Primer Examen Parcial

ZARATE MACIAS ALEJANDRO
CEDEÑO GONZALEZ SEBASTIAN
GUZMAN LOMELI ERNESTO RUBEN
LASSO SALDAÑA JAVIER
ORNELAS NANDE EMILIANO JAVIER
MEDINA CASTELLANOS DAVID

17 de octubre de 2025

1. Problema 1

Suponga que se quiere hacer un estudio estadístico relacionado con alguna variable aleatoria de una población en particular. Para este fin, se han recopilado datos relacionados con dicha variable a través de diversos muestreos. Los datos recopilados se presentan en el archivo CSV adjunto (exam_data.csv), donde los datos en cada una de las columnas representa un conjunto muestral de n = 5000 elementos. Utilizando dicha información determinar lo siguiente:

a Una estimación de la media y deviación estándar de la población de la cual fueron sustraídas las muestras.

b ¿Qué puede decir con respecto de la distribución de la media muestral?

c ¿Qué observaciones tiene respecto al proceso de muestreo?

1.1. Librerias

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

1.2. Leer CSV

Para la lectura del csv solamente debemos de tener precaucion con el hecho de que los datos vienen con una columna adicional al inicio llamada "Unnamed: 0", la cual es simplemente el index del dataframe original de donde salieron los datos, por lo que debemos de eliminarla al momento de leer el archivo.

```
df1 = pd.read_csv('data/exam_data.csv')
# elimina Unnamed: 0
df1 = df1.drop(columns=['Unnamed: 0'])
```

```
df1.head(10)
```

	0	1	2	3	4	5	6	7	8	9	...	
0	49.836306	45.101541	63.531488	69.009866	40.095654	84.906605	81.128278	86.037926	22.687573	89.401010	...	5
1	49.994347	45.773061	62.897416	68.556472	39.163533	85.087188	81.363247	85.743628	22.345356	88.697771	...	5
2	50.156428	45.000371	62.585317	69.176546	39.984561	85.171361	80.517995	85.979397	22.081638	89.397012	...	5
3	49.696662	45.493168	62.385483	68.946933	40.178232	84.701594	80.987396	85.579412	21.121060	88.496242	...	5
4	49.481798	44.913164	63.169940	69.125740	39.513078	84.637430	81.422197	85.073721	21.382952	88.695730	...	5
5	50.028949	45.251895	62.784015	69.964565	39.971670	85.595685	80.148644	86.341459	22.054040	89.440632	...	5
6	49.488570	44.342417	62.410067	68.973961	39.707925	85.201554	81.674802	87.206745	21.619606	88.618408	...	5
7	50.993590	44.115163	62.926313	68.687825	40.007977	85.239049	81.146014	86.339812	23.173693	89.511708	...	5
8	49.405534	44.462871	63.677012	69.391253	40.109682	86.039841	81.512589	86.119336	21.679343	88.711364	...	5
9	50.264496	45.557865	63.048611	69.854431	40.045069	85.406365	81.065284	85.939307	23.593950	88.344303	...	5

10 rows × 100 columns

1.3. Calcular media y desviación estándar poblacional

De acuerdo con lo que se solicita en el inciso a, para estimar la media y desviación estándar poblacional a partir de las muestras podemos partir de lo que nos indica el Teorema del Límite Central. Para este caso, el TLC dice que que la media muestral deberia de seguir una distribución normal, tal que:

$$\overline{X}_i = \frac{1}{n}\sum_{i=1}^{n}(x_i) \in N(\mu_{\overline{x}}, \sigma_{\overline{x}}) \tag{1}$$

Y segun lo que nos indica el TLC, la media y desviación estándar de la media muestral estan dadas por:

$$\mu_{\overline{x}} = \mu \tag{2}$$

$$\sigma_{\overline{x}} = \frac{\sigma}{\sqrt{n}} \tag{3}$$

Por lo tanto, lo que haremos primero es calcular para cada una de nuestras muestras la media y desviación estándar (NOTA: no es necesario calcular la desviación estándar para cada muestra, ya que la estimacion la podemos obtener usando la media muestral, pero para fines ilustrativos, lo haremos).

```
# Calcular la media de cada columna para obtener 'S_dist'
Xmean_dist = df1.mean(axis=0)
Xsigma_dist = df1.std(axis=0, ddof=1)  # ddof=1 Se ajusta el parametro Grados de
↪  Libertad para hacer una Desviación estándar muestral
```

Ya que tenemos estos valores, podemos imprimir un histograma de como es que se ven las distribuciones de media y desviacion estandar de cada una de nuestras muestas. Para esto vamos a requerir de una pequeña funcion que nos dará el numero de bins adecuado para la cantidad de nuestros datos usando la regla de Stugres.

```python
def get_n_bins(data):
    # Calcula el número óptimo de bins para un histograma usando la regla de Sturges.
    M = len(data)
    c = int(np.ceil(1+np.log2(M)))
    return c

n_bins = get_n_bins(Xmean_dist)
n_bins
```

Dada la cantidad de nuestros datos, el numero adecuado de bins es de 8, por lo que el siguiente paso es imprimir nuestras graficas de distribucion.

1.4. Gráficas

```python
fig, ax = plt.subplots(1, 2, figsize=(14, 5))

ax[0].hist(Xmean_dist, bins=n_bins, color='steelblue', edgecolor='black', alpha=0.7)
ax[0].set_title('Histograma de Medias Muestrales', fontsize=12, fontweight='bold')
ax[0].set_xlabel('Media Muestral', fontsize=10)
ax[0].set_ylabel('Frecuencia', fontsize=10)
ax[0].grid(axis='y', alpha=0.3, linestyle='--')

ax[1].hist(Xsigma_dist, bins=n_bins, color='coral', edgecolor='black', alpha=0.7)
ax[1].set_title('Histograma de Desviaciones Típicas Muestrales', fontsize=12,
↪  fontweight='bold')
ax[1].set_xlabel('Desviación Típica Muestral', fontsize=10)
ax[1].set_ylabel('Frecuencia', fontsize=10)
ax[1].grid(axis='y', alpha=0.3, linestyle='--')

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

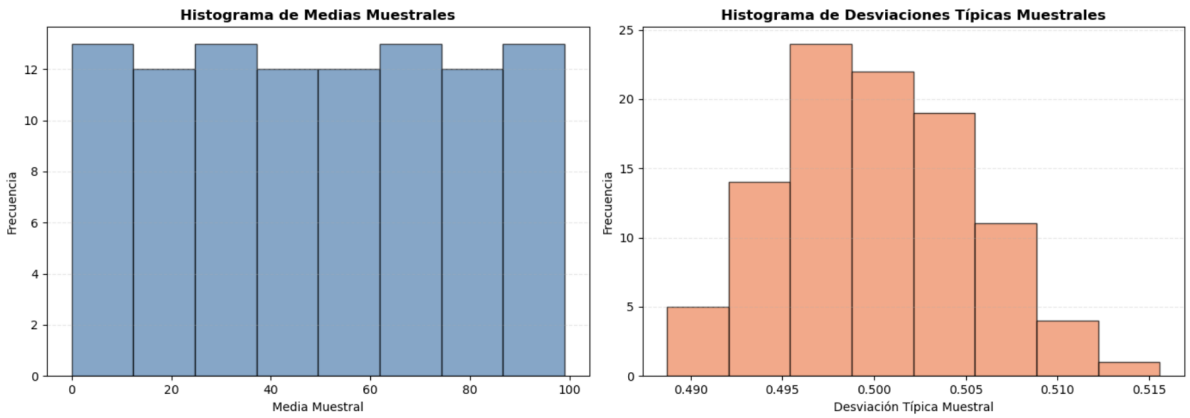


Figura 1: Histograma de datos poblacionales.

Podemos observar que la distribucion de las medias muestrales tiene una forma uniforme, mientras que la desviacion estandar muestral tiene una forma similar a una distribucion normal.

Esto ya nos indica bastante sobre si el proceso de muestreo fue adecuado o no, pero eso lo veremos mas adelante.

Lo que sigue es obtener la estimacion de la media y desviacion estandar poblacional. Para esto, usaremos las formulas que nos da el TLC.

Para el caso de la media poblacional μ, la estimacion es simplemente la media de las medias muestrales. Por otro lado, para la desviacion estandar poblacional σ, la estimacion es la desviacion estandar de las medias muestrales multiplicada por la raiz cuadrada del tamano de las muestras (n=5000).

```
n = 5000

Xmean_mu = np.mean(Xmean_dist)
Xmean_sigma = np.std(Xmean_dist)

mu = Xmean_mu
sigma = Xmean_sigma * np.sqrt(n)

print("Xmean_mu = ", Xmean_mu)
print("Xmean_sigma = ", Xmean_sigma)

print("Media de la poblacion = ", mu)
print("Desviacion estandar de la poblacion = ", sigma)
```

```
Xmean_mu =  49.49887914087672
Xmean_sigma =  29.01193008270201
Media de la poblacion =  49.49887914087672
Desviacion estandar de la poblacion =  2051.4532496788584
```

1.5. Media muestral

¿Qué puede decir con respecto de la distribución de la media muestral? La distribución de la media muestral para el caso de los datos prorcionados no sigue una distribución normal, sino que sigue una distribución uniforme. Esto puede ser un indicativo de que el proceso de muestreo no fue el adecuado, ya que de acuerdo con el Teorema del Límite Central, la media muestral debería de seguir una distribución normal.

1.6. Observaciones

¿Qué observaciones tiene respecto al proceso de muestreo? El proceso de muestreo no fue el adecuado, ya que la distribución de la media muestral no sigue una distribución normal, lo cual es un indicativo de que las muestras no fueron tomadas de manera sesgada. Una pista de como es que fueron tomadas estas muestras la podemos ver usando la funcion de pandas describe(), la cual nos da un resumen estadístico de los datos, como se muestra a continuación:

```
df1.describe()
```

	0	1	2	3	4	5	6	7	
count	5000.000000	5000.000000	5000.000000	5000.000000	5000.000000	5000.000000	5000.000000	5000.000000	5000.000(
mean	50.003154	45.002244	63.005938	69.002520	39.987563	85.005311	80.992974	85.997633	21.992
std	0.506513	0.497358	0.502912	0.502643	0.499671	0.498787	0.505004	0.494344	0.503
min	48.192637	43.251066	61.166000	67.169046	38.258716	82.943134	78.956481	84.341196	20.167
25%	49.657600	44.668883	62.664810	68.660587	39.653657	84.682096	80.652926	85.665564	21.659
50%	50.003003	44.996781	63.003768	69.004969	39.987608	85.008008	80.992202	85.990974	21.996
75%	50.351387	45.337208	63.348957	69.329303	40.323485	85.336381	81.329014	86.331532	22.333
max	51.924468	46.719402	64.857114	70.753352	41.830355	86.896359	82.673025	87.776073	23.874

8 rows × 100 columns

Podemos observar que en los valores de min y max de cada una de las muestras son muy similares entre si, lo cual nos indica que las muestras fueron tomadas de manera sesgada (casi de manera consecutiva sobre una lista ordenada), ya que en un muestreo aleatorio, los valores de minimos y maximos deberian de variar mas entre las diferentes muestras.

2. Problema 2

Una empresa dedicada a la distribución de agua potable ha recabado datos históricos relacionada con la cantidad de sulfatos y nitratos presentes en el agua. Dicha empresa tiene la suposición de que la cantidad de sulfatos en el agua está fuertemente correlacionada con la cantidad de nitratos presentes en la misma, tal que a mayor cantidad de los primeros, mayor cantidad de los segundos.

Los datos históricos relacionados con el análisis de estos componentes en el agua se presenta en la carpeta comprimida adjunta (distributed-data.zip). Nótese que en los datos existen instancias en las cuales no hay registro de las cantidades de sulfatos/nitratos, las cuales se asumen se derivan de fallas en los instrumentos utilizado para hacer estas mediciones.

Analice dichos datos y realice las siguientes tareas:

a Construir un único dataset utilizando toda la información histórica proporcionada por la empresa.

b Generar un histograma que presente el número de mediciones "utiles" VS las mediciones erroneas presentadas por los instrumentos de medición e indique: ¿Qué tan grande es la proporción de mediciones erróneas en comparación a las mediciones útiles del instrumento?

c Construir un modelo de regresión lineal en el que se establezca la relación entre las variables sulfato/nitratos utilizando el dataset generado. Indique los parámetros de dicho modelo.

d Indicar si la correlación entre la cantidad de sulfatos/nitratos es tal como la plantea la empresa. Justificar su respuesta.

2.1. Librerias

```
import os
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
```

2.2. Construir dataset

Dado que nuestros datos se encuentran en multiples archivos CSV, lo primero que haremos sera leer todos los archivos que se encuentren en la carpeta "distributed-data", por precaucuión, ignoraremos cualquier archivo que no tenga la extensión .csv. Posteriormente, concatenaremos todos los datos en un solo DataFrame de pandas usando la función pd.concat().

```
data_dir = 'data/distributed-data'
all_files = os.listdir(data_dir)
csv_files = [f for f in all_files if f.endswith('.csv')]
df2 = pd.DataFrame()
for file in csv_files:
    df_temp = pd.read_csv(os.path.join(data_dir, file))
    df2 = pd.concat([df2, df_temp], ignore_index=True)

df2.info()
```

```
RangeIndex: 772087 entries, 0 to 772086
Data columns (total 4 columns):
 #   Column    Non-Null Count   Dtype
---  ------    --------------   -----
 0   Date      772087 non-null  object
 1   sulfate   118783 non-null  float64
 2   nitrate   114349 non-null  float64
 3   ID        772087 non-null  int64
dtypes: float64(2), int64(1), object(1)
memory usage: 23.6+ MB
```

2.3. Mediciones útiles vs erróneas

Dado que las columnas de Fecha e ID no son relevantes para nuestro análisis, ignoraremos dichas columnas y nos quedaremos unicamente con las columnas de Sulfatos y Nitratos. Ademas de esto, vamos a necesitar 4 dataframes diferentes:

- df_total: Contiene todos los datos originales extraidos de los archivos CSV.
- df_erroneas: Contiene unicamente registros de las mediciones con al menos 1 valor faltante.
- df_incompletos: Contiene unicamente las filas que tengan a lo mucho un dato faltante en alguna de las dos columnas (sulfato o nitrato).
- df_completos: Contiene unicamente las filas que tengan ambos datos (sulfato y nitrato) completos.

```
df_total = df2[['sulfate', 'nitrate']]
df_erroneas = df_total[df_total.isnull().any(axis=1)]
df_incompletos = df_total[df_total.isnull().sum(axis=1) <= 1]
df_completos = df_total.dropna()

print("Total de filas en df_total:", len(df_total))
print("Total de filas en df_erroneas:", len(df_erroneas))
print("Total de filas en df_incompletos:", len(df_incompletos))
print("Total de filas en df_completos:", len(df_completos))
```

```
Total de filas en df_total: 772087
Total de filas en df_erroneas: 660285
Total de filas en df_incompletos: 121330
Total de filas en df_completos: 111802
```

2.4. Gráficas

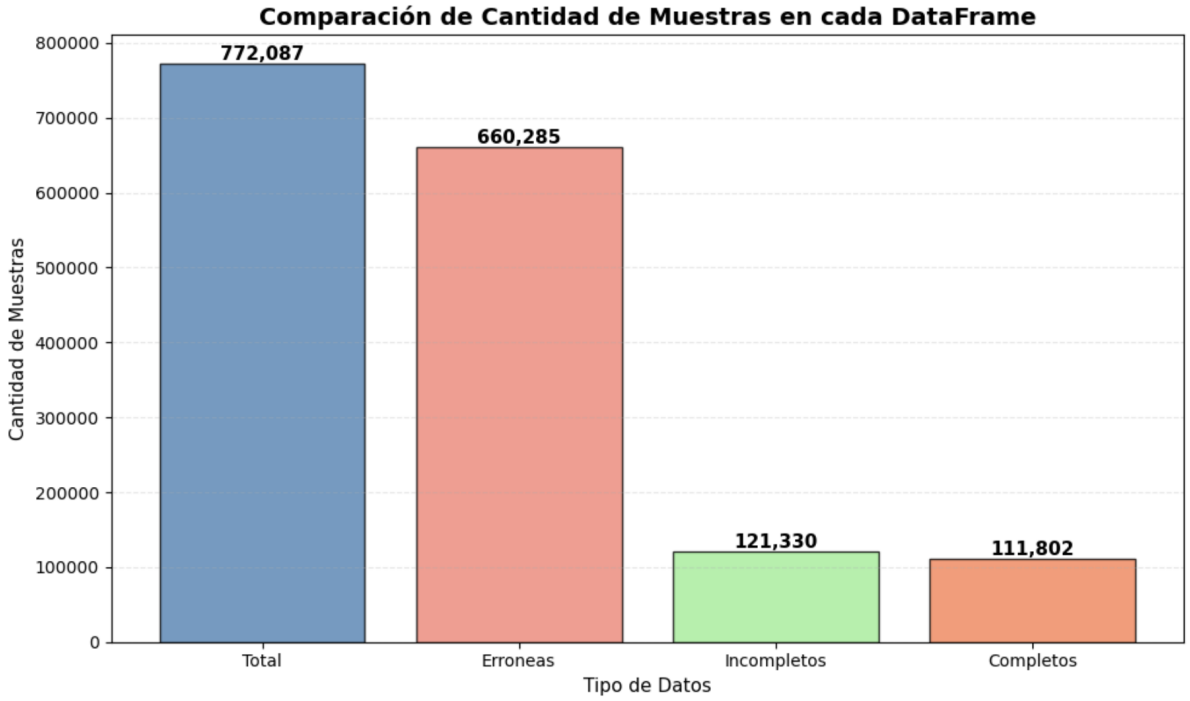
Generaremos unas graficas de barras para poder observar el numero de mediciones que tiene el dataset en cada una de sus versiones.

```
cantidad_muestras = [len(df_total), len(df_erroneas), len(df_incompletos),
↪  len(df_completos)]
nombres_dataframes = ['Total', 'Erroneas', 'Incompletos', 'Completos']

plt.figure(figsize=(10, 6))
bars = plt.bar(nombres_dataframes, cantidad_muestras, color=['steelblue', 'salmon',
↪  'lightgreen', 'coral'],
               edgecolor='black', alpha=0.8)

# Agregar valores en cada barra
for bar in bars:
    height = bar.get_height()
    plt.text(bar.get_x() + bar.get_width()/2., height,
             f'{int(height):,}',
             ha='center', va='bottom', fontsize=11, fontweight='bold')

plt.xlabel('Tipo de Datos', fontsize=11)
plt.ylabel('Cantidad de Muestras', fontsize=11)
plt.title('Comparación de Cantidad de Muestras en cada DataFrame', fontsize=14,
↪  fontweight='bold')
plt.grid(axis='y', alpha=0.3, linestyle='--')
plt.tight_layout()
plt.show()
```




```
111802*100/772087
```

14.480492483360036

Como se puede apreciar, originalmente el dataframe cuenta con 772087 registros de las mediciones historicas, pero solamente 111802 (14.48 %) de ellas pueden considerarse como mediciones útiles, es decir, aquellas que tienen tanto el valor de sulfatos como el de nitratos completos.

Lo que sigue es hacer un histograma para observar la distribución de los valores de sulfatos y nitratos en las mediciones útiles (completas). Para esto, reutilizaremos la función de get_n_bins que se usó en el problema 1.

```python
def get_n_bins(data):
    # Calcula el número óptimo de bins para un histograma usando la regla de Sturges.
    M = len(data)
    c = int(np.ceil(1+np.log2(M)))
    return c
```

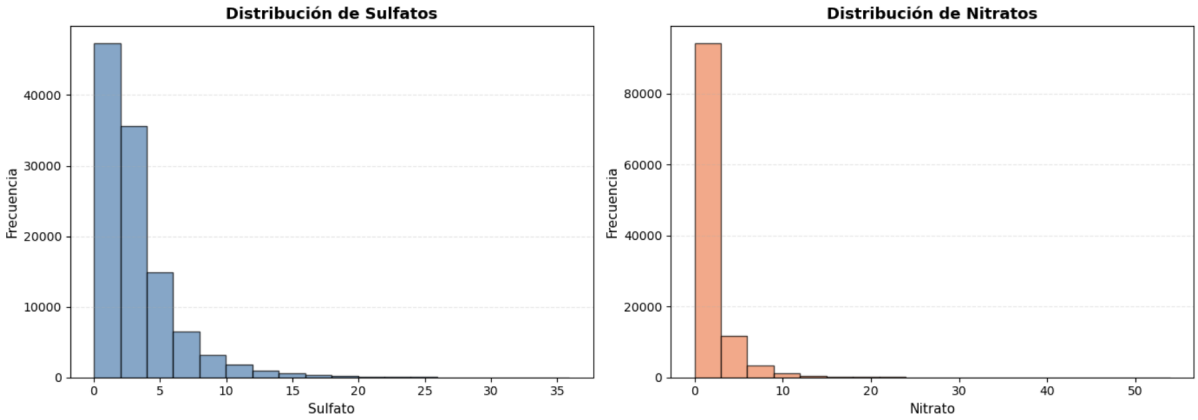
Generamos nuestros histogramas para los valores de sulfatos y nitratos en las mediciones completas.

```
fig, axes = plt.subplots(1, 2, figsize=(14, 5))

n_bins_sulfatos = get_n_bins(df_completos['sulfate'])
axes[0].hist(df_completos['sulfate'], bins=n_bins_sulfatos, color='steelblue',
             edgecolor='black', alpha=0.7)
axes[0].set_xlabel('Sulfato', fontsize=11)
axes[0].set_ylabel('Frecuencia', fontsize=11)
axes[0].set_title('Distribución de Sulfatos', fontsize=13, fontweight='bold')
axes[0].grid(axis='y', alpha=0.3, linestyle='--')

n_bins_nitratos = get_n_bins(df_completos['nitrate'])
axes[1].hist(df_completos['nitrate'], bins=n_bins_nitratos, color='coral',
             edgecolor='black', alpha=0.7)
axes[1].set_xlabel('Nitrato', fontsize=11)
axes[1].set_ylabel('Frecuencia', fontsize=11)
axes[1].set_title('Distribución de Nitratos', fontsize=13, fontweight='bold')
axes[1].grid(axis='y', alpha=0.3, linestyle='--')

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



A simple vista podemos observar que ambos histogramas tienen una distribución similar, con un pico pronunciado en valores bajos (cerca de 0) y una disminución gradual a medida que los valores aumentan. Para la hipotesis planteada de que a mayor cantidad de sulfatos, mayor cantidad de nitratos, podemos observar que en ambos histogramas, la mayoría de las mediciones se concentran en valores bajos, lo que sugiere una posible correlación positiva entre ambas variables, pero estas graficas pueden ser engañosas, ya que las escalas de ambos histogramas son diferentes, por lo que no podemos concluir nada con certeza a partir de estas graficas.

2.5. Modelo de regresión lineal

Antes que nada, dividiremos nuestro dataframe de mediciones completas (df_completos) en un conjunto de entrenamiento y un conjunto de prueba. Utilizaremos el 70 % de los datos para entrenar el modelo y el 30 % restante para evaluar su desempeño.

Para el modelo de regresión lineal, utilizaremos la librería scikit-learn. Primero, definiremos cual sera X o la entrada de nuestro modelo, y cual sera Y o la etique de nuestro modelo. En este caso, X sera la columna de sulfatos y Y sera la columna de nitratos del dataframe de mediciones completas (df_completos).

Ya con los datos preparados, podemos proceder a entrenar el modelo de regresión lineal utilizando el conjunto de entrenamiento.

```
df_train, df_test = train_test_split(df_completos, test_size=0.3)

lm = LinearRegression()
lm.fit(df_train[['sulfate']], df_train['nitrate'])
```

Para evaluar el desempeño del modelo, hicimos una pequeña funcion llamada val_model que recibe como parámetros el modelo entrenado, las entradas (X) y las etiquetas (Y) del conjunto de datos a evaluar. La función utiliza el modelo para hacer predicciones sobre las entradas proporcionadas y luego calcula un par de metricas (RSE) para ver cual fue la bondad de ajuste de nuestro modelo. Finalmente, retorna un vector de predicciones realizadas por el modelo para su posterior análisis.

```
def val_model(model: LinearRegression, x, y):
    preds = model.predict(x)
    SSD = np.sum((y-preds)**2)
    RSE = np.sqrt(SSD/(len(x)-1))
    # Calcular RSE_%
    y_mean = np.mean(y)
    RSE_rate = RSE/y_mean
    print("Score = ",model.score(x,y))
    print("SSD = ",SSD)
    print("RSE = ",RSE)
    print("RSE_% = ",RSE_rate)
    return preds
```

Validamos el modelo con nuestro conjunto de entrenamiento y nuestro conjunto de prueba para ver cómo se desempeña en ambos casos.

```
print("Datos de Entrenamiento:")
pred_train = val_model(lm, df_train[['sulfate']], df_train['nitrate'])
print("\nDatos de Prueba:")
pred_test = val_model(lm, df_test[['sulfate']], df_test['nitrate'])
```

```
Datos de Entrenamiento:
Score =  0.0036335622506815657
SSD =  500656.3424110264
RSE =  2.5292976432186958
RSE_% =  1.4799682836147412
```

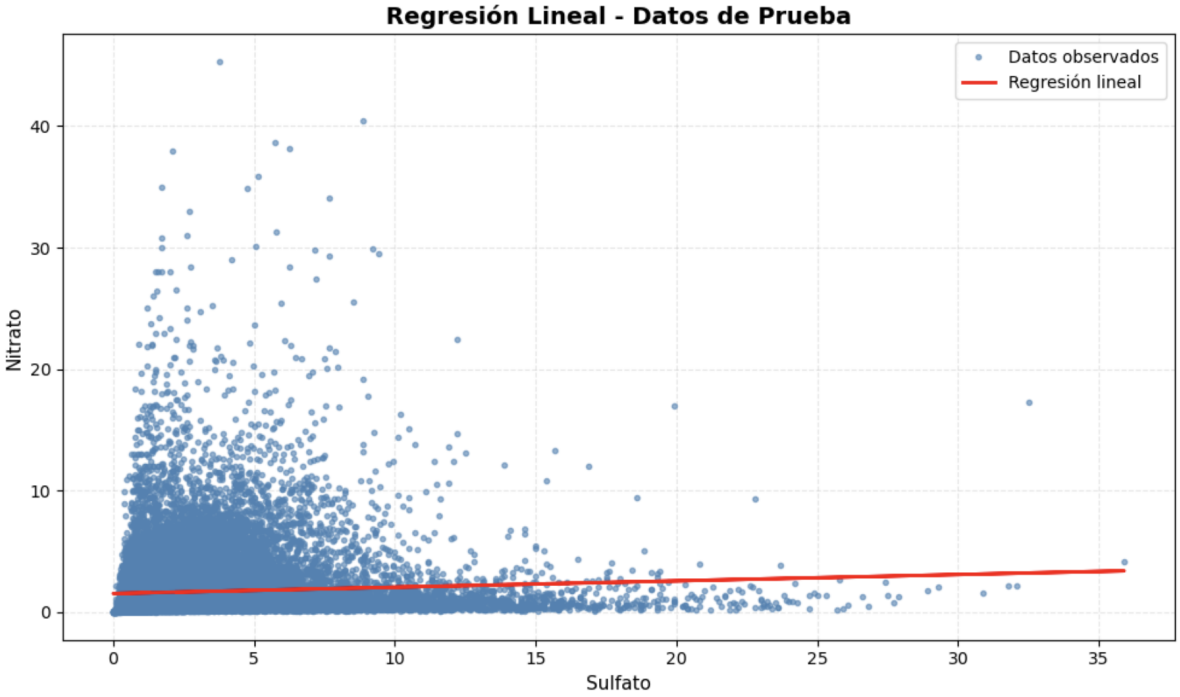
```
Datos de Prueba:
Score =  0.0045235204159411735
SSD =  206063.39866087824
RSE =  2.478671082605502
RSE_% =  1.4543059082493832
```

Como se puede observar, el modelo tiene una bondad de ajuste muy baja tanto en los datos de entrenamiento como en los datos de prueba, con un score muy cercano a 0 y un porcentaje de error (RSE) muy alto. Esto indica que el modelo no es capaz de encontrar un ajuste lineal adecuado entre las variables de sulfatos y nitratos en los datos proporcionados.

2.6. Correlación entre sulfatos y nitratos

Si imprimimos los puntos en una grafica haciendo relacion del valor de nitratos con sulfatos, podemos observar que los puntos estan muy dispersos y no siguen una tendencia lineal clara, ademas de que si agregamos la linea de regresion obtenida por el modelo, podemos ver que esta no se ajusta totalmente a los datos.

```
plt.figure(figsize=(10, 6))
plt.plot(df_test['sulfate'], df_test['nitrate'], "o", color='steelblue',
↪  markersize=3, alpha=0.6, label='Datos observados')
plt.plot(df_test['sulfate'], pred_test, color='red', linewidth=2, label='Regresión
↪  lineal')
plt.title("Regresión Lineal - Datos de Prueba", fontsize=14, fontweight='bold')
plt.xlabel('Sulfato', fontsize=11)
plt.ylabel('Nitrato', fontsize=11)
plt.legend(fontsize=10)
plt.grid(True, alpha=0.3, linestyle='--')
plt.tight_layout()
plt.show()
```



La libreria de pandas nos permite calcular correlaciones entre columnas de un DataFrame de manera sencilla utilizando el método .corr(), e incluso nos permite especificar el método de correlación que queremos utilizar (Pearson, Spearman, Kendall, etc.). En este caso, utilizaremos el método de Pearson para calcular la correlación entre las columnas de sulfatos y nitratos en nuestro DataFrame de mediciones completas (df_completos).

```
print("Matriz de Correlación:")
print(df_completos.corr())

correlacion = df_completos['sulfate'].corr(df_completos['nitrate'], method='pearson')
print(f"\nCorrelación de Pearson: {correlacion:.4f}")
```

```
Matriz de Correlacion:
            sulfate   nitrate
sulfate    1.000000  0.062434
nitrate    0.062434  1.000000

Correlacion de Pearson: 0.0624
```

Por lo tanto, podemos concluir que la hipótesis planteada por la empresa no se sostiene con los datos proporcionados, ya que la correlación entre las cantidades de sulfatos y nitratos es muy baja (0.0624), lo que indica que no existe una relación lineal significativa entre ambas variables en el conjunto de datos analizado.