0.4)
axes[2, 0].axhline(y = 0, linestyle = '--', color = 'black', lw=2)
axes[2, 0].set_title('Residuos del modelo vs predicción', fontsize = 10, fontweight = "bold")
axes[2, 0].set_xlabel('Predicción')
axes[2, 0].set_ylabel('Residuo')
axes[2, 0].tick_params(labelsize = 7)

# Se eliminan los axes vacíos
fig.delaxes(axes[2,1])

fig.tight_layout()
plt.subplots_adjust(top=0.9)
fig.suptitle('Diagnóstico residuos', fontsize = 12, fontweight = "bold");
```



## Diagnostico de los Residuos sin variable Periodico

```
B_train = B_train.flatten()
prediccion_trainB = modeloB.predict(exog = A_train)
residuos_trainB   = prediccion_trainB - B_train

figB, axesB = plt.subplots(nrows=3, ncols=2, figsize=(9, 8))

axesB[0, 0].scatter(B_train, prediccion_trainB, edgecolors=(0, 0, 0), alpha = 0.4)
axesB[0, 0].plot([B_train.min(), B_train.max()], [B_train.min(), B_train.max()],
                'k--', color = 'black', lw=2)
axesB[0, 0].set_title('Valor predicho vs valor real', fontsize = 10, fontweight = "bold")
axesB[0, 0].set_xlabel('Real')
axesB[0, 0].set_ylabel('Predicción')
axesB[0, 0].tick_params(labelsize = 7)

axesB[0, 1].scatter(list(range(len(B_train))), residuos_trainB,
                   edgecolors=(0, 0, 0), alpha = 0.4)
axesB[0, 1].axhline(y = 0, linestyle = '--', color = 'black', lw=2)
axesB[0, 1].set_title('Residuos del modelo', fontsize = 10, fontweight = "bold")
axesB[0, 1].set_xlabel('id')
axesB[0, 1].set_ylabel('Residuo')
axesB[0, 1].tick_params(labelsize = 7)

sns.histplot(
    data    = residuos_trainB,
    stat    = "density",
    kde     = True,
    line_kws= {'linewidth': 1},
    color   = "firebrick",
    alpha   = 0.3,
    ax      = axesB[1, 0]
)

axesB[1, 0].set_title('Distribución residuos del modelo', fontsize = 10,
                     fontweight = "bold")
axesB[1, 0].set_xlabel("Residuo")
axesB[1, 0].tick_params(labelsize = 7)


sm.qqplot(
    residuos_trainB,
    fit   = True,
    line  = 'q',
    ax    = axesB[1, 1],
    color = 'firebrick',
    alpha = 0.4,
    lw    = 2
)
axesB[1, 1].set_title('Q-Q residuos del modelo', fontsize = 10, fontweight = "bold")
axesB[1, 1].tick_params(labelsize = 7)

axesB[2, 0].scatter(prediccion_trainB, residuos_trainB,
                   edgecolors=(0, 0, 0), alpha = 0.4)
axesB[2, 0].axhline(y = 0, linestyle = '--', color = 'black', lw=2)
axesB[2, 0].set_title('Residuos del modelo vs predicción', fontsize = 10, fontweight = "bold")
axesB[2, 0].set_xlabel('Predicción')
axesB[2, 0].set_ylabel('Residuo')
axesB[2, 0].tick_params(labelsize = 7)

# Se eliminan los axes vacíos
figB.delaxes(axesB[2,1])

figB.tight_layout()
plt.subplots_adjust(top=0.9)
figB.suptitle('Diagnóstico residuos', fontsize = 12, fontweight = "bold");
```



## Test de Normalidad

```
shapiro_test = stats.shapiro(residuos_train)
shapiro_test
```

```
    ShapiroResult(statistic=0.9120649695396423, pvalue=1.5303970712921e-07)
```

```
#Sin Periodico
shapiro_test = stats.shapiro(residuos_trainB)
shapiro_test
```

```
    ShapiroResult(statistic=0.9107311367988586, pvalue=1.2771810986578203e-07)
```

## Predicciones del modelo

```
X_test = sm.add_constant(X_test, prepend=True)
predicciones = modelo.predict(exog = X_test)

mae = mean_absolute_error(
        y_true  = y_test,
        y_pred  = predicciones)

mse = mean_squared_error(
        y_true  = y_test,
        y_pred  = predicciones,
        squared = True
       )

rmse = mean_squared_error(
        y_true  = y_test,
        y_pred  = predicciones,
        squared = False
       )

print(f"El error (mae) de test es: {mae} \n El error (mse) de test es: {mse} \n El error (rmse) de test es: {rmse}")
```

```
El error (mae) de test es: 1.2029579536281794
 El error (mse) de test es: 2.6697792772530184
 El error (rmse) de test es: 1.6339459223771815
```

```
A_test = sm.add_constant(A_test, prepend=True)
prediccionesB = modeloB.predict(exog = A_test)

mae = mean_absolute_error(
        y_true  = B_test,
        y_pred  = prediccionesB)

mse = mean_squared_error(
        y_true  = B_test,
        y_pred  = prediccionesB,
        squared = True
       )

rmse = mean_squared_error(
        y_true  = B_test,
        y_pred  = prediccionesB,
        squared = False
       )

print(f"El error (mae) de test es: {mae} \n El error (mse) de test es: {mse} \n El error (rmse) de test es: {rmse}")
```

```
El error (mae) de test es: 1.201979427367724
 El error (mse) de test es: 2.6595342544482565
 El error (rmse) de test es: 1.6308078533194081
```

## Interpretacion

### Caso 1

El modelo posee un 89.2% de la varianza observada de las ventas y su ecuacion de la recta es la siguiente

```
ventas = 0.0461tv + 0.1854radio + 0.0016periodico + 2.9389
```

En cuanto al F-statistic es de 3.19e-66, el cual nos indica que el modelo es significativo y en conjunto al R-squared, nos indica el modelo podria predecir la variable respuesta.

Por otro lado, en el test de normalidad, el resultado fue menor al 5% y que se deberia rechazar la hipotesis y que el modelo no posee una distribucion normal.

Las predicciones del modelo final se alejan en promedio 1.633 unidades (rmse) del valor real.

### Caso 2 sin variable "Periodico"

Sin embargo la variable "Periodico" posee un p-value > 0.05 por lo que se podria eliminarse del modelo.

Al eliminar la variable "Periodico", el modelo ahora posee un 89.3% de la varianza observada y su ecuacion de la recta es la siguiente

```
ventas = 0.046tv + 0.1861radio + 2.9722
```

En cuanto al F-statistic es de 1.20e-67, el cual nos indica que el modelo es significativo y en conjunto al R-squared, nos indica el modelo podria predecir la variable respuesta.

Al igual que en el anterior caso, en el test de normalidad, el resultado fue menor al 5% y que se deberia rechazar la hipotesis y que el modelo no posee una distribucion normal.

Las predicciones del modelo final se alejan en promedio 1.630 unidades (rmse) del valor real.