Análisis cuantiativo - Examen 1

Participantes:

- Kevin Rodriguez
- Luis Vasquez
- Alejandro Martinez
- Jhonatan Valencia

Punto 1

1) Considere el conjunto de datos "data1" del fichero data_exam1.xlsx.

- Realice un análisis exploratorio de datos ¿Considera que podría generar un modelo de regresión lineal con variable categórica (sin interacción) para la variable **Y**? Justifique. Si la respuesta a la pregunta es SI, genere un modelo de regresión sin interacción e interpretelo.

Analisis exploratorio de los datos

```
import pandas as pd

data_1 = pd.read_excel('data_exam1.xlsx', sheet_name='data1')
data_1.head()
```

```
           Y          X  Ind
0  66.199147  12.653765    0
1  44.311301   8.204418    0
2  48.390783   8.768596    0
3  58.087413  16.169568    1
4  60.708671   9.980310    0
```

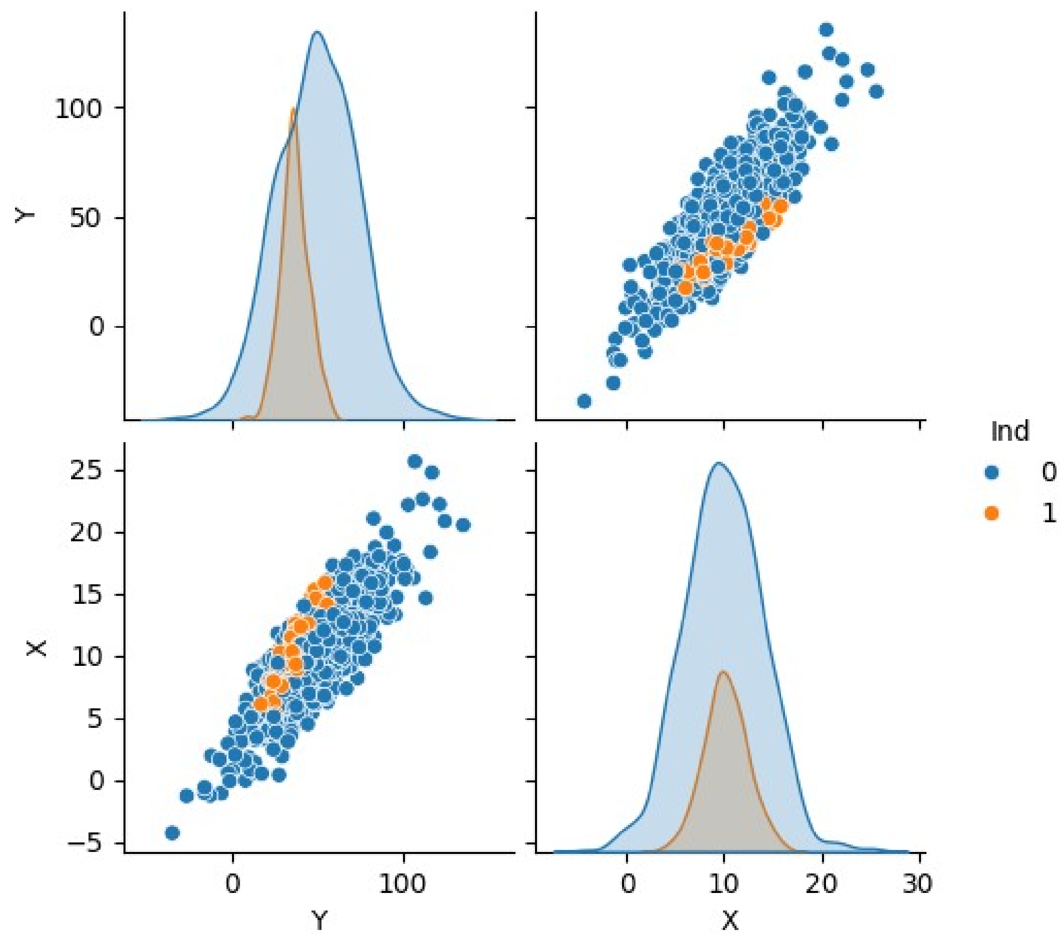
```
data_1.describe()
```

```
                 Y            X        Ind
count  1000.000000  1000.000000  1000.0000
mean     46.953751     9.976858     0.2000
std      22.046143     3.762567     0.4002
min     -34.894319    -4.263757     0.0000
25%      32.427643     7.638899     0.0000
50%      45.460252     9.952888     0.0000
75%      61.587567    12.379984     0.0000
max     135.542574    25.628678     1.0000
```

```
data_1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 3 columns):
```

```
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   Y       1000 non-null   float64
 1   X       1000 non-null   float64
 2   Ind     1000 non-null   int64
dtypes: float64(2), int64(1)
memory usage: 23.6 KB

import matplotlib.pyplot as plt
import seaborn as sns

sns.pairplot(data_1, hue='Ind')

<seaborn.axisgrid.PairGrid at 0x215abd11f10>
```

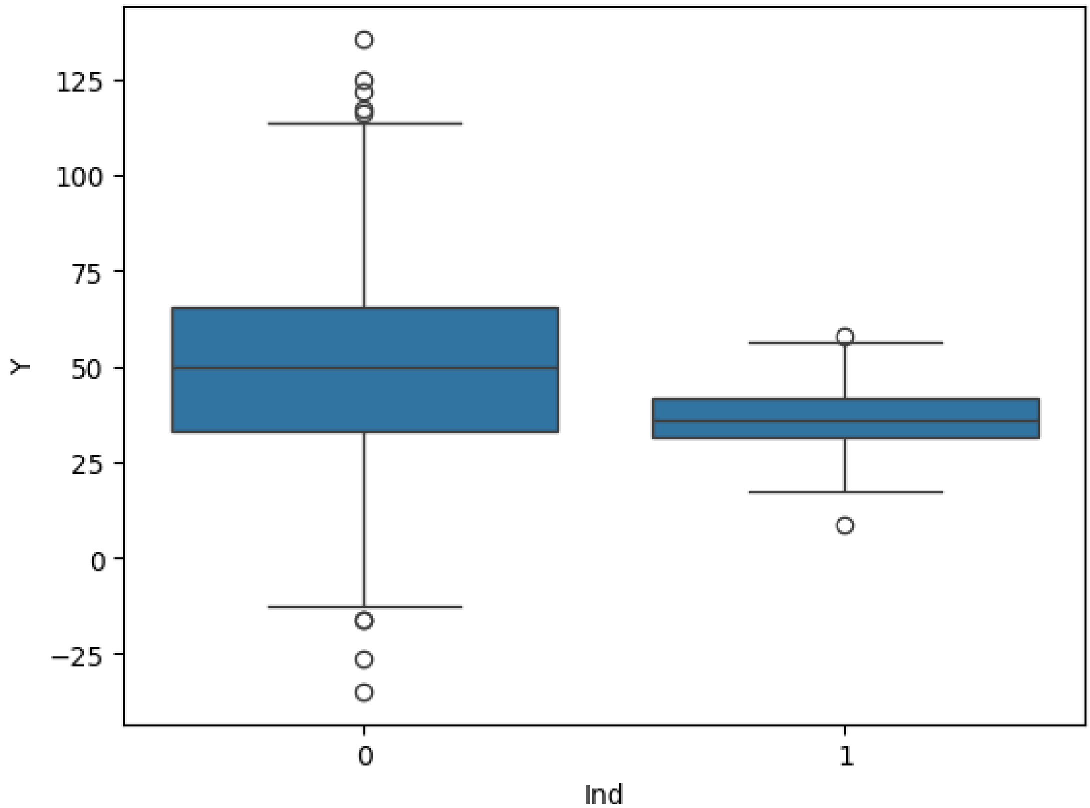


Boxplot de la variable indicadora y los valores en Y.

Como se puede observar, no existe un desplazamiento de las cajas como uno esperaria en unos datos sin interacción, lo que supone que no se puede hacer un modelo de regresion sin interacción

```python
sns.boxplot(x='Ind', y='Y', data=data_1)
```

```
<Axes: xlabel='Ind', ylabel='Y'>
```




```python
datos_ind_0 = data_1[data_1['Ind'] == 0]
datos_ind_1 = data_1[data_1['Ind'] == 1]

print(datos_ind_0.corr(method='pearson'))
print(datos_ind_0.corr(method='spearman'))
print(datos_ind_0.corr(method='kendall'))
```

```
            Y         X  Ind
Y    1.000000  0.866559  NaN
X    0.866559  1.000000  NaN
Ind       NaN       NaN  NaN
            Y         X  Ind
Y    1.000000  0.855324  NaN
X    0.855324  1.000000  NaN
Ind       NaN       NaN  NaN
            Y         X  Ind
Y    1.000000  0.664193  NaN
X    0.664193  1.000000  NaN
Ind       NaN       NaN  1.0
```

```
print(datos_ind_1.corr(method='pearson'))
print(datos_ind_1.corr(method='spearman'))
print(datos_ind_1.corr(method='kendall'))
```

```
            Y         X  Ind
Y    1.000000  0.867989  NaN
X    0.867989  1.000000  NaN
Ind       NaN       NaN  NaN
            Y         X  Ind
Y    1.000000  0.856319  NaN
X    0.856319  1.000000  NaN
Ind       NaN       NaN  NaN
            Y         X  Ind
Y    1.000000  0.672965  NaN
X    0.672965  1.000000  NaN
Ind       NaN       NaN  1.0
```

Aqui se puede observar los valores de los beta para los datos con Ind = 0 e Ind = 1. En un modelo sin interaccion se esperaria que las pendientes fueran iguales lo que generaria un paralelismo. Sin embargo, se puede observar que las pendientes son bastante distintas, tipico de un conjunto de datos con interaccion

```
import numpy as np
beta_1_datos_ind_0 = np.cov(datos_ind_0['X'], datos_ind_0['Y'])[0, 1]
/ np.var(datos_ind_0['X'])
beta_1_datos_ind_0
```

```
5.047437831283938
```

```
beta_1_datos_ind_1 = np.cov(datos_ind_1['X'], datos_ind_1['Y'])[0, 1]
/ np.var(datos_ind_1['X'])
beta_1_datos_ind_1
```

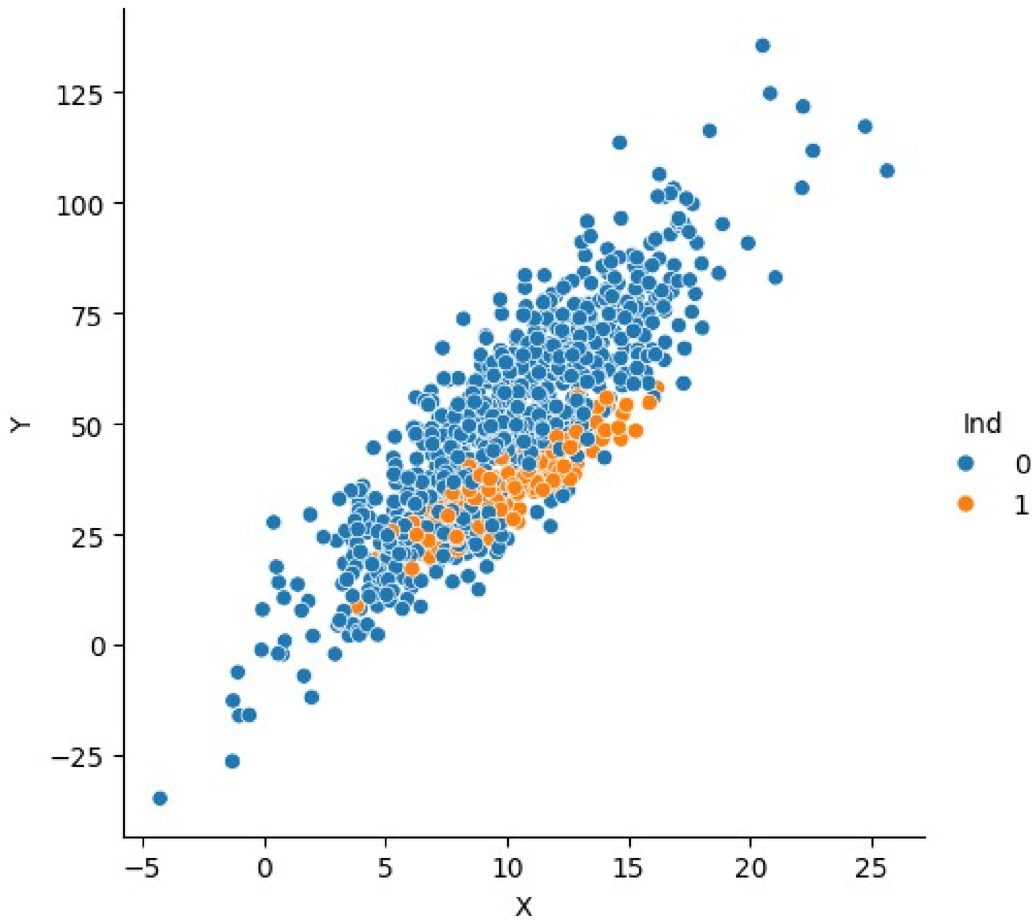
```
3.2105412208964688
```

- Realice un gráfico de dispersión para **Y** vs **X**, considerando para cada observación su respectivo valor en la variable Ind ¿Hay evidencia muestral que sugiera un cambio en la tasa media de cambio de Ycondicionado a incrementos unitarios de **X**? ¿Considera que un modelo con interacciones sería más adecuado? Si la respuesta a estas preguntas es afirmativa, genere el respectivo modelo, interprete detalladamente los resultados y valide los supuestos del modelo propuesto.

Grafico de dispersión para X vs Y

```
sns.relplot(x='X', y='Y', data=data_1, hue='Ind')
```

```
<seaborn.axisgrid.FacetGrid at 0x215c88ab990>
```



Se puede observar 2 cosas:

1. Existen cambios en Y condicionado a incrementos unitarios en X
2. Si se considera un modelo con interacciones como una mejor aproximación.

```
import statsmodels.api as sm

data_1['X_IND_interaction'] = data_1['X'] * data_1['Ind']
x = sm.add_constant(data_1.drop("Y", axis=1))
y = data_1["Y"]

model = sm.OLS(y, x).fit()
model.summary()

<class 'statsmodels.iolib.summary.Summary'>
"""
                              OLS Regression Results
==============================================================================
Dep. Variable:                      Y   R-squared:
0.765
```

```
Model:                            OLS   Adj. R-squared:
0.764
Method:                 Least Squares   F-statistic:
1081.
Date:                Sat, 27 Apr 2024   Prob (F-statistic):
1.34e-312
Time:                        12:55:55   Log-Likelihood:
-3787.5
No. Observations:                1000   AIC:
7583.
Df Residuals:                     996   BIC:
7603.
Df Model:                           3

Covariance Type:            nonrobust

======================================================================
===============
                        coef    std err          t      P>|t|
[0.025      0.975]
----------------------------------------------------------------------
---------------
const                -0.4991      1.001     -0.498      0.618      -
2.464       1.466
X                     5.0411      0.093     53.997      0.000
4.858       5.224
Ind                   4.5491      3.674      1.238      0.216      -
2.661      11.759
X_IND_interaction    -1.8466      0.353     -5.239      0.000      -
2.538      -1.155
======================================================================
========
Omnibus:                        4.301   Durbin-Watson:
1.985
Prob(Omnibus):                  0.116   Jarque-Bera (JB):
4.811
Skew:                           0.065   Prob(JB):
0.0902
Kurtosis:                       3.314   Cond. No.
119.
======================================================================
========

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

La formula quedaría de la siguiente forma

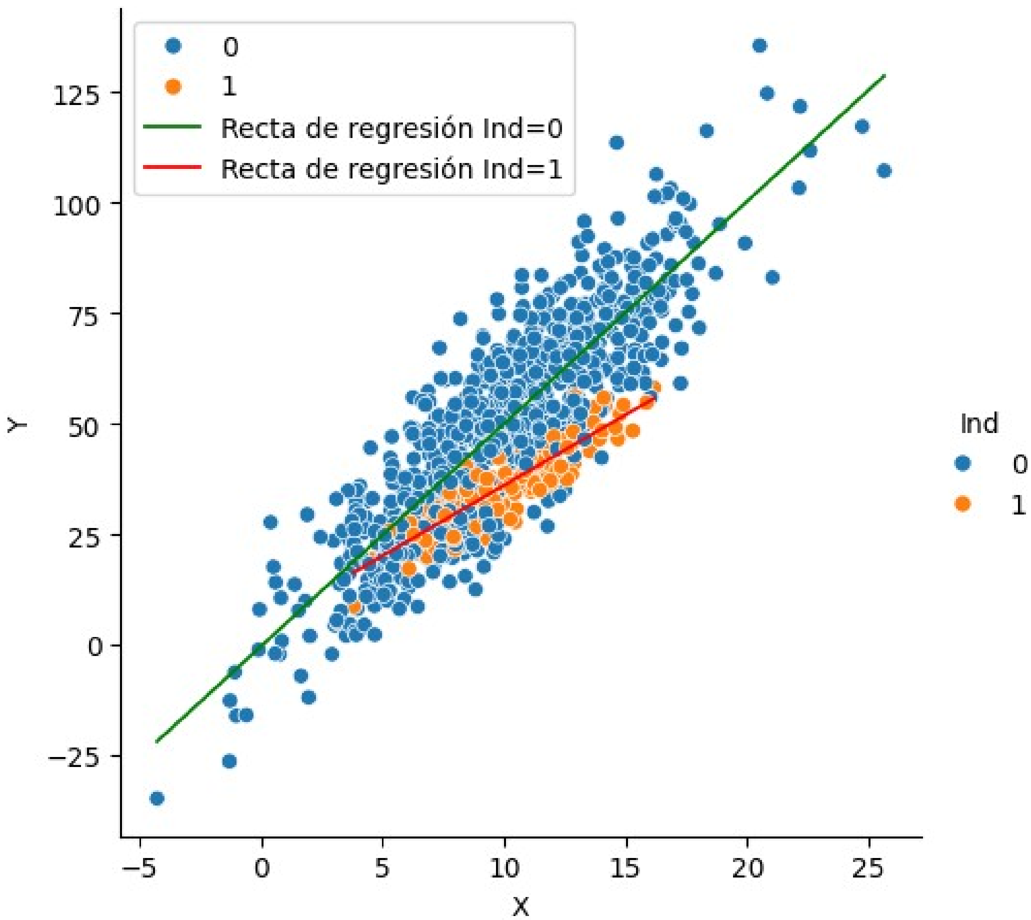
$Y = \beta_0 + \beta_1 X + \beta_2 Ind + \beta_3 (X\ Ind)$

Y = -0.4991 + 5.0411 X + 4.5491 Ind - 1.8466 (X Ind)

```
coefs = list(model.params)
coefs
```

```
[-0.4990969581235204,
 5.0411285339948275,
 4.549081221693464,
 -1.8466400192028478]
```

```
sns.relplot(x='X', y='Y', data=data_1, hue='Ind')

data_1['Y_pred'] = coefs[0] + coefs[1] * data_1['X'] + coefs[2] *
data_1['Ind'] + coefs[3] * data_1['X_IND_interaction']
data_1 = data_1.sort_values(by='X')
data_1_0 = data_1[data_1['Ind'] == 0]
data_1_1 = data_1[data_1['Ind'] == 1]

plt.plot(data_1_0['X'], data_1_0['Y_pred'], color='green',
label='Recta de regresión Ind=0', linewidth=1.3)
plt.plot(data_1_1['X'], data_1_1['Y_pred'], color='red', label='Recta
de regresión Ind=1', linewidth=1.3)
plt.legend()
plt.show()
```



Interpretación
En la figura anterior, se empleó un modelo ajustado con interacción sobre los datos originales. Se evidencia que la línea se alinea con el conjunto de puntos cuando Ind=0. Sin embargo, al probar los datos con Ind=1, la línea modifica su trayectoria para adaptarse a los valores originales.

```
model.summary()

<class 'statsmodels.iolib.summary.Summary'>
"""
                            OLS Regression Results

==============================================================================
========
Dep. Variable:                      Y   R-squared:
0.765
Model:                            OLS   Adj. R-squared:
0.764
Method:                 Least Squares   F-statistic:
1081.
Date:                Sat, 27 Apr 2024   Prob (F-statistic):
```

```
1.34e-312
Time:                            12:55:55   Log-Likelihood:
-3787.5
No. Observations:                    1000   AIC:
7583.
Df Residuals:                         996   BIC:
7603.
Df Model:                               3

Covariance Type:            nonrobust

======================================================================
===============
                        coef    std err          t      P>|t|
[0.025      0.975]
----------------------------------------------------------------------
---------------
const                -0.4991      1.001     -0.498      0.618      -
2.464       1.466
X                     5.0411      0.093     53.997      0.000
4.858       5.224
Ind                   4.5491      3.674      1.238      0.216      -
2.661      11.759
X_IND_interaction    -1.8466      0.353     -5.239      0.000      -
2.538      -1.155
======================================================================
========
Omnibus:                        4.301   Durbin-Watson:
1.985
Prob(Omnibus):                  0.116   Jarque-Bera (JB):
4.811
Skew:                           0.065   Prob(JB):
0.0902
Kurtosis:                       3.314   Cond. No.
119.
======================================================================
========

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

residuals = model.resid
residuals

0       2.908989
1       3.450871
2       4.686263
3       2.383929
```

```
4       10.895744
          ...
995     20.564510
996      3.730920
997    -19.979872
998     13.883818
999      1.957304
Length: 1000, dtype: float64
```

Prueba de normalidad

```
p_value = sm.stats.jarque_bera(residuals)[1]
print(f"P-Value JB: {p_value}")
```

```
P-Value JB: 0.09020850574930754
```

Se cumple normalidad

Independencia

```
from statsmodels.stats.stattools import durbin_watson

dw_statistic = durbin_watson(residuals)
print("Estadístico de Durbin-Watson:", dw_statistic)
```

```
Estadístico de Durbin-Watson: 1.984805425927445
```

Se cumple independencia

Media cero

```
from scipy.stats import ttest_1samp

t_statistic, p_value = ttest_1samp(residuals, 0)
print("p-valor de la prueba de t de Student:", p_value)
```

```
p-valor de la prueba de t de Student: 0.9999999999999968
```

Se cumple media cero

Homocedasticidad

```
from statsmodels.stats.diagnostic import het_breuschpagan

lm, lm_p_value, fvalue, f_p_value = het_breuschpagan(residuals,
model.model.exog)

print("Estadístico Lagrange Multiplier:", lm)
print("P-valor Lagrange Multiplier:", lm_p_value)
print("Estadístico F:", fvalue)
print("P-valor F:", f_p_value)
```

```
Estadístico Lagrange Multiplier: 83.04028836791721
P-valor Lagrange Multiplier: 6.835236597869036e-18
Estadístico F: 30.06607093901455
P-valor F: 1.3067597184142322e-18
```

No se cumple homocedasticidad

Punto 2

2) Considere el conjunto de datos “data2” del fichero data_exam1.xlsx.

- Realice un análisis exploratorio de datos, tanto univariante como bivariante ¿Qué puede decir acerca del comportamiento distribucional de cada variable? ¿Considera que la dispersión bivariante da indicios para generar un modelo de regresión para **Y**? Justifique detalladamente.

Analisis exploratorio de los datos

```python
import pandas as pd

data_2 = pd.read_excel('data_exam1.xlsx', sheet_name='data2')
data_2.head()
```

```
           Y         X
0  12.189142  0.226957
1  12.187456  0.088938
2  11.782692  0.199069
3   5.732032  0.003812
4   7.026970  0.004573
```

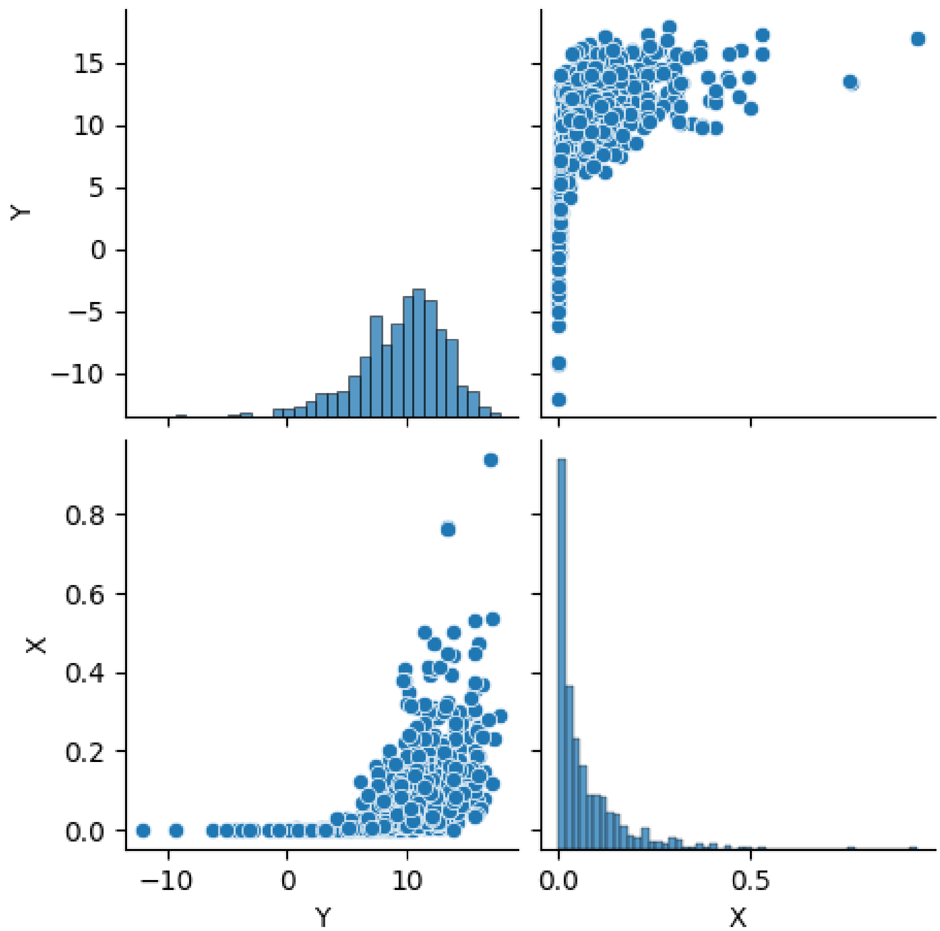
```python
data_2.describe()
```

```
                 Y             X
count  1000.000000  1.000000e+03
mean      9.445622  7.234805e-02
std       3.908189  9.753985e-02
min     -12.073239  1.343729e-08
25%       7.411486  8.450417e-03
50%      10.072134  3.655172e-02
75%      12.082546  9.992523e-02
max      17.838788  9.397465e-01
```

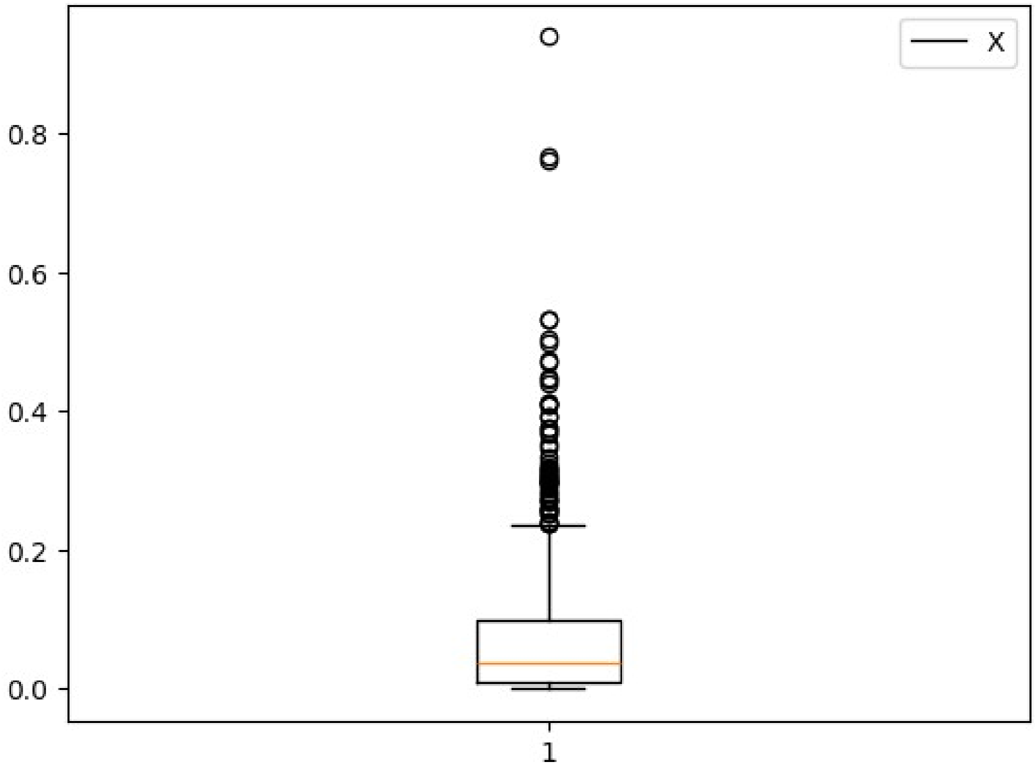
```python
data_2.info()
```

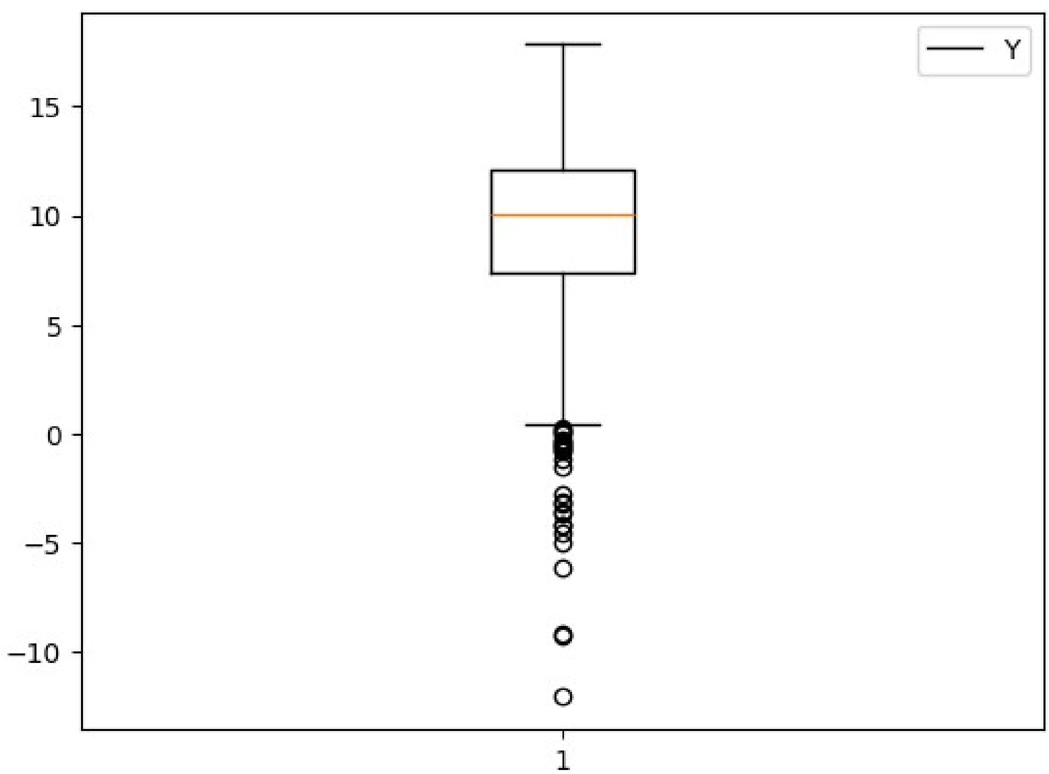
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
```

```
 0   Y        1000 non-null   float64
 1   X        1000 non-null   float64
dtypes: float64(2)
memory usage: 15.8 KB

import seaborn as sns

sns.pairplot(data_2)

<seaborn.axisgrid.PairGrid at 0x215ca8f6190>
```




```
import matplotlib.pyplot as plt

plt.boxplot(data_2['X'])
plt.legend(['X'])
plt.figure()
plt.boxplot(data_2['Y'])
plt.legend(['Y'])

<matplotlib.legend.Legend at 0x215cab3d610>
```


X
0.8
0.6
0.4
0.2
0.0
1


Y
15
10
5
0
−5
−10
1

Interpretacion
A partir del análisis exploratorio de de los datos, tanto univariante como bivariante, se obtuvieron las siguientes conclusiones:

- Los valores de X e Y se mueven en rangos muy distintos, para el caso de X son valores pequeños cercanos a 0 y en el caso de Y se mueve en valores más grandes.
- Los valores de X tienen una distribución exponencial
- Los valores en Y tienen una distribucion normal pero desplazada hacia la derecha
- La nube de puntos no presenta un comportamiento que pueda ser representado por una linea.

Teniendo en cuenta estas y más observaciones, no se considera oportuno realizar un modelo de regresión lineal para Y sin hacer algún tipo de transformación.

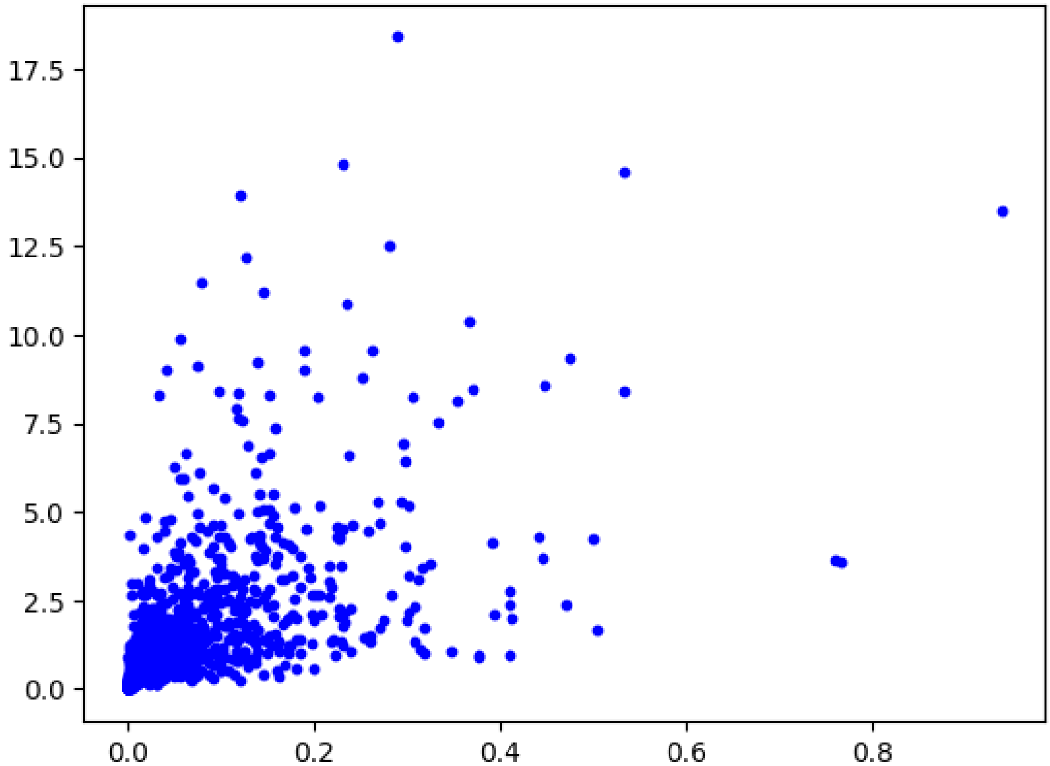
- De acuerdo al análisis del ítem anterior proponga una transformación (raiz, potencia, logarítmica, sinusoidal, etc.) para alguna de las variables y justifique por qué. Dado lo anterior, proponga un modelo de regresión lineal, interpretelo y valide los supuestos del modelo.

Transformación

Después de observar el comportamiento bivariante de los valores X e Y, se considera que su comportamiento tiene similitud con un logaritmo, por lo que se procede a sacar la inversa al usar como exponente a los valores de Y con una base definida

```
data_2['Y_transform'] = 1.45**(data_2['Y'] - 10)
plt.scatter(data_2['X'], data_2['Y_transform'], c='blue', s=10)
```

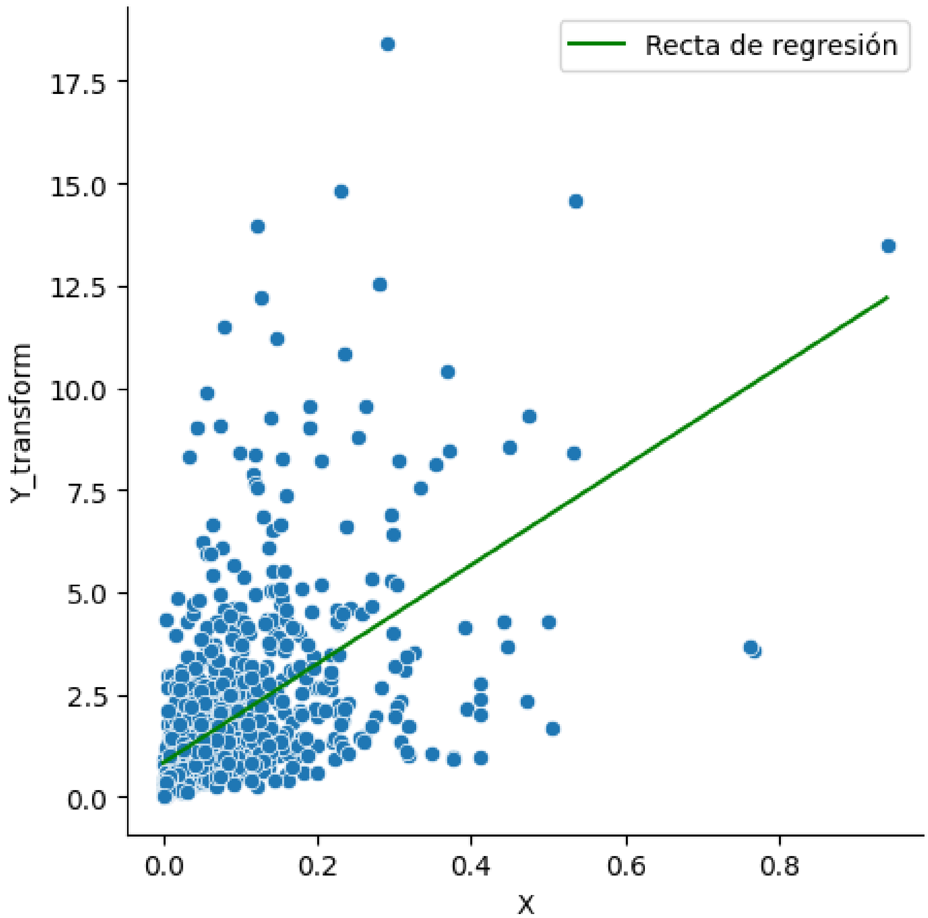
```
<matplotlib.collections.PathCollection at 0x215cabe6410>
```




```
import statsmodels.api as sm

x = sm.add_constant(data_2.drop(["Y", "Y_transform"], axis=1))
y = data_2["Y_transform"]

model = sm.OLS(y, x).fit()
model.summary()

<class 'statsmodels.iolib.summary.Summary'>
"""
                            OLS Regression Results

======================================================================
========
Dep. Variable:            Y_transform   R-squared:
0.308
Model:                            OLS   Adj. R-squared:
0.308
Method:                 Least Squares   F-statistic:
444.8
Date:                Sat, 27 Apr 2024   Prob (F-statistic):
5.94e-82
Time:                        12:55:56   Log-Likelihood:
-1988.0
No. Observations:                1000   AIC:
3980.
```

```
Df Residuals:                     998   BIC:
3990.
Df Model:                           1

Covariance Type:            nonrobust

==============================================================================
========
                 coef    std err          t      P>|t|      [0.025
0.975]
------------------------------------------------------------------------------
--------
const          0.8285      0.070     11.898      0.000       0.692
0.965
X             12.0973      0.574     21.091      0.000      10.972
13.223
==============================================================================
========
Omnibus:                      599.839   Durbin-Watson:
2.051
Prob(Omnibus):                  0.000   Jarque-Bera (JB):
7638.048
Skew:                           2.527   Prob(JB):
0.00
Kurtosis:                      15.561   Cond. No.
10.3
==============================================================================
========

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

sns.relplot(x='X', y='Y_transform', data=data_2, )

coefs = model.params
data_2['Y_pred'] = coefs[0] + coefs[1] * data_2['X']
data_2 = data_2.sort_values(by='X')

plt.plot(data_2['X'], data_2['Y_pred'], color='green', label='Recta de
regresión', linewidth=1.5)
plt.legend()
plt.show()

C:\Users\GlobE\AppData\Local\Temp\ipykernel_15556\3159884870.py:4:
FutureWarning: Series.__getitem__ treating keys as positions is
deprecated. In a future version, integer keys will always be treated
as labels (consistent with DataFrame behavior). To access a value by
```

```
position, use `ser.iloc[pos]`
  data_2['Y_pred'] = coefs[0] + coefs[1] * data_2['X']
```




```
residuals = model.resid
```

Prueba de normalidad

```
p_value = sm.stats.jarque_bera(residuals)[1]
print(f"P-Value JB: {p_value}")

P-Value JB: 0.0
```

No se cumple normalidad

Independencia

```
from statsmodels.stats.stattools import durbin_watson

dw_statistic = durbin_watson(residuals)
print("Estadístico de Durbin-Watson:", dw_statistic)
```

```
Estadístico de Durbin-Watson: 2.05119674385883
```

Existe independencia

Media cero

```python
from scipy.stats import ttest_1samp

t_statistic, p_value = ttest_1samp(residuals, 0)
print("p-valor de la prueba de t de Student:", p_value)
```

```
p-valor de la prueba de t de Student: 0.999999999999992
```

Existe media cero

Homocedasticidad

```python
from statsmodels.stats.diagnostic import het_breuschpagan

lm, lm_p_value, fvalue, f_p_value = het_breuschpagan(residuals,
model.model.exog)

print("Estadístico Lagrange Multiplier:", lm)
print("P-valor Lagrange Multiplier:", lm_p_value)
print("Estadístico F:", fvalue)
print("P-valor F:", f_p_value)
```

```
Estadístico Lagrange Multiplier: 91.07191156922589
P-valor Lagrange Multiplier: 1.3854444964246184e-21
Estadístico F: 99.99665419406803
P-valor F: 1.677162191370048e-22
```

No se cumple homocedasticidad

Punto 3

3) Considere el conjunto de datos "Wine Quality" del fichero datos.xls. Defina como variable respuesta (*Y*) la columna Densidad y elimine las variables pH, Sulfatos, Cloruros, Acidez Volátil, Acidez Fija y Calidad de Vino.

Carga de los datos

```python
import pandas as pd

df = pd.read_excel('datos.xls', sheet_name='Wine Quality', skiprows=2)
df
```

```
      Calidad del Vino  Acidez Fija  Acidez Volátil  Ácido Cítrico  \
0                    6          7.0            0.27           0.36
1                    6          6.3            0.30           0.34
```

```
2                        6            8.1              0.28              0.40
3                        6            7.2              0.23              0.32
4                        6            7.2              0.23              0.32
...                    ...            ...               ...               ...
4893                     6            6.2              0.21              0.29
4894                     5            6.6              0.32              0.36
4895                     6            6.5              0.24              0.19
4896                     7            5.5              0.29              0.30
4897                     6            6.0              0.21              0.38

      Azúcar Residual  Cloruros  Dióxido de Azúfre Libre  \
0                20.7     0.045                     45.0
1                 1.6     0.049                     14.0
2                 6.9     0.050                     30.0
3                 8.5     0.058                     47.0
4                 8.5     0.058                     47.0
...               ...       ...                      ...
4893              1.6     0.039                     24.0
4894              8.0     0.047                     57.0
4895              1.2     0.041                     30.0
4896              1.1     0.022                     20.0
4897              0.8     0.020                     22.0

      Dióxido de Azúfre Total  Densidad    pH  Sulfatos  Alcohol
0                       170.0   1.00100  3.00      0.45      8.8
1                       132.0   0.99400  3.30      0.49      9.5
2                        97.0   0.99510  3.26      0.44     10.1
3                       186.0   0.99560  3.19      0.40      9.9
4                       186.0   0.99560  3.19      0.40      9.9
...                       ...       ...   ...       ...      ...
4893                     92.0   0.99114  3.27      0.50     11.2
4894                    168.0   0.99490  3.15      0.46      9.6
4895                    111.0   0.99254  2.99      0.46      9.4
4896                    110.0   0.98869  3.34      0.38     12.8
4897                     98.0   0.98941  3.26      0.32     11.8

[4898 rows x 12 columns]
```

```
y = df['Densidad']
x = df.drop(columns=['Densidad', 'pH', 'Sulfatos', 'Cloruros', 'Acidez Volátil', 'Acidez Fija', 'Calidad del Vino'])
df_full= pd.concat([x, y], axis=1)
```

- Estandarice las variables, calcule las matrices de correlación de Pearson ($\hat{\rho}(P)$), Kendall($\hat{\rho}(K)$) y Spearman ($\hat{\rho}(Sp)$) y compárelas ¿Qué diferencia encuentra entre las estructuras de dependencias obtenidas?

```
mean = df_full.mean()
std = df_full.std()
```

```
df_full = (df_full - mean) / std
df_full.head()
```

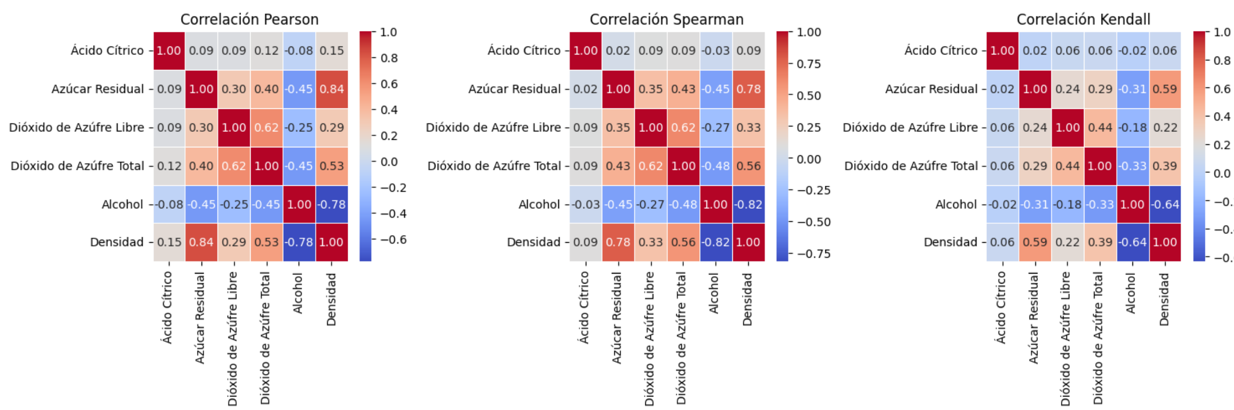
```
   Ácido Cítrico  Azúcar Residual  Dióxido de Azúfre Libre  \
0       0.213258         2.821061                 0.569873
1       0.047996        -0.944669                -1.252891
2       0.543783         0.100272                -0.312109
3      -0.117266         0.415726                 0.687471
4      -0.117266         0.415726                 0.687471

   Dióxido de Azúfre Total   Alcohol  Densidad
0                 0.744489 -1.393010  2.331274
1                -0.149669 -0.824192 -0.009153
2                -0.973236 -0.336633  0.358628
3                 1.120977 -0.499152  0.525802
4                 1.120977 -0.499152  0.525802
```

```
y = df_full['Densidad']
x = df_full.drop(columns=['Densidad'])

df_full.head()
```

```
   Ácido Cítrico  Azúcar Residual  Dióxido de Azúfre Libre  \
0       0.213258         2.821061                 0.569873
1       0.047996        -0.944669                -1.252891
2       0.543783         0.100272                -0.312109
3      -0.117266         0.415726                 0.687471
4      -0.117266         0.415726                 0.687471

   Dióxido de Azúfre Total   Alcohol  Densidad
0                 0.744489 -1.393010  2.331274
1                -0.149669 -0.824192 -0.009153
2                -0.973236 -0.336633  0.358628
3                 1.120977 -0.499152  0.525802
4                 1.120977 -0.499152  0.525802
```

```
import seaborn as sns
import matplotlib.pyplot as plt

corr_pearson = df_full.corr(method='pearson')
corr_spearman = df_full.corr(method='spearman')
corr_kendall = df_full.corr(method='kendall')

plt.figure(figsize=(15, 5))

plt.subplot(1, 3, 1)
sns.heatmap(corr_pearson, annot=True, cmap='coolwarm', fmt=".2f",
linewidths=.5)
plt.title('Correlación Pearson')

plt.subplot(1, 3, 2)
```

```
sns.heatmap(corr_spearman, annot=True, cmap='coolwarm', fmt=".2f",
linewidths=.5)
plt.title('Correlación Spearman')

plt.subplot(1, 3, 3)
sns.heatmap(corr_kendall, annot=True, cmap='coolwarm', fmt=".2f",
linewidths=.5)
plt.title('Correlación Kendall')

plt.tight_layout()

plt.show()
```



Observaciones

Se pueden detallar los siguientes aspectos según las matrices de correlación:

- Los valores en cada una de las matrices de correlación presentan el mismo comportamiento respecto a los datos, como son la relación entre densidad y alcohol, densidad y azúcar entre otros.
- Los valores en la matriz de correlación de Kendall presentan valores más cercanos a cero en comparación con las otras matrices.
- Las matrices de Pearson y Spearman presentan los valores más similares al momento de mostrar la relación entre las variables.
- Realice una partición de los datos tipo 80–20, donde el primer 80 % de los datos es una muestra de entrenamiento y el restante 20 % una muestra de prueba/predicción. Luego, construya 3 modelos RLM con las matrices estimadas en el primer ítem($\hat{\beta}(\cdot) = \hat{\rho}^{-1}(\cdot)_{XX} \hat{\rho}(\cdot)_{XY}$ y $\hat{\beta}_0(\cdot) = \hat{\mu}_Y - \hat{\mu}_X \hat{\beta}(\cdot)$). Compare e interprete los valores de los coeficientes de regresión obtenidos por cada método.

Partición de datos 80/20

```
from sklearn.model_selection import train_test_split

x_train, x_test, y_train, y_test = train_test_split(x, y,
test_size=0.2, random_state=42)
```

```
import numpy as np

def Betas_RLM(corr_matrix, x, y):
    # Beta values
    corr_matrix_xx = corr_matrix.drop('Densidad', axis=0)
    corr_matrix_xx = corr_matrix_xx.drop('Densidad', axis=1)

    corr_matrix_xy = corr_matrix['Densidad'].drop('Densidad')

    beta_matrix = np.matmul(np.linalg.inv(corr_matrix_xx),
corr_matrix_xy)

    #Beta zero
    beta_zero = y.mean() - np.matmul(x.mean(), beta_matrix)

    return beta_matrix, beta_zero

beta_pearson, beta_zero_pearson = Betas_RLM(corr_pearson, x_train,
y_train)
beta_spearman, beta_zero_spearman = Betas_RLM(corr_spearman, x_train,
y_train)
beta_kendall, beta_zero_kendall = Betas_RLM(corr_kendall, x_train,
y_train)

print('Beta Pearson:', beta_pearson, 'Beta 0 Pearson:',
beta_zero_pearson)
print('Beta Spearman:', beta_spearman, 'Beta 0 Spearman:',
beta_zero_spearman)
print('Beta Kendall:', beta_kendall, 'Beta 0 Kendall:',
beta_zero_kendall)
```

```
Beta Pearson: [ 0.05038549  0.59646843 -0.08299052  0.12306553 -
0.4730478 ] Beta 0 Pearson: 0.006016060388578296
Beta Spearman: [ 0.05825041  0.50470699 -0.0815831   0.1252712  -
0.55796176] Beta 0 Spearman: 0.006845457037379612
Beta Kendall: [ 0.03964713  0.4139839  -0.02310357  0.12074798 -
0.47267202] Beta 0 Kendall: 0.008753438457285993
```

Interpretación de los valores

Se consideran las siguientes interpretaciones:

- El valor de beta_0 para cada modelo representa el lugar de intercepto, por lo que cada uno presenta un intercepto parecido cercano a 1, donde usando la matriz de Pearson, se obtiene el intercepto más bajo.
- Los valores de beta de Pearson y Spearman resultan los más similares, a diferencia de Kendall.
- Se observa la influencia de la matriz de correlación en cada uno de los modelos.

```
#def RLM_predict(beta, beta_zero, x):
#    return beta_zero + beta[0]*x['Ácido Cítrico'] + beta[1]*x['Azúcar
```

```
Residual'] + beta[2]*x['Dióxido de Azúfre Libre'] + beta[3]*x['Dióxido
de Azúfre Total'] + beta[4]*x['Alcohol']
def RLM_predict(beta, beta_zero, x):
    return beta_zero + np.matmul(x, beta)

RLM_predict_pearson = RLM_predict(beta_pearson, beta_zero_pearson,
x_test)
RLM_predict_spearman = RLM_predict(beta_spearman, beta_zero_spearman,
x_test)
RLM_predict_kendall = RLM_predict(beta_kendall, beta_zero_kendall,
x_test)
```

- Realice una predicción con los datos de prueba de acuerdo a los modelos ajustados y calcule el RMSE (√MSE) de la predicción ¿Cuál de los modelos lineales propuestos predice mejor?

```
from sklearn.metrics import mean_squared_error

mse_pearson = mean_squared_error(y_test, RLM_predict_pearson)
mse_spearman = mean_squared_error(y_test, RLM_predict_spearman)
mse_kendall = mean_squared_error(y_test, RLM_predict_kendall)
print('MSE Pearson:', mse_pearson)
print('MSE Spearman:', mse_spearman)
print('MSE Kendall:', mse_kendall)

MSE Pearson: 0.06553269164056633
MSE Spearman: 0.07030885473560868
MSE Kendall: 0.0869543635262268

print('RMSE Pearson:', mse_pearson**0.5)
print('RMSE Spearman:', mse_spearman**0.5)
print('RMSE Kendall:', mse_kendall**0.5)

RMSE Pearson: 0.2559935382789306
RMSE Spearman: 0.2651581692794108
RMSE Kendall: 0.29488025285906616
```

Interpretación de los resultados

El modelo que mejor predice es el que utiliza la Matriz de Correlación de Pearson.

- Valide los supuestos teóricos de cada modelo (εiiid∼ N(0,σ2)) y concluya.

Validación de los supuestos

```
residuals_pearson = y_test - RLM_predict_pearson
residuals_spearman = y_test - RLM_predict_spearman
residuals_kendall = y_test - RLM_predict_kendall

from statsmodels.stats.stattools import durbin_watson
import statsmodels.api as sm
```

```python
from scipy.stats import ttest_1samp
from statsmodels.stats.diagnostic import het_breuschpagan

def check_assumptions(residuals,x_test):

    p_value = sm.stats.jarque_bera(residuals)[1]
    print(f"P-Value JB: {p_value}")

    dw_statistic = durbin_watson(residuals)
    print("Estadístico de Durbin-Watson:", dw_statistic)

    t_statistic, p_value = ttest_1samp(residuals, 0)
    print("p-valor de la prueba de t de Student:", p_value)

    lm, lm_p_value, fvalue, f_p_value = het_breuschpagan(residuals,
x_test)

    print("Estadístico Lagrange Multiplier:", lm)
    print("P-valor Lagrange Multiplier:", lm_p_value)
    print("Estadístico F:", fvalue)
    print("P-valor F:", f_p_value)

x_test = sm.add_constant(x_test)
check_assumptions(residuals_pearson, x_test)
```

```
P-Value JB: 7.409128336579679e-21
Estadístico de Durbin-Watson: 1.9987758614653832
p-valor de la prueba de t de Student: 0.00022698993146960363
Estadístico Lagrange Multiplier: 41.07346709163056
P-valor Lagrange Multiplier: 9.067538944157553e-08
Estadístico F: 8.521552122577495
P-valor F: 6.568541801982141e-08
```

```python
check_assumptions(residuals_spearman, x_test)
```

```
P-Value JB: 4.117349878729519e-08
Estadístico de Durbin-Watson: 2.0367298011748476
p-valor de la prueba de t de Student: 5.0535926719003954e-05
Estadístico Lagrange Multiplier: 74.44307631990522
P-valor Lagrange Multiplier: 1.2157042778168537e-14
Estadístico F: 16.013914628563406
P-valor F: 3.4453478204299243e-15
```

```python
check_assumptions(residuals_kendall, x_test)
```

```
P-Value JB: 5.714081641917175e-10
Estadístico de Durbin-Watson: 2.020947705644254
p-valor de la prueba de t de Student: 3.0593887236920502e-06
Estadístico Lagrange Multiplier: 15.367541877267108
P-valor Lagrange Multiplier: 0.008902083147962217
```

```
Estadístico F: 3.103355202786212
P-valor F: 0.008717423520814126
```

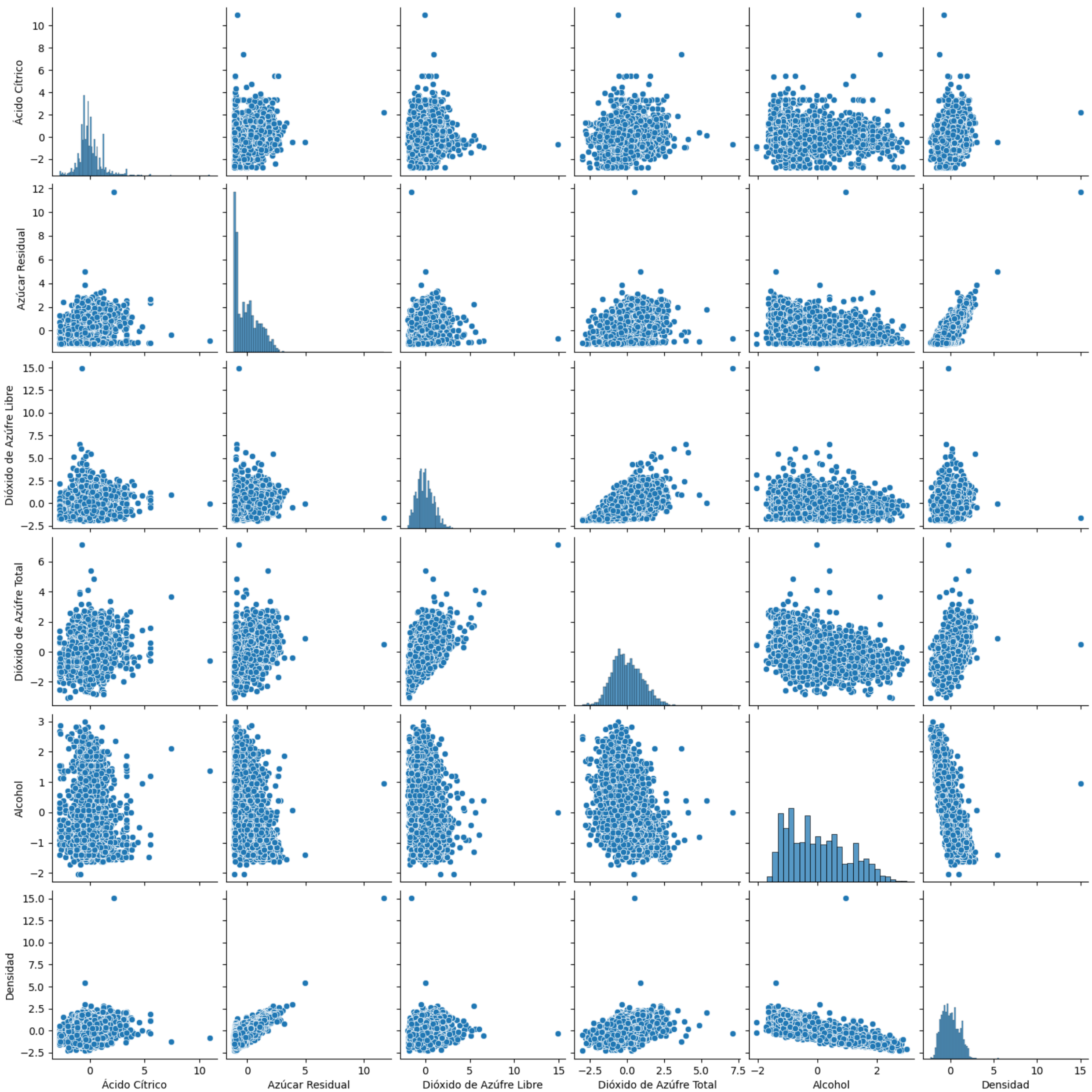
Supuestos para todos los modelos

- No presenta distribución normal en los residuos
- Existe independencia entre los datos
- No hay media cero
- No hay homocedasticidad
- Realice un análisis del diagrama de dispersión del conjunto de datos ¿Se evidencian comportamientos totalmente lineales? Si la respuesta es negativa, sugiera y realice transformaciones de variables (Ejemplo: exp(Xi), √Xi, log(Xi), X2i , 1Xi , etc.) y justifique el por qué de esa transformación. Finalmente, genere un modelo RLM e interprételo detalladamente.

Histograma de todo el dataset

```
import seaborn as sns

sns.pairplot(df_full)
```

```
<seaborn.axisgrid.PairGrid at 0x215cc3f7590>
```



Se observan comportamientos lineales en el conjunto de datos en relación a la variable de densidad

Generación de modelo RLM usando el conjunto de datos

```
x_design = x.copy()
x_design.insert(0, 'const', 1)

beta = np.matmul(np.linalg.inv(np.matmul(x_design.T, x_design)),
np.matmul(x_design.T, y))
beta
```

```
array([-2.21455778e-14,  5.03854925e-02,  5.96468430e-01, -
8.29905163e-02,
        1.23065533e-01, -4.73047804e-01])

y_pred = np.matmul(x_design, beta)

mean_squared_error(y, y_pred)

0.0821976076994994

check_assumptions(y - y_pred, x_design)

P-Value JB: 0.0
Estadístico de Durbin-Watson: 1.4786690138202532
p-valor de la prueba de t de Student: 1.0
Estadístico Lagrange Multiplier: 212.9166293491638
P-valor Lagrange Multiplier: 4.886290491750913e-44
Estadístico F: 44.46401775050657
P-valor F: 5.142185516338182e-45
```

Interpretación

- No se cumple ningun supuesto
- Se obtiene un MSE similar al encontrado usando las matrices de correlación, específicamente la de Kendall. Aproximadamente de 0,082.
- Se debe de tener en cuenta que los valores se encuentran estandarizados.

Punto 4

4) Se tiene un conjunto de datos que registra la cantidad de anuncios publicitarios en redes sociales que realiza una empresa y su correspondiente retorno de inversión en ventas. Se desea determinar si existe una relación lineal significativa entre la cantidad de anuncios publicitarios y el retorno de inversión.

- El conjunto de datos "publicidad.csv" consta de 200 observaciones y 4 variables que representan los gastos en publicidad (en miles de dólares) y las ventas (en miles de unidades) de un producto en un mercado específico: - TV: Gasto en publicidad en televisión. - Radio: Gasto en publicidad en radio. - Newspaper: Gasto en publicidad en periódicos. - Sales: Número de unidades vendidas (en miles)

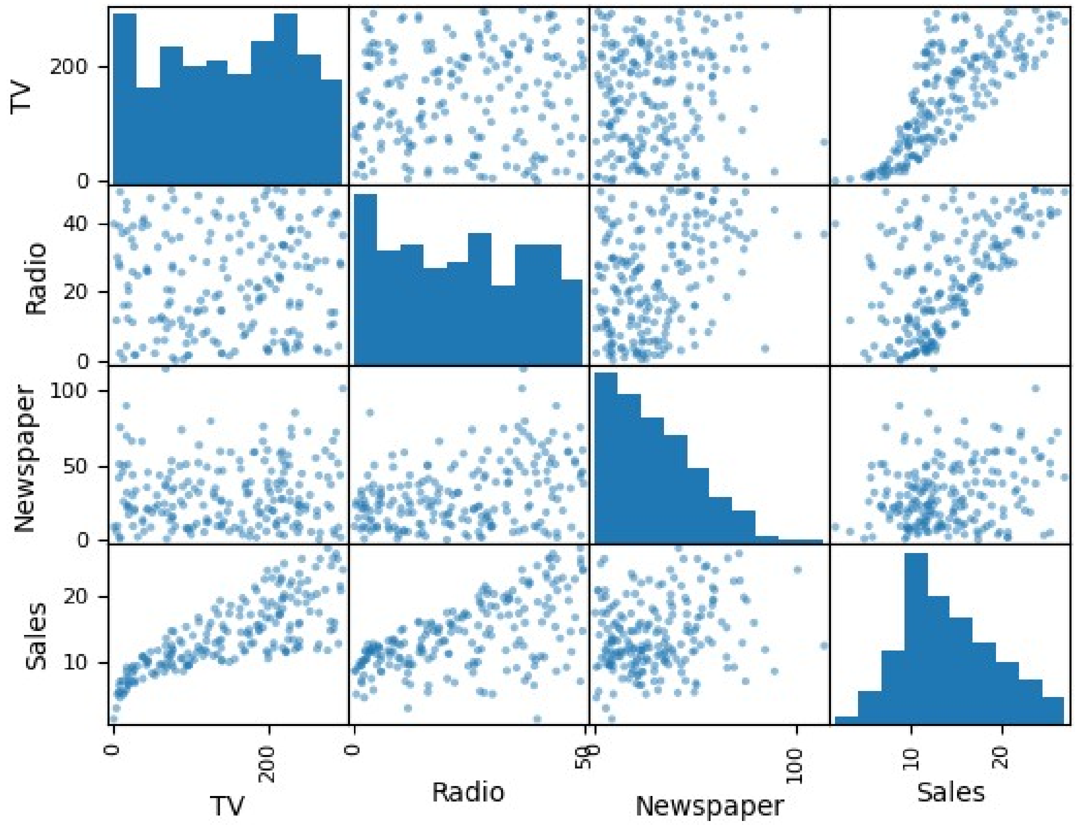
Carga de los datos

```
import pandas as pd

df = pd.read_csv('publicidad.csv', sep=',')
df = df.drop(columns=['Unnamed: 0'], axis=1)
df

        TV  Radio  Newspaper  Sales
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
3    151.5   41.3       58.5   18.5
4    180.8   10.8       58.4   12.9
..     ...    ...        ...    ...
195   38.2    3.7       13.8    7.6
196   94.2    4.9        8.1    9.7
197  177.0    9.3        6.4   12.8
198  283.6   42.0       66.2   25.5
199  232.1    8.6        8.7   13.4

[200 rows x 4 columns]
```

- Graficar el retorno de inversión (variable “Sales”) vs la cantidad de anuncios publicitarios por canal (“TV”, “Radio”, “Newspaper”). Para ello use la función scatter_matrix() del paquete pandas e interprete los graficos de las variables dos a dos, teniendo en cuenta que nuestra variable respuesta es “Sales”.

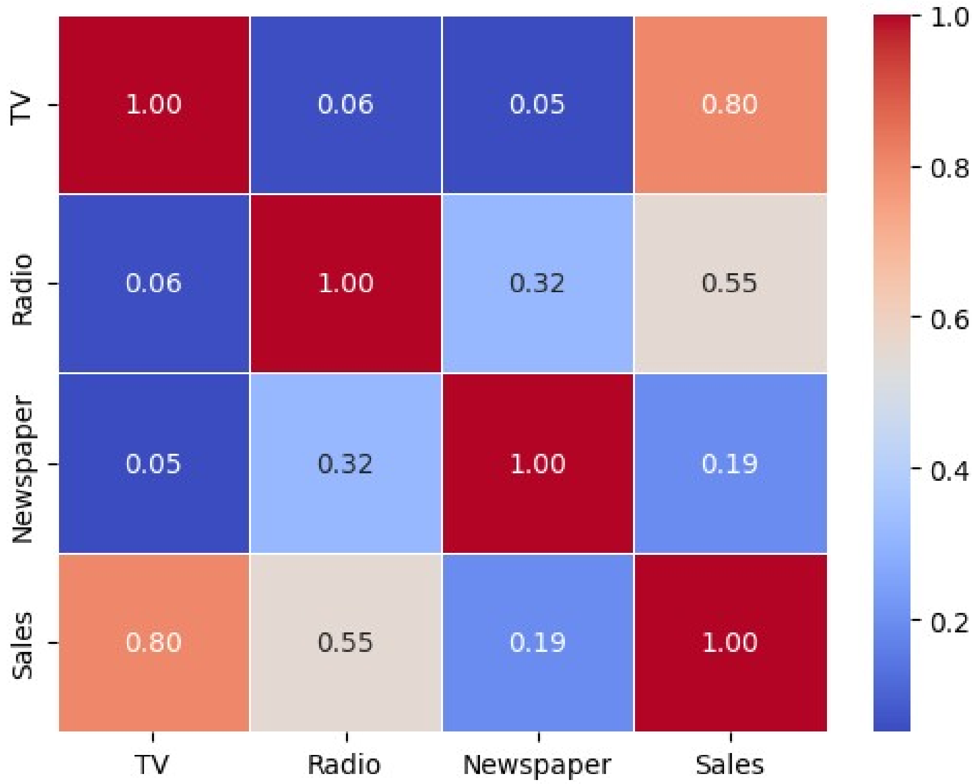
```
pd.plotting.scatter_matrix(df)

array([[<Axes: xlabel='TV', ylabel='TV'>,
        <Axes: xlabel='Radio', ylabel='TV'>,
        <Axes: xlabel='Newspaper', ylabel='TV'>,
        <Axes: xlabel='Sales', ylabel='TV'>],
       [<Axes: xlabel='TV', ylabel='Radio'>,
        <Axes: xlabel='Radio', ylabel='Radio'>,
        <Axes: xlabel='Newspaper', ylabel='Radio'>,
        <Axes: xlabel='Sales', ylabel='Radio'>],
       [<Axes: xlabel='TV', ylabel='Newspaper'>,
        <Axes: xlabel='Radio', ylabel='Newspaper'>,
        <Axes: xlabel='Newspaper', ylabel='Newspaper'>,
        <Axes: xlabel='Sales', ylabel='Newspaper'>],
       [<Axes: xlabel='TV', ylabel='Sales'>,
        <Axes: xlabel='Radio', ylabel='Sales'>,
        <Axes: xlabel='Newspaper', ylabel='Sales'>,
        <Axes: xlabel='Sales', ylabel='Sales'>]], dtype=object)
```



Interpretacion

- Se observa un comportamiento cuadratico entre la variable de TV y Sales
- El comportamiento entre Radio y Sales es similar al anterior, sin embargo tiene una mayor dispersion
- El comportamiento entre las variables de Newspaper y Sales tiene un comportamiento constante a lo largo de los valores horizontales
- Calcular el coeficiente de correlación entre todas las variables y mediante un mapa de calor represente estas correlaciones. ¿Interprete las estructuras de dependencia encontradas?

```
import seaborn as sns
import matplotlib.pyplot as plt

correlation = df.corr(method = 'spearman')

sns.heatmap(correlation, annot=True, cmap='coolwarm', fmt=".2f",
linewidths=.5)
plt.show()
```



Interpretacion

Se observa una correlacion media entre las variables de **TV-Sales** asi como **Radio-Sales**

- Teniedo en cuenta el punto anterior, elija solo una variable explicativa ("TV", "Radio", o "Newspaper"; la más conveniente) para modelar las ventas ("Sales"), ajuste el modelo de regresión lineal simple y encuentra la ecuación de la recta. ¿Cuál es el valor del coeficiente de determinación R2? ¿Cómo se interpreta este valor?

Seleccion de variable

Se escoge la variable de TV

```
mean_est = df[['TV', 'Sales']].cov().to_numpy()[1, 0] / df['TV'].var()
beta_est = df['Sales'].mean() - mean_est * df['TV'].mean()
print(f"Mean: {mean_est}, Beta: {beta_est}")

Mean: 0.04753664043301976, Beta: 7.032593549127694

from sklearn.metrics import r2_score

y_pred = mean_est * df['TV'] + beta_est
r2 = r2_score(df['Sales'], y_pred)
print(f"R2: {r2}")

R2: 0.611875050850071
```

- Alrededor del 61.19% de la variabilidad en los valores de la variable de respuesta puede ser explicada por las variables independientes incluidas en el modelo.
- El 38.81% restante de la variabilidad no está explicada por el modelo.
- Realiza una predicción del retorno de inversión esperado cuando se realizan 5 anuncios por el canal de la variable escogida en el ítem anterior. ¿Cuál es el intervalo de confianza del 95 % para la predicción?

Prediccion cuando se realizan 5 anuncios por el canal de TV

```
y_tv_5 = mean_est * 5 + beta_est
print(f"TV 5: {y_tv_5}")
```

```
TV 5: 7.270276751292792
```

```
import numpy as np

standard_error = np.sqrt(np.sum((df['Sales'] - y_pred) ** 2) /
(df.shape[0] - 2))
print(f"Standard error: {standard_error}")
```

```
Standard error: 3.258656368650463
```

```
from scipy.stats import t

t_value = t.ppf(0.975, df.shape[0] - 2)
print(f"T value: {t_value}")
```

```
T value: 1.9720174778338955
```

```
confidence_interval = t_value * standard_error
print(f"Confidence interval: {confidence_interval}")

lower_bound = y_tv_5 - confidence_interval
upper_bound = y_tv_5 + confidence_interval
print(f"Lower bound: {lower_bound}, Upper bound: {upper_bound}")
```

```
Confidence interval: 6.426127313233446
Lower bound: 0.844149438059346, Upper bound: 13.696404064526238
```

Punto 5

5) Se desea predecir la resistencia a la compresión del concreto (Concrete compressive strength) en función de diferentes variables predictoras como el cemento (Cement), la escoria (Slag), la ceniza volante (Fly ash), el agua (Water), el superplastificante (Superplasticizer), el agregado grueso (Coarse aggregate) y el agregado fino (Fine aggregate). Para ello se dispone de un conjunto de datos con 1030 observaciones. Se desea construir un modelo de regresión lineal múltiple para predecir la resistencia a la compresión del concreto en función de las variables predictoras.

- Cargar los datos del archivo “Concrete_Data.xls” y examinar las características del con- junto de datos.
- Realizar un análisis exploratorio de los datos para entender la relación entre las variables predictoras y la variable respuesta.

Carga de los datos

```
import pandas as pd

df = pd.read_excel('Concrete_Data.xls')
df
```

```
      Cement (component 1)(kg in a m^3 mixture)  \
0                                         540.0
1                                         540.0
2                                         332.5
3                                         332.5
4                                         198.6
...                                         ...
1025                                      276.4
1026                                      322.2
1027                                      148.5
1028                                      159.1
1029                                      260.9

      Blast Furnace Slag (component 2)(kg in a m^3 mixture)  \
0                                                   0.0
1                                                   0.0
2                                                 142.5
3                                                 142.5
4                                                 132.4
...                                                 ...
1025                                              116.0
1026                                                0.0
1027                                              139.4
1028                                              186.7
1029                                              100.5

      Fly Ash (component 3)(kg in a m^3 mixture)  \
0                                           0.0
1                                           0.0
2                                           0.0
3                                           0.0
4                                           0.0
...                                         ...
1025                                       90.3
1026                                      115.6
1027                                      108.6
1028                                        0.0
1029                                       78.3
```

```
      Water  (component 4)(kg in a m^3 mixture)  \
0                                          162.0
1                                          162.0
2                                          228.0
3                                          228.0
4                                          192.0
...                                          ...
1025                                       179.6
1026                                       196.0
1027                                       192.7
1028                                       175.6
1029                                       200.6

      Superplasticizer (component 5)(kg in a m^3 mixture)  \
0                                                   2.5
1                                                   2.5
2                                                   0.0
3                                                   0.0
4                                                   0.0
...                                                 ...
1025                                                8.9
1026                                               10.4
1027                                                6.1
1028                                               11.3
1029                                                8.6

      Coarse Aggregate  (component 6)(kg in a m^3 mixture)  \
0                                                1040.0
1                                                1055.0
2                                                 932.0
3                                                 932.0
4                                                 978.4
...                                                 ...
1025                                              870.1
1026                                              817.9
1027                                              892.4
1028                                              989.6
1029                                              864.5

      Fine Aggregate (component 7)(kg in a m^3 mixture)  Age (day)  \
0                                                 676.0         28
1                                                 676.0         28
2                                                 594.0        270
3                                                 594.0        365
4                                                 825.5        360
...                                                 ...        ...
1025                                              768.3         28
1026                                              813.4         28
1027                                              780.0         28
```

```
1028                                                         788.9          28
1029                                                         761.5          28

      Concrete compressive strength(MPa, megapascals)
0                                                     79.986111
1                                                     61.887366
2                                                     40.269535
3                                                     41.052780
4                                                     44.296075
...                                                         ...
1025                                                  44.284354
1026                                                  31.178794
1027                                                  23.696601
1028                                                  32.768036
1029                                                  32.401235

[1030 rows x 9 columns]
```

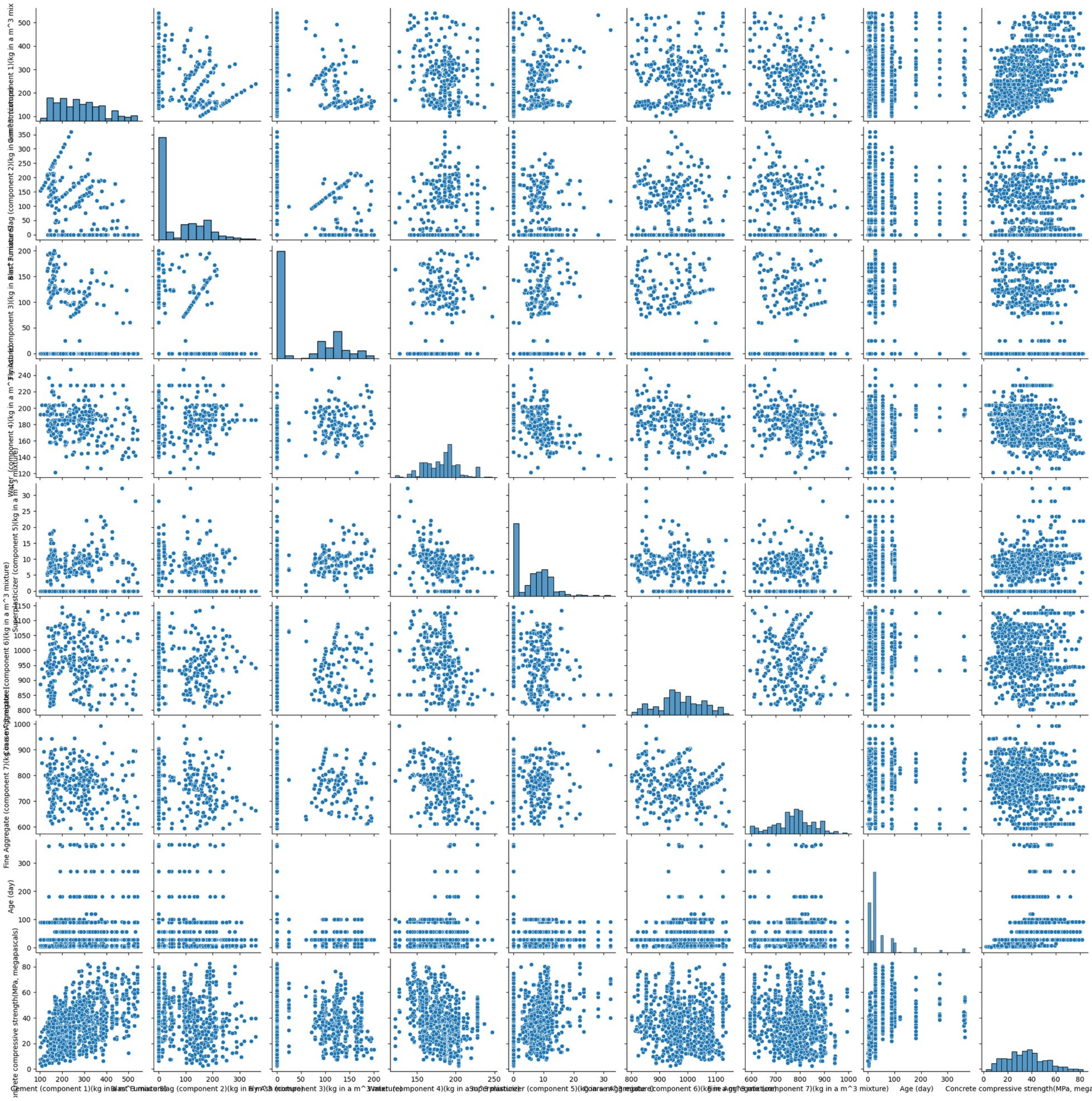
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1030 entries, 0 to 1029
Data columns (total 9 columns):
 #   Column                                                 Non-Null
Count  Dtype
---  ------
--------------  -----
 0   Cement (component 1)(kg in a m^3 mixture)              1030 non-
null   float64
 1   Blast Furnace Slag (component 2)(kg in a m^3 mixture)  1030 non-
null   float64
 2   Fly Ash (component 3)(kg in a m^3 mixture)             1030 non-
null   float64
 3   Water  (component 4)(kg in a m^3 mixture)              1030 non-
null   float64
 4   Superplasticizer (component 5)(kg in a m^3 mixture)    1030 non-
null   float64
 5   Coarse Aggregate  (component 6)(kg in a m^3 mixture)   1030 non-
null   float64
 6   Fine Aggregate (component 7)(kg in a m^3 mixture)      1030 non-
null   float64
 7   Age (day)                                              1030 non-
null   int64
 8   Concrete compressive strength(MPa, megapascals)        1030 non-
null   float64
dtypes: float64(8), int64(1)
memory usage: 72.6 KB
```

```
import seaborn as sns
import matplotlib.pyplot as plt
```

```
sns.pairplot(df)
```

```
<seaborn.axisgrid.PairGrid at 0x215cd3f9890>
```



```
df.describe()
```

```
       Cement (component 1)(kg in a m^3 mixture)  \
count                                  1030.000000
mean                                    281.165631
std                                     104.507142
min                                     102.000000
```

```
25%                                           192.375000
50%                                           272.900000
75%                                           350.000000
max                                           540.000000

       Blast Furnace Slag (component 2)(kg in a m^3 mixture)  \
count                                         1030.000000
mean                                            73.895485
std                                             86.279104
min                                              0.000000
25%                                              0.000000
50%                                             22.000000
75%                                            142.950000
max                                            359.400000

       Fly Ash (component 3)(kg in a m^3 mixture)  \
count                                  1030.000000
mean                                     54.187136
std                                      63.996469
min                                       0.000000
25%                                       0.000000
50%                                       0.000000
75%                                     118.270000
max                                     200.100000

       Water  (component 4)(kg in a m^3 mixture)  \
count                                 1030.000000
mean                                   181.566359
std                                     21.355567
min                                    121.750000
25%                                    164.900000
50%                                    185.000000
75%                                    192.000000
max                                    247.000000

       Superplasticizer (component 5)(kg in a m^3 mixture)  \
count                                         1030.000000
mean                                             6.203112
std                                              5.973492
min                                              0.000000
25%                                              0.000000
50%                                              6.350000
75%                                             10.160000
max                                             32.200000

       Coarse Aggregate  (component 6)(kg in a m^3 mixture)  \
count                                         1030.000000
mean                                           972.918592
std                                             77.753818
min                                            801.000000
```

```
25%                                                      932.000000
50%                                                      968.000000
75%                                                     1029.400000
max                                                     1145.000000

       Fine Aggregate (component 7)(kg in a m^3 mixture)    Age (day)
\
count                                                   1030.000000  1030.000000

mean                                                     773.578883    45.662136

std                                                       80.175427    63.169912

min                                                      594.000000     1.000000

25%                                                      730.950000     7.000000

50%                                                      779.510000    28.000000

75%                                                      824.000000    56.000000

max                                                      992.600000   365.000000


       Concrete compressive strength(MPa, megapascals) 
count                                        1030.000000
mean                                           35.817836
std                                            16.705679
min                                             2.331808
25%                                            23.707115
50%                                            34.442774
75%                                            46.136287
max                                            82.599225
```

```
df.columns
```

```
Index(['Cement (component 1)(kg in a m^3 mixture)',
       'Blast Furnace Slag (component 2)(kg in a m^3 mixture)',
       'Fly Ash (component 3)(kg in a m^3 mixture)',
       'Water  (component 4)(kg in a m^3 mixture)',
       'Superplasticizer (component 5)(kg in a m^3 mixture)',
       'Coarse Aggregate  (component 6)(kg in a m^3 mixture)',
       'Fine Aggregate (component 7)(kg in a m^3 mixture)', 'Age
(day)',
       'Concrete compressive strength(MPa, megapascals) '],
      dtype='object')
```

- Entrenar un modelo de regresión lineal múltiple utilizando el conjunto de datos y evalue si hay significancia en el modelo.

Modelo RLM

```
import statsmodels.api as sm

X = sm.add_constant(df.drop("Concrete compressive strength(MPa,
megapascals) ", axis=1))
y = df["Concrete compressive strength(MPa, megapascals) "]

model_full = sm.OLS(y, X).fit()
model_full.summary()
```

```
<class 'statsmodels.iolib.summary.Summary'>
"""
                                                     OLS Regression Results

==============================================================================
======================================
Dep. Variable:     Concrete compressive strength(MPa, megapascals)
R-squared:                            0.615
Model:                                                                    OLS
Adj. R-squared:                       0.612
Method:                                                         Least Squares
F-statistic:                          204.3
Date:                                                        Sat, 27 Apr 2024
Prob (F-statistic):               6.76e-206
Time:                                                                12:56:16
Log-Likelihood:                     -3869.0
No. Observations:                                                        1030
AIC:                                  7756.
Df Residuals:                                                            1021
BIC:                                  7800.
Df Model:                                                                   8

Covariance Type:                                                    nonrobust

==============================================================================
===================================================
                                                                  coef
std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
---------------------------------------------------
const                                                         -23.1638
26.588     -0.871      0.384     -75.338      29.010
Cement (component 1)(kg in a m^3 mixture)                       0.1198
0.008     14.110      0.000       0.103       0.136
Blast Furnace Slag (component 2)(kg in a m^3 mixture)     0.1038
0.010     10.245      0.000       0.084       0.124
Fly Ash (component 3)(kg in a m^3 mixture)                      0.0879
0.013      6.988      0.000       0.063       0.113
Water  (component 4)(kg in a m^3 mixture)                      -0.1503
0.040     -3.741      0.000      -0.229      -0.071
```

```
Superplasticizer (component 5)(kg in a m^3 mixture)       0.2907
0.093      3.110      0.002       0.107       0.474
Coarse Aggregate  (component 6)(kg in a m^3 mixture)      0.0180
0.009      1.919      0.055      -0.000       0.036
Fine Aggregate (component 7)(kg in a m^3 mixture)         0.0202
0.011      1.883      0.060      -0.001       0.041
Age (day)                                                 0.1142
0.005     21.046      0.000       0.104       0.125
======================================================================
========
Omnibus:                        5.379   Durbin-Watson:
1.281
Prob(Omnibus):                  0.068   Jarque-Bera (JB):
5.305
Skew:                          -0.174   Prob(JB):
0.0705
Kurtosis:                       3.045   Cond. No.
1.06e+05
======================================================================
========

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is
correctly specified.
[2] The condition number is large, 1.06e+05. This might indicate that
there are
strong multicollinearity or other numerical problems.
"""
```

- Se observa que el modelo presenta una significancia segun el p-valor del estadistico F, el cual es de 6.76e-206
- Las variables predictoras con un p-valor menor a 0.05 se consideran significativas, por lo tanto los siguientes atributos (p >= 0.05) resultan poco significantes para el modelo:
  - Coarse Aggregate (component 6)(kg in a m^3 mixture)
  - Fine Aggregate (component 7)(kg in a m^3 mixture)
- Analizar la significancia estadística de las variables predictoras y construir un modelo de regresión lineal múltiple reducido con las variables significativas. Revise su desempeño con respecto al modelo completo revisando el Adj −R2 y los criterios de información de Akaike y de Bayes (AIC y BIC).

Modelo usando las variables significativas

```
columns_to_drop = ["Concrete compressive strength(MPa, megapascals) ",
                "Coarse Aggregate  (component 6)(kg in a m^3
mixture)",
                "Fine Aggregate (component 7)(kg in a m^3 mixture)"]
X = sm.add_constant(df.drop(columns_to_drop, axis=1))
y = df["Concrete compressive strength(MPa, megapascals) "]
```

```
model_min = sm.OLS(y, X).fit()
model_min.summary()
```

```
<class 'statsmodels.iolib.summary.Summary'>
"""
                                           OLS Regression Results

======================================================================
======================================
Dep. Variable:     Concrete compressive strength(MPa, megapascals)
R-squared:                       0.614
Model:                                                          OLS
Adj. R-squared:                  0.612
Method:                                               Least Squares
F-statistic:                     271.2
Date:                                              Sat, 27 Apr 2024
Prob (F-statistic):          1.78e-207
Time:                                                      12:56:16
Log-Likelihood:                -3871.0
No. Observations:                                              1030
AIC:                             7756.
Df Residuals:                                                  1023
BIC:                             7791.
Df Model:                                                         6

Covariance Type:                                          nonrobust

======================================================================
===================================================
                                                             coef
std err          t      P>|t|      [0.025      0.975]
----------------------------------------------------------------------
---------------------------------------------------
const                                                     29.0302
4.212      6.891      0.000      20.764      37.296
Cement (component 1)(kg in a m^3 mixture)                  0.1054
0.004     24.821      0.000       0.097       0.114
Blast Furnace Slag (component 2)(kg in a m^3 mixture)     0.0865
0.005     17.386      0.000       0.077       0.096
Fly Ash (component 3)(kg in a m^3 mixture)                 0.0687
0.008      8.881      0.000       0.054       0.084
Water  (component 4)(kg in a m^3 mixture)                 -0.2183
0.021    -10.332      0.000      -0.260      -0.177
Superplasticizer (component 5)(kg in a m^3 mixture)        0.2390
0.085      2.826      0.005       0.073       0.405
Age (day)                                                  0.1135
0.005     20.987      0.000       0.103       0.124
======================================================================
========
Omnibus:                        5.233   Durbin-Watson:
```

```
1.286
Prob(Omnibus):                  0.073   Jarque-Bera (JB):
5.193
Skew:                          -0.174   Prob(JB):
0.0745
Kurtosis:                       3.019   Cond. No.
4.66e+03
======================================================================
========

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is
correctly specified.
[2] The condition number is large, 4.66e+03. This might indicate that
there are
strong multicollinearity or other numerical problems.
"""
```

```
print("Adj-R2 Full Model: ", model_full.rsquared_adj)
print("Adj-R2 Min Model: ", model_min.rsquared_adj)
```

```
Adj-R2 Full Model:  0.6124517253305723
Adj-R2 Min Model:  0.6117109257365057
```

```
print(f"AIC Full Model: {model_full.aic} BIC Full Model:
{model_full.bic}")
print(f"AIC Min Model: {model_min.aic} BIC Min Model:
{model_min.bic}")
```

```
AIC Full Model: 7756.063911394405 BIC Full Model: 7800.499738125419
AIC Min Model: 7756.046536116364 BIC Min Model: 7790.60773468493
```

- Comparando los resultados de los dos modelos, observamos que el modelo completo (Full Model) y el modelo reducido (Min Model) tienen coeficientes de determinación ajustados (R2 ajustados) muy similares, con valores de 0.612 y 0.612, respectivamente. Esto sugiere que ambos modelos explican aproximadamente la misma cantidad de variabilidad en la variable dependiente.

- En cuanto a los criterios de información, tanto el AIC como el BIC son ligeramente más bajos en el modelo reducido en comparación con el modelo completo. Esto indica que el modelo reducido tiene un mejor ajuste relativo a los datos y a la complejidad del modelo, ya que penaliza la inclusión de variables adicionales.

- Valide los supuestos del modelo (εi iid$\sim$ N(0,σ2)) y en caso de no cumplir alguno, proponga una solución. Evalúe la conveniencia de usar un enfoque robusto en este caso.

Validacion de supuestos

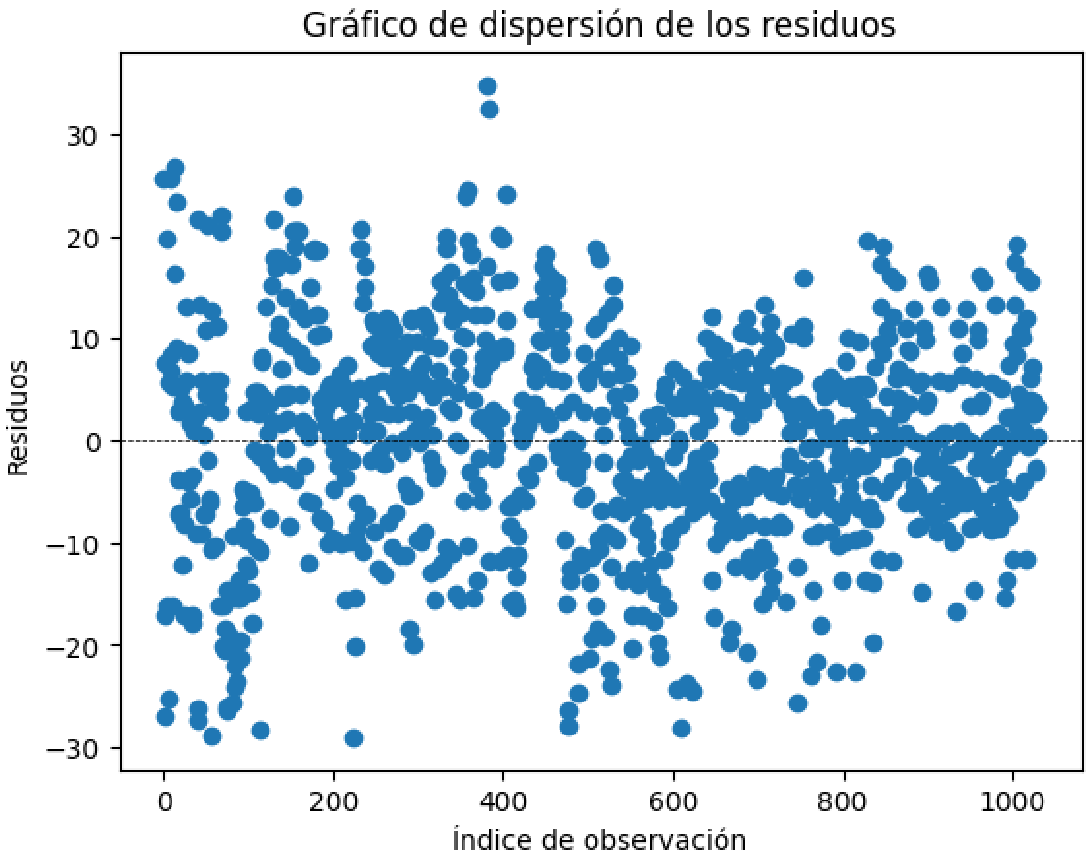
```
from statsmodels.stats.stattools import durbin_watson
from statsmodels.stats.diagnostic import het_breuschpagan
```

```
import statsmodels.api as sm
from scipy.stats import ttest_1samp

def check_assumptions(residuals, model):

    p_value = sm.stats.jarque_bera(residuals)[1]
    print(f"P-Value JB: {p_value}")

    dw_statistic = durbin_watson(residuals)
    print("Estadístico de Durbin-Watson:", dw_statistic)

    t_statistic, p_value = ttest_1samp(residuals, 0)
    print("p-valor de la prueba de t de Student:", p_value)

    lm, lm_p_value, fvalue, f_p_value = het_breuschpagan(residuals,
model.model.exog)

    print("Estadístico Lagrange Multiplier:", lm)
    print("P-valor Lagrange Multiplier:", lm_p_value)
    print("Estadístico F:", fvalue)
    print("P-valor F:", f_p_value)

residuals = model_min.resid
y_pred = model_min.predict(X)

check_assumptions(residuals, model_min)
```

```
P-Value JB: 0.07452558338743674
Estadístico de Durbin-Watson: 1.2859452611038251
p-valor de la prueba de t de Student: 0.9999999999996787
Estadístico Lagrange Multiplier: 139.18162251188713
P-valor Lagrange Multiplier: 1.4915855798332046e-27
Estadístico F: 26.638950472924435
P-valor F: 1.3882212736892169e-29
```

- Se cumple normalidad
- No se cumple independencia
- Se cumple media cero
- No se cumple homocedasticidad

```
plt.scatter(range(len(residuals)), residuals)
plt.axhline(0, color='black', linestyle='--', linewidth=0.5)
plt.xlabel('Índice de observación')
plt.ylabel('Residuos')
plt.title('Gráfico de dispersión de los residuos')
plt.show()
```




```
from sklearn.metrics import mean_absolute_error

mae = mean_absolute_error(y, y_pred)
print("mae:", mae)

mae: 8.261463918842809
```

Evaluacion de la conveniencia de usar un enfoque robusto

Se debería considerar el uso de un modelo robusto cuando se sospecha que los supuestos del modelo de regresión clásico no se cumplen completamente debido a la presencia de datos atípicos, heterocedasticidad o distribuciones no normales de los errores. Los modelos robustos pueden proporcionar estimaciones más confiables y resistentes a la presencia de estas violaciones de los supuestos.

Propuesta de solucion: Emplear un modelo robusto

```
robust_model = sm.RLM(y, X).fit()
robust_model.summary()

<class 'statsmodels.iolib.summary.Summary'>
"""
                                   Robust linear Model Regression
```

```
Results
==============================================================================
======================================
Dep. Variable:     Concrete compressive strength(MPa, megapascals)
No. Observations:                 1030
Model:                                                             RLM
Df Residuals:                     1023
Method:                                                           IRLS
Df Model:                            6
Norm:                                                           HuberT

Scale Est.:                                                        mad

Cov Type:                                                           H1

Date:                                                 Sat, 27 Apr 2024

Time:                                                         12:56:16

No. Iterations:                                                     20

==============================================================================
===================================================
                                                                 coef
std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
---------------------------------------------------
const                                                         28.8922
4.165      6.937      0.000      20.729      37.056
Cement (component 1)(kg in a m^3 mixture)                      0.1078
0.004     25.671      0.000       0.100       0.116
Blast Furnace Slag (component 2)(kg in a m^3 mixture)          0.0890
0.005     18.099      0.000       0.079       0.099
Fly Ash (component 3)(kg in a m^3 mixture)                     0.0724
0.008      9.464      0.000       0.057       0.087
Water  (component 4)(kg in a m^3 mixture)                     -0.2258
0.021    -10.808      0.000      -0.267      -0.185
Superplasticizer (component 5)(kg in a m^3 mixture)            0.2267
0.084      2.711      0.007       0.063       0.391
Age (day)                                                      0.1320
0.005     24.686      0.000       0.122       0.142
==============================================================================
===================================================

If the model instance has been used for another fit with different fit
parameters, then the fit options might not be the correct ones anymore
.
"""
```

Punto 6

Short Communication: Roubst Locally Weigth Regression
Basado en: (Cleveland, 1979)

Alejandro Martínez, Kevin Rodríguez, Luis Vásquez, Jhonatan Valencia

Introducción

Los diagramas de dispersión o scatterplots muestran la relación entre dos variables cuantitativas, lo cual los convierte en herramientas visuales fundamentales en el análisis de datos. Cada punto en el scatterplot representa una observación individual, donde una variable, generalmente la explicativa, se representa en el eje horizontal (eje x) y la otra, la respuesta, en el eje vertical (eje y) (Grima Cintas, Marco Almagro , & Tort-Martorell , 2012).

Debido a la variabilidad aleatoria de los datos, puede ser difícil identificar patrones o relaciones entre las variables. Por ejemplo, en la figura 1 se presentan datos generados de forma sintética usando $y_i = 0.02x_i + \epsilon_i$, ($\epsilon_i$ representa el error aleatorio con distribución normal). Al fijarse en los datos, no resulta para nada evidente que la función que mejor describe la relación entre los datos es una recta. De esta manera, se evidencia la necesidad de contar con técnicas de suavizado para el análisis de datos. Adicionalmente, el suavizado de scatterplots no solo es una herramienta de exploración de datos muy útil, sino que un ingrediente esencial en muchas técnicas de ajuste para datos (Härdle & Marron, 1995).

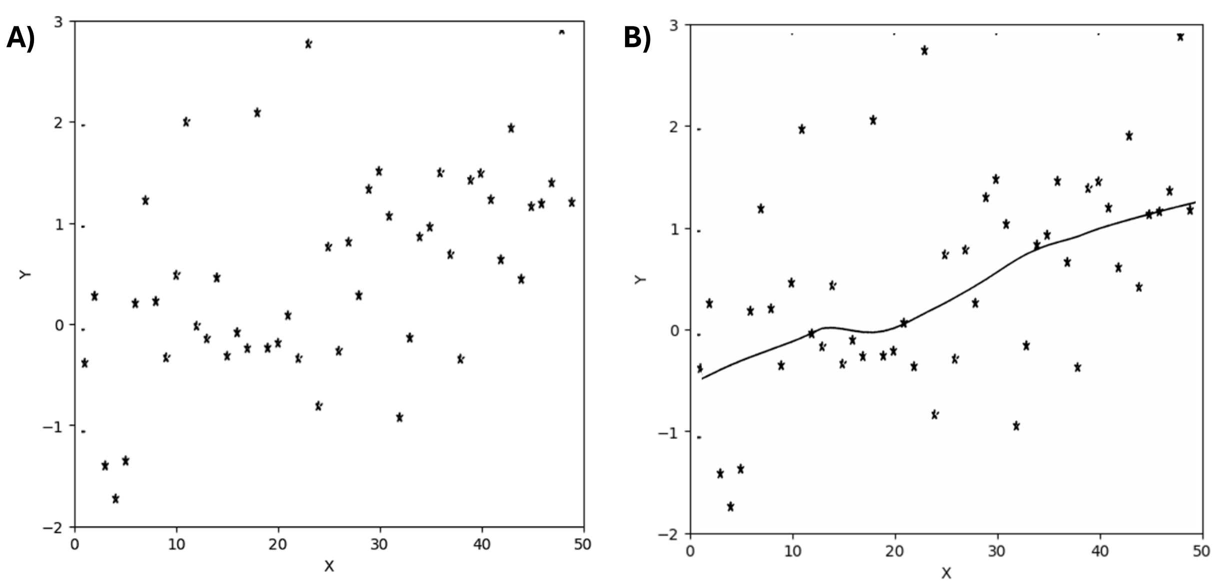


Figura 1. A) *Scatterplot generado de forma sintética.* ***B)*** *Scatterplot A suavizado. Adaptado de (Cleveland, 1979)*

Teniendo en cuenta lo anterior, numerosos autores han propuesto métodos para suavizar scatterplots. Algunos de estos métodos son simples como el propuesto por Ezeiel, 1941, basado en la media de los datos, mientras que otros métodos son más robustos. Por ejemplo, Clark, 1977, propone unir puntos sucesivos con líneas rectas mediante interpolación y luego suavizar mediante convolución con una función de peso. Otro ejemplo son las estimaciones de regresión no paramétrica, que Clark, 1977, incluyó en su revisión de bibliografía (Cleveland, 1979).

A pesar de que este tipo de métodos, como los últimos mencionados, pueden resultar precisos, existe una necesidad por métodos rápidos y sencillos que conserven su robustez para suavizar scatterplots. Esto se debe a que la sencillez es deseable para su integración con otros métodos y para facilitar su implementación en algoritmos, lo que permite una ejecución rápida y eficiente (Härdle & Marron, 1995). En este contexto, el profesor William S. Cleveland discute algunos aspectos estadísticos y presenta una guía para aplicar el método conocido como Roubst Locally Weigth Regression. El cual es un método sencillo y computacionalmente económico para obtener más información visual de scatterplots.

Sobre el método Roubst Locally Weigth Regression

Roubst Locally Weigth Regression (RLWR) o Regresión Ponderada Local Robusta es un método de suavizado para scatterplots o gráficos de dispersión. Este es una combinación de técnicas como Locally Weigth Regression, que resulta de una extensión de la técnica de Local Fitting comúnmente usada para suavizar series de tiempo y de un procedimiento robusto que resulta de adaptar la técnica conocida como Iterative Weighted Least Square.

De forma general, RLWR genera nuevos puntos suavizados (x_k, y_k) en un gráfico de dispersión (x_i, y_i), donde el valor ajustado y_k es el valor que resulta de evaluar x_k en un polinomio ajustado a los datos usando mínimos cuadrados ponderados. Estos pesos se asignan según la proximidad de los puntos x_i a un valor específico x_k; si x_i está cerca de x_k, el peso asignado a esa observación es grande, lo que indica que tiene más influencia en el proceso de estimación.

La figura 2, muestra un esquema para aplicar el método RLWR. En este esquema se resalta que el método consiste esencialmente de dos etapas: en la primera se realizan estimaciones iniciales, usando la técnica Locally Weigth Regression. En la segunda, el método adquiere la característica “Robust” al realizar un proceso iterativo de definición de nuevos pesos, denominados pesos de robustez, y realizar ajuste usando Weighted Least Squares.

A lo largo de la metodología descrita, el usuario debe elegir el valor de 4 parámetros: d, que representa el orden del polinomio que se ajusta localmente a cada punto en el scatterplot. W, que se refiere a la función utilizada para determinar los pesos asignados a cada punto en el proceso de ajuste. t, el cual indica el número de iteraciones del procedimiento de ajuste robusto. Y el parámetro f, que permite determinar qué tan suavizado quedará el resultado, pues al aumentar f, aumenta el neighborhood de puntos que influencian el resultado, y por lo tanto, tiende a aumentar la suavidad de los puntos estimados.

Discusión y Análisis del método

Roubst Locally Weigth Regression no tiene en cuenta de forma directa el valor de cada observación, como lo hace el método propuesto por Ezeiel, 1941, por el contrario, tiene en cuenta la distribución de estas observaciones. Esta característica representa una ventaja, pues el efecto los outliers o datos atípicos atípicos es menor, comparado con los métodos que sólo tienen en cuenta el valor de las observaciones. Adicionalmente, el carácter iterativo y la dinámica de asignar pesos según la posición permiten reducir más el efecto de los outliers sobre la estimación de los puntos suavizados, pues los outliers tienden a tener menores pesos debido a su posición.

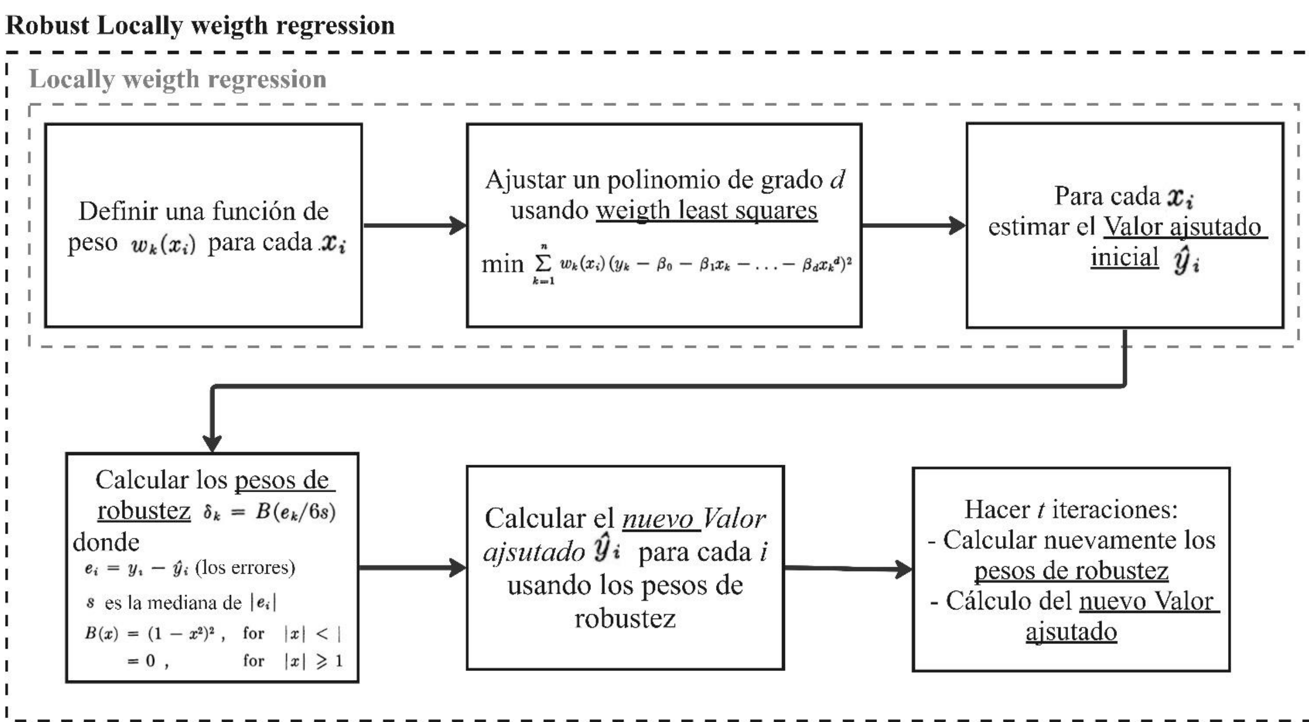


Figura 2. *Esquema para suavizar un scatterplot usando Roubst Locally Weigth Regression*

Conforme se consideran puntos x_i más distantes para la regresión, es decir se recopila más información de los datos para cada punto, se reduce la varianza e incrementa el bias. La reducción de la varianza se da porque se tienen en cuenta más datos para hacer la predicción, de forma que se reducen las fluctuaciones, obteniendo un mayor suavizado. Sin embargo, esto puede llevar a incluir valores lejanos al punto de interés, lo cual puede aumentar el bias debido a que se están teniendo en cuenta valores que pueden no ser representativos del punto x_k de interés.

Por lo discutido anteriormente, se prefiere el método Roubst Locally Weigth Regression por encima de otros debido a su sencillez, adaptabilidad (mediante modificación de los parámetros), robustez e interpretabilidad. Este permite capturar relaciones no lineales entre variables y ajustarse a diferentes niveles de ruido y distribución de datos, siendo robusto frente a valores atípicos. La adaptabilidad permite que el usuario pueda ajustar el suavizado según las necesidades del análisis y conservar la interpretabilidad.

Ejemplo: aplicando RLWR a un caso de Ingeniería en Bioprocesos

El artículo del profesor Cleveland contiene una guía para aplicar RLWR y seleccionar adecuadamente los 4 parámetros. Con base en esta guía, se utilizó RLWR para obtener más información visual de un scatterplot que relaciona la concentración de la fuente de carbono (mM) y titer o concentración final de producto (g/L) de fermentaciones realizadas con *E. coli*. Para entrar en contexto, cabe mencionar que la bacteria *E. coli* es conocida como una fábrica microbiana debido a su versatilidad en la producción de proteínas recombinantes, enzimas y productos farmacéuticos. Su rápido crecimiento y facilidad de manipulación genética la convierten en un microorganismo modelo en investigación y le otorga protagonismo en el sector industrial.

Al igual que otros procesos biológicos, las fermentaciones con *E. coli* son procesos dinámicos y complejos. La calidad y productividad son resultado conjunto de la intrincada red metabólica y las condiciones de operación, por lo que la optimización de este tipo de procesos representa un reto. En los últimos años, se ha evidenciado una tendencia en la adopción de soluciones digitales por parte de la industria biotecnológica. Esto debido a que las herramientas digitales y basadas en datos, como el machine learning, presentan gran potencial para abordar este tipo de retos en las que el modelamiento fenomenológico se queda corto para considerar la gran cantidad de variables que afectan el proceso.

Teniendo en cuenta esta tendencia en la industria biotecnológica y la relevancia de las fermentaciones de *E. coli* en esta misma industria, Oyetunde, Liu, Garcia y Tang propusieron un enfoque para integrar métodos basados en datos con un modelo metabólico (Oyetunde, Liu, Garcia Martin, & Tang, 2019). Para esto, construyeron una base de datos, de la cual se tomaron y procesaron los datos referentes a la concentración de la fuente de carbono (mM) y titer o concentración final de producto (g/L) de fermentaciones realizadas con *E. coli*.

La figura 3 A es un scatterplot de los datos de las variables de interés. Estos datos están previamente tratados, se realizó imputación de valores nulos y eliminación de algunos atípicos extremos que podrían afectar la interpretación general. Vale la pena mencionar, que las observaciones se realizaron bajo distintas configuraciones y condiciones de operación. De forma que la concentración de la fuente de carbono y titer se seleccionaron como variables de interés buscando identificar fenómenos como a la represión metabólica, inhibición por sustrato, y no pensando en obtener un modelo para predicción. De hecho, no sería recomendable tener en cuenta únicamente o considerar la concentración de sustrato como un único predictor.

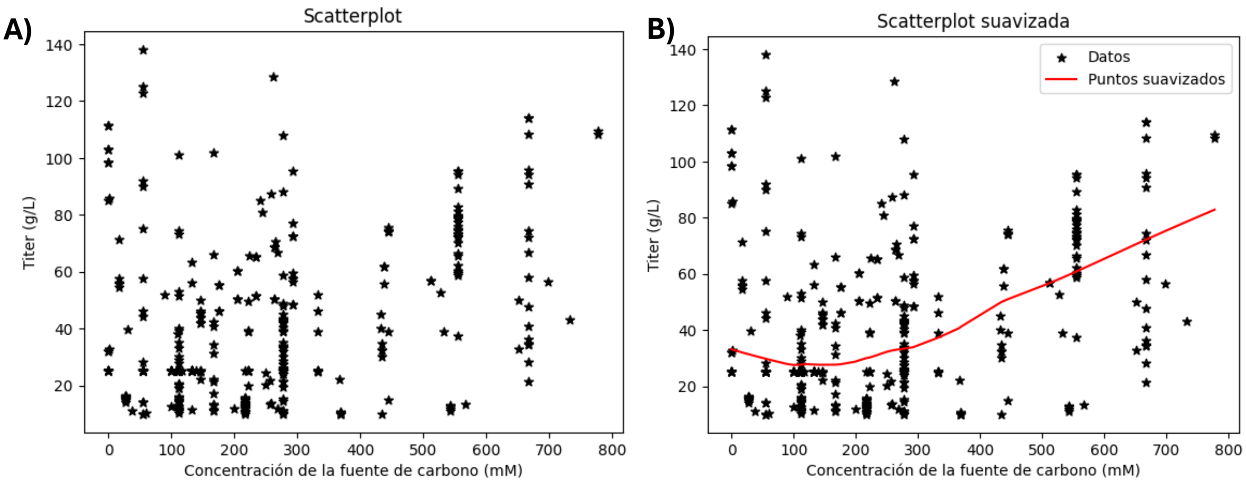


Figura 3. A) *Scatterplot con datos de concentración de sustrato y titer. Tomado de (Oyetunde, Liu, Garcia Martin, & Tang, 2019).* ***B)*** *Scatterplot A suavizado.*

Se utilizó el módulo de Python statsmodel, el cual está basado en el mismo trabajo del profesor Cleveland que se analiza en el presente escrito (Seabold, Skipper, & Perktold, 2024). De acuerdo con la guía elaborada por Cleveland, se seleccionó $t = 2$ y $f = 0.55$. Este valor de f se estableció tras varias ejecuciones del método, este valor permite reducir la variabilidad comparado con valores menores de f, y permite evidenciar mejor la forma de la distribución de los datos comparado con valores mayores de f. En el caso de w, la statsmodel utiliza una función de peso tricubo, la cual debería ser adecuada para casi todas las situaciones según Cleveland. La documentación no especifica cual es el valor de d, pero se esperaría que este tenga valor de 1, ya que ofrece un balance entre buen ajuste y esfuerzo computacional. Con esta configuración, se obtuvieron los puntos suavizados que se grafican en la figura 3 B.

El propósito de este suavizado fue el de mejorar la percepción visual de los patrones en el scatterplot. El resultado obtenido sugiere que la concentración final de producto tiende aumentar en configuraciones en las que se utiliza mayor concentración de sustrato. Este hallazgo resulta interesante, puesto que en fermentaciones con levaduras crabtree-positivas se obtiene una relación distinta cuando el producto es biomasa; conforme aumenta la concentración de sustrato se obtiene una menor concentración del producto de interés debido al fenómeno de desbordamiento del metabolismo oxidativo.

Estos resultados sugieren que las configuraciones de reactor utilizadas, condiciones definidas y cepas de *E. coli* empleadas dan lugar a un proceso con selectividad alta. Es decir, que gran parte del sustrato es aprovechado. No obstante, cabe resaltar que las pendientes de las rectas que unen los puntos suavizados no son muy pronunciadas. Lo que demuestra que aumentar la concentración final del producto implica un mayor costo, ya que se necesita proporcionalmente más sustrato.

Visualización

Finalmente, el ejemplo anteriormente planteado es utilizado en esta sección para mostrar las distintas formas de visualizar los scatterplots suavizados según Cleveland.

La primera forma es la presentada en la figura 3B, en la que los puntos suavizados se unen mediante rectas. Esta es una buena alternativa para comparar los puntos suavizados con los datos en el mismo scatterplot. Además, permite evidenciar de mejor manera las tendencias según los autores del presente trabajo, por esta razón fue el método de visualización usado en el ejemplo. Sin embargo, a pesar de ser una buena forma de visualizar, usar este método puede llevar a la falsa idea de que se está realizando una interpolación entre los puntos suavizados, lo cual resultaría inadecuado.

Otro método para visualizar consiste en no unir los puntos suavizados con líneas, de esta forma evitando la idea de que se realizó interpolación. Este método se muestra en la figura 4 A y resulta útil cuando las líneas continuas pueden llevar a la distorsión de los resultados. Finalmente, el ultimo método sugerido por Cleveland consiste en graficar los puntos suavizados en otro scatterplot usando la misma escala del original como se muestra en la figura 4 B. Este resulta útil para ocasiones donde se tiene baja resolución de los gráficos.

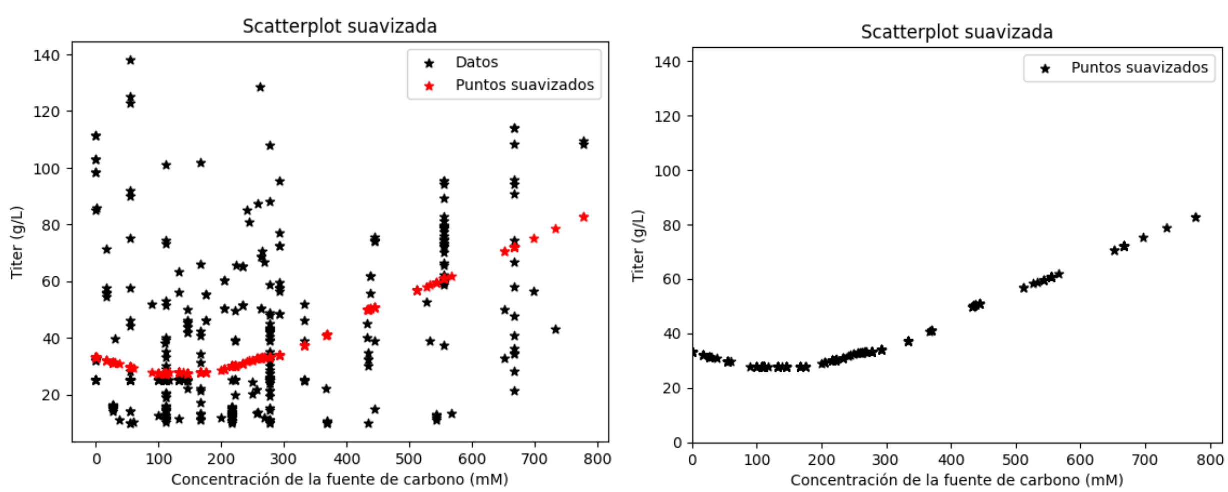


Figura 4. A) *Scatterplot suavizado sin líneas.* ***B)*** *Scatterplot de los puntos de suavizado.*

Referencias

Cleveland, W. (1979). Robust Locally Weighted Regression and Smoothing Scatterplots. Journal of the American Statistical Association. https://doi.org/10.1080/01621459.1979.10481038, 829-836.

Grima Cintas, P., Marco Almagro , L., & Tort-Martorell , X. (2012). Industrial Statics with Minitab. Wiley.

Härdle, W., & Marron, J. (1995). Fast and simple scatterplot smoothing. Computational Statistics & Data Analysis, Volumen 20, 1-17.

Oyetunde, T., Liu, D., Garcia Martin, H., & Tang, Y. (2019). Machine learning framework for assessment of microbial factory performance. PLoS ONE 14(1).

Seabold, Skipper, & Perktold, J. (2024, Abril 19). statsmodels.org. Retrieved from statsmodels: Econometric and statistical modeling with python: https://www.statsmodels.org/dev/generated/statsmodels.nonparametric.smoothers_lowess.lowess.html