

```
In [235... import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
import random
```

Cargar dataset y dividir en 40% train 40% validation y 20% test

```
In [236... # Cargar dataset
datos = pd.read_csv('Valhalla23.csv')

# Definir semilla
semilla = 2992
random.seed(semilla)

# Dividir en subsets
X = datos[['Celsius']]
y = datos['Valks']

X_train_val, X_test, y_train_val, y_test = train_test_split(X, y, test_size=
X_train, X_val, y_train, y_val = train_test_split(X_train_val, y_train_val,
```

Modelo base

```
In [237... # Se crea el objeto del modelo y se entrena
from sklearn.linear_model import SGDRegressor
model = SGDRegressor(max_iter= 10000000, eta0 = 1e-4, random_state = semilla
model.fit(X_train, y_train)
```

```
Out[237... SGDRegressor
SGDRegressor(eta0=0.0001, max_iter=10000000, penalty=None, random_s
tate=2992)
```

Error cuadrático medio

```
In [238... from sklearn.metrics import mean_squared_error

# Error de datos de entrenamiento
y_train_pred = model.predict(X_train)
MSE_train = mean_squared_error(y_train, y_train_pred)
print('MSE train=', MSE_train)

# Error de datos de validacion
y_val_pred = model.predict(X_val)
```

```

MSE_val = mean_squared_error(y_val, y_val_pred)
print('MSE val=', MSE_val)

# Error de datos de entrenamiento
y_test_pred = model.predict(X_test)
MSE_test = mean_squared_error(y_test, y_test_pred)
print('MSE test=', MSE_test)

```

MSE train= 1140.8513979184747

MSE val= 1052.4601285152605

MSE test= 1278.414945946573

Gráfica

```

In [239... # Crear los datos para la recta
slope = model.coef_[0]
intercept = model.intercept_[0]
X_plot = np.linspace(X.min(), X.max(), 100)
y_plot = slope * X_plot + intercept

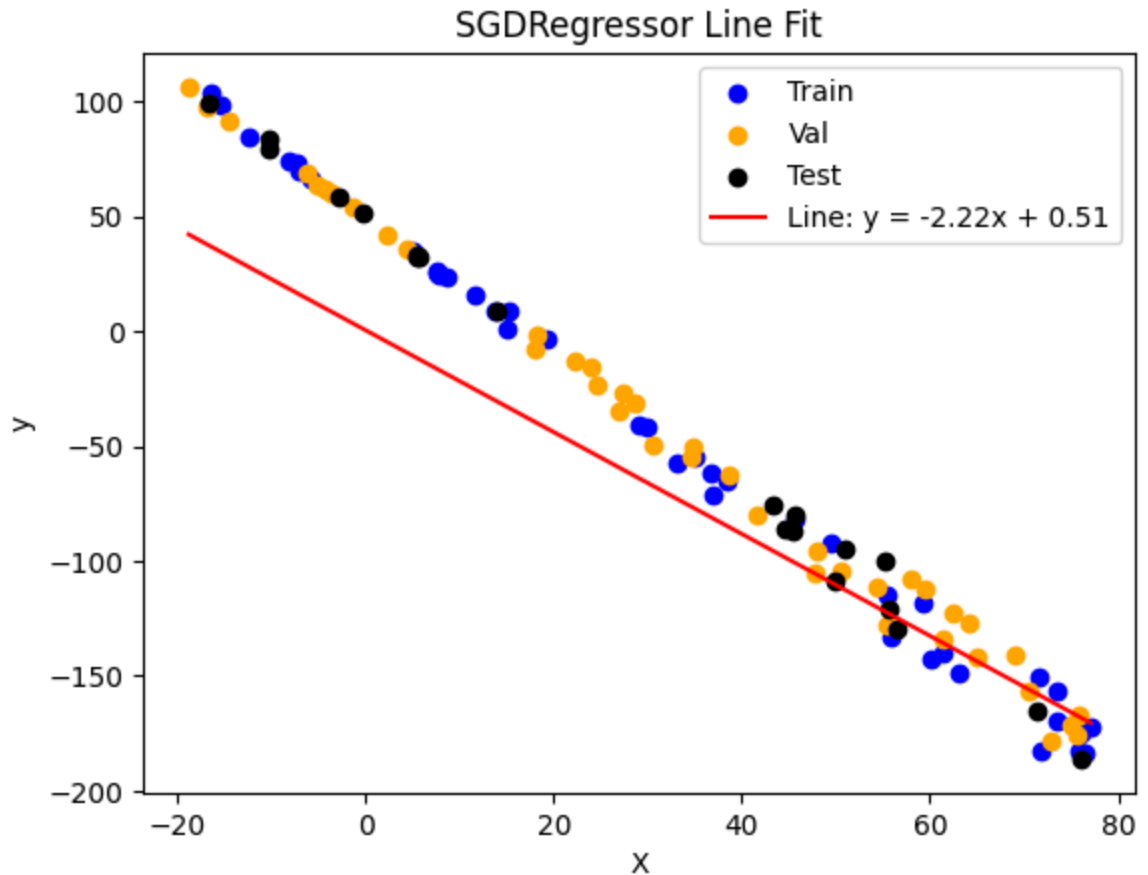
# Graficar los datos
plt.scatter(X_train, y_train, color='blue', label='Train')
plt.scatter(X_val, y_val, color='orange', label='Val')
plt.scatter(X_test, y_test, color='black', label='Test')

# Graficar la recta
plt.plot(X_plot, y_plot, color='red', label=f'Line: y = {slope:.2f}x + {intercept:.2f}')

plt.xlabel('X')
plt.ylabel('y')
plt.title('SGDRegressor Line Fit')
plt.legend()

plt.show()

```



Prueba con múltiples modelos

```
In [240... cant_datos = np.linspace(2, 39, 20).astype(int)
errores_train_prom = []
errores_val_prom = []
errores_train_std = []
errores_val_std = []

train_index = list(X_train.index)

for i in cant_datos:
    errores_train = np.zeros(100)
    errores_val = np.zeros(100)

    for k in range(100):
        # Selección aleatoria de índices
        train_index_i = random.sample(range(len(X_train)), i)

        # Crear subconjuntos de entrenamiento y validación
        X_train_i = X_train.iloc[train_index_i]
        y_train_i = y_train.iloc[train_index_i]

        model = SGDRegressor(max_iter=1000000, eta0=1e-4, random_state=semi1
        model.fit(X_train_i, y_train_i)

        # Predecir en el subconjunto de entrenamiento y validación
        y_train_pred = model.predict(X_train_i)
```

```
y_val_pred = model.predict(X_val)

# Calcular el error cuadrático medio
errores_train[k] = mean_squared_error(y_train_i, y_train_pred)
errores_val[k] = mean_squared_error(y_val, y_val_pred)

# Promedio y desviación estándar de los errores
errores_train_prom.append(np.mean(errores_train))
errores_val_prom.append(np.mean(errores_val))
errores_train_std.append(np.std(errores_train))
errores_val_std.append(np.std(errores_val))

# Agregar errores base
errores_train_prom.append(MSE_train)
errores_val_prom.append(MSE_val)
errores_train_std.append(np.std([MSE_train])) # Desviación estándar para un
errores_val_std.append(np.std([MSE_val]))      # Desviación estándar para un

# Crear un DataFrame con los resultados
resultados = pd.DataFrame({
    'Subconjunto': cant_datos.tolist() + ['Línea base'],
    'Error_Entrenamiento': errores_train_prom,
    'Error_Validacion': errores_val_prom,
    'STD_Entrenamiento': errores_train_std,
    'STD_Validacion': errores_val_std
})

print(resultados)
```

| | Subconjunto | Error_Entrenamiento | Error_Validacion | STD_Entrenamiento | \ |
|----|-------------|---------------------|------------------|-------------------|---|
| 0 | 2 | 180.212423 | 5155.204524 | 105.688800 | |
| 1 | 3 | 205.074417 | 1020.083286 | 101.120961 | |
| 2 | 5 | 120.401793 | 213.002744 | 49.083920 | |
| 3 | 7 | 401.777448 | 529.410428 | 418.763064 | |
| 4 | 9 | 328.099899 | 403.666770 | 413.749875 | |
| 5 | 11 | 406.158374 | 493.938731 | 441.956331 | |
| 6 | 13 | 608.985834 | 656.343954 | 479.415801 | |
| 7 | 15 | 350.699960 | 382.206798 | 439.871878 | |
| 8 | 17 | 675.012573 | 688.067884 | 487.987883 | |
| 9 | 19 | 641.207631 | 660.300300 | 482.938053 | |
| 10 | 21 | 920.283261 | 892.991153 | 393.760947 | |
| 11 | 23 | 828.224962 | 805.250138 | 457.770172 | |
| 12 | 25 | 795.143610 | 774.361938 | 466.594387 | |
| 13 | 27 | 894.002620 | 835.406970 | 440.642901 | |
| 14 | 29 | 810.959602 | 775.301194 | 453.383393 | |
| 15 | 31 | 1065.084197 | 983.222853 | 273.608093 | |
| 16 | 33 | 813.417773 | 761.904722 | 465.199191 | |
| 17 | 35 | 1052.851137 | 960.759866 | 274.354136 | |
| 18 | 37 | 920.864327 | 850.941399 | 405.955687 | |
| 19 | 39 | 803.439297 | 741.763435 | 451.434078 | |
| 20 | Línea base | 1140.851398 | 1052.460129 | 0.000000 | |

| | STD_Validacion |
|----|----------------|
| 0 | 17599.886648 |
| 1 | 4433.805527 |
| 2 | 179.469896 |
| 3 | 458.208974 |
| 4 | 442.939466 |
| 5 | 481.423076 |
| 6 | 476.018568 |
| 7 | 444.417006 |
| 8 | 462.306587 |
| 9 | 472.812446 |
| 10 | 355.781312 |
| 11 | 416.301665 |
| 12 | 429.910348 |
| 13 | 392.195596 |
| 14 | 420.338267 |
| 15 | 237.128968 |
| 16 | 422.819102 |
| 17 | 243.437263 |
| 18 | 367.827299 |
| 19 | 410.009730 |
| 20 | 0.000000 |

```
In [241... # Crear la gráfica
plt.figure(figsize=(10, 6))

# Graficar los errores de entrenamiento
plt.plot(cant_datos, errores_train_prom[: -1], label='Error de Entrenamiento')

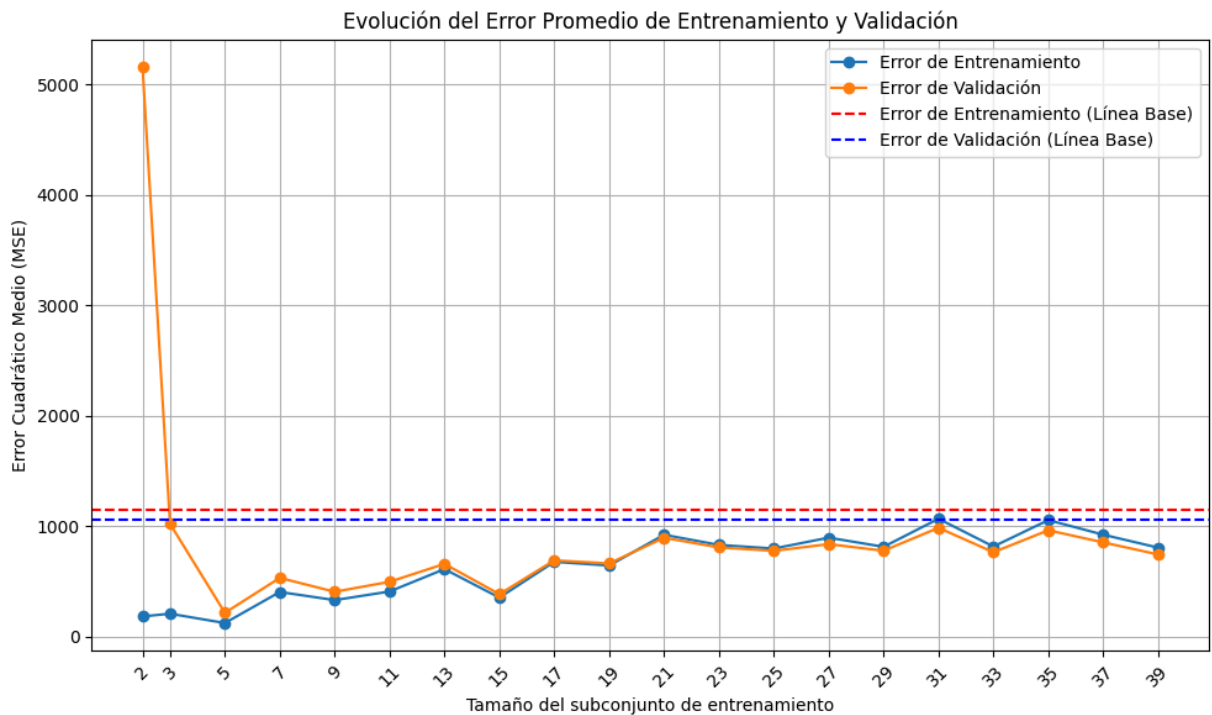
# Graficar los errores de validación
plt.plot(cant_datos, errores_val_prom[: -1], label='Error de Validación', mar

# Graficar los errores de la línea base
```

```
plt.axhline(y=errores_train_prom[-1], color='r', linestyle='--', label='Error
plt.axhline(y=errores_val_prom[-1], color='b', linestyle='--', label='Error

# Añadir etiquetas y leyenda
plt.xlabel('Tamaño del subconjunto de entrenamiento')
plt.ylabel('Error Cuadrático Medio (MSE)')
plt.title('Evolución del Error Promedio de Entrenamiento y Validación')
plt.legend()
plt.grid(True)
plt.xticks(cant_datos, rotation=45)
plt.tight_layout()

# Mostrar la gráfica
plt.show()
```



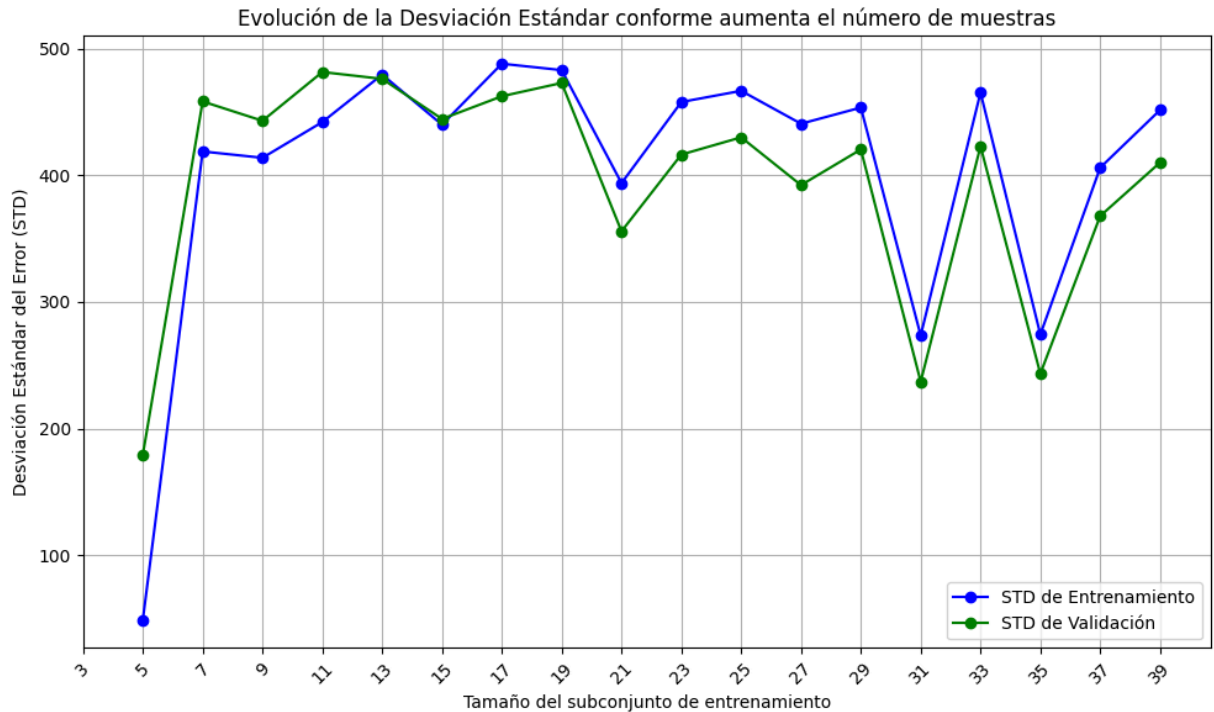
```
In [242... # Crear la gráfica de la desviación estándar
plt.figure(figsize=(10, 6))

# Graficar la desviación estándar de los errores de entrenamiento
plt.plot(cant_datos[2:], errores_train_std[2:-1], label='STD de Entrenamiento')

# Graficar la desviación estándar de los errores de validación
plt.plot(cant_datos[2:], errores_val_std[2:-1], label='STD de Validación', m

# Agregar etiquetas y leyenda
plt.xlabel('Tamaño del subconjunto de entrenamiento')
plt.ylabel('Desviación Estándar del Error (STD)')
plt.title('Evolución de la Desviación Estándar conforme aumenta el número de
plt.legend()
plt.grid(True)
plt.xticks(cant_datos[1:], rotation=45)
plt.tight_layout()
```

```
# Mostrar la gráfica
plt.show()
```



Sesgo y varianza

En la gráfica de error contra tamaño de subconjunto se puede observar que cuando el tamaño de subconjunto es de 2, el sesgo es demasiado grande puesto que el modelo se ajusta solamente a esos dos puntos, dando como resultado un modelo que tiene mucho error en los datos de validación. El resto de los modelos tienen menos sesgo y por lo tanto menos error, por debajo de 1100, sin embargo, hay mucha varianza entre ellos ya que se tiene un rango de error 200 hasta 1050, además de que la desviación estándar de los errores para cada tamaño de subconjunto de entrenamiento también es considerable, esto significa que el cambio en la selección aleatoria de datos influye significativamente en el desempeño (varianza alta). Esto quiere decir que la complejidad o parámetros del modelo deberían cambiarse.

Ajustes obtenidos

En el caso del modelo que utiliza 2 elementos para entrenar se tiene un modelo con underfit, puesto que el error es demasiado grande y no logra hacer predicciones adecuadamente. Por otro lado, cuando tenemos muchos elementos es probable que el modelo no generalice bien y no obtenga buenas soluciones para datos no vistos, esto quiere decir que el modelo sufre de overfit. A lo largo de todos los modelos se tiene un error relativamente grande, esto puede ser debido a que los parámetros seleccionados para entrenar los modelos no son los mejores.

Con estas condiciones, el modelo que tiene menor error y menor variación es el que utiliza 5 muestras para entrenamiento.

Modelo seleccionado

```
In [243... # Selección aleatoria de índices
train_index = random.sample(range(len(X_train)), 5)

# Crear subconjuntos de entrenamiento y validación
X_train2 = X_train.iloc[train_index]
y_train2 = y_train.iloc[train_index]
model2 = SGDRegressor(max_iter= 10000000, eta0 = 1e-4, random_state = semilla)
model2.fit(X_train2, y_train2)
```

```
Out[243... SGDRegressor
SGDRegressor(eta0=0.0001, max_iter=10000000, penalty=None, random_state=2992)
```

```
In [244... # Error de datos de entrenamiento
y_train_pred2 = model2.predict(X_train2)
MSE_train2 = mean_squared_error(y_train2, y_train_pred2)
print('MSE train=', MSE_train2)

# Error de datos de validación
y_val_pred2 = model2.predict(X_val)
MSE_val2 = mean_squared_error(y_val, y_val_pred2)
print('MSE val=', MSE_val2)

# Error de datos de entrenamiento
y_test_pred2 = model2.predict(X_test)
MSE_test2 = mean_squared_error(y_test, y_test_pred2)
print('MSE test=', MSE_test2)
```

MSE train= 132.37275387562096

MSE val= 220.51885952604394

MSE test= 269.1333022049183

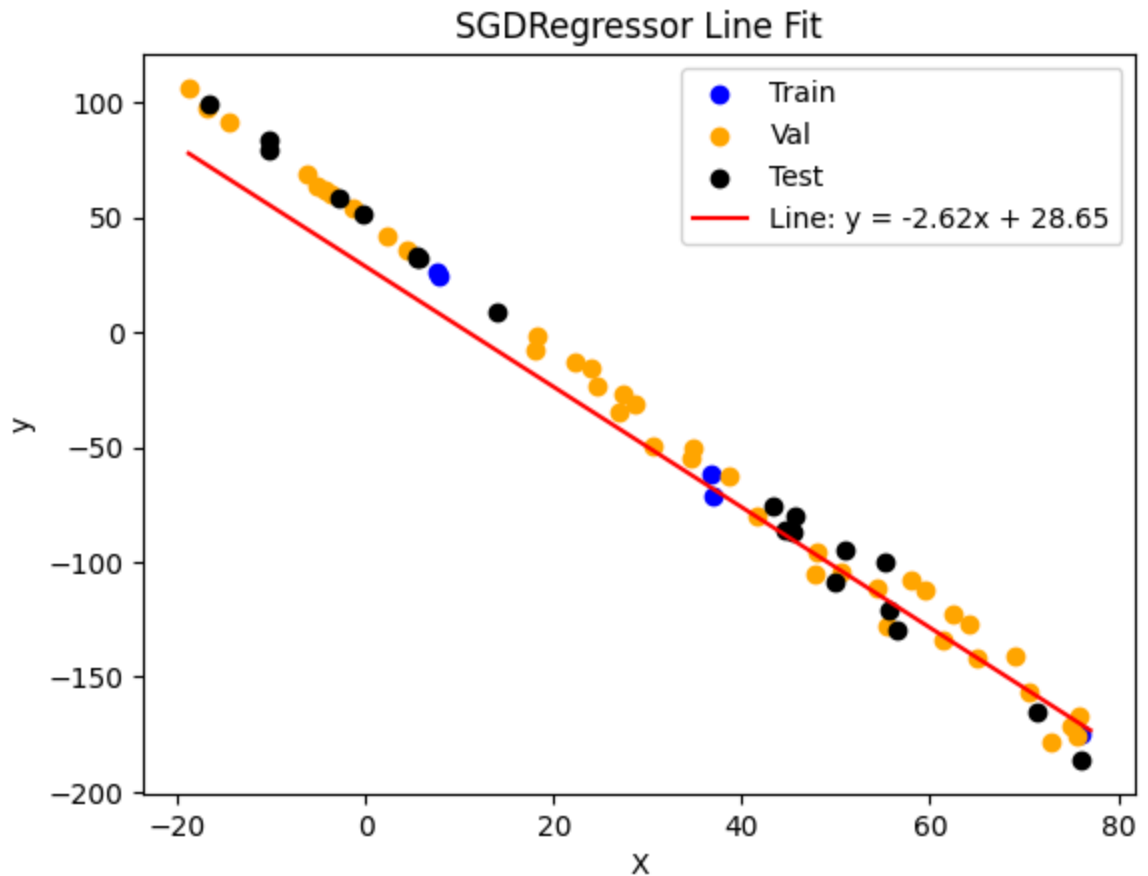
```
In [245... # Crear los datos para la recta
slope = model2.coef_[0]
intercept = model2.intercept_[0]
X_plot = np.linspace(X.min(), X.max(), 100)
y_plot = slope * X_plot + intercept

# Graficar los datos
plt.scatter(X_train2, y_train2, color='blue', label='Train')
plt.scatter(X_val, y_val, color='orange', label='Val')
plt.scatter(X_test, y_test, color='black', label='Test')

# Graficar la recta
plt.plot(X_plot, y_plot, color='red', label=f'Line: y = {slope:.2f}x + {intercept:.2f}')
```



```
plt.xlabel('X')  
plt.ylabel('y')  
plt.title('SGDRegressor Line Fit')  
plt.legend()  
  
plt.show()
```



Conclusiones

Los errores obtenidos para el modelo que utiliza 5 muestras son entre 100 y 200, mientras que para el modelo base son de aproximadamente 1,000. Esto es debido a los parámetros seleccionados para entrenar los modelos, si reducimos α y la tolerancia, el modelo que utiliza 40 muestras obtendría menor error ya que sería capaz de converger en mejores valores para todos los pesos θ , sin embargo, con estas condiciones, el que utiliza 5 es el que logra obtener una mejor solución. Otro factor que puede afectar los resultados es la semilla, si se prueba con una semilla diferente los resultados cambian.