In [ ]:
```
%pip install pandas
%pip install nbconvert
```

```
Requirement already satisfied: pandas in /usr/local/lib/python3.10/dist-packages (2.
1.4)
Requirement already satisfied: numpy<2,>=1.22.4 in /usr/local/lib/python3.10/dist-pac
kages (from pandas) (1.26.4)
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.10/di
st-packages (from pandas) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-package
s (from pandas) (2024.1)
Requirement already satisfied: tzdata>=2022.1 in /usr/local/lib/python3.10/dist-packa
ges (from pandas) (2024.1)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (f
rom python-dateutil>=2.8.2->pandas) (1.16.0)
Requirement already satisfied: nbconvert in /usr/local/lib/python3.10/dist-packages
(6.5.4)
Requirement already satisfied: lxml in /usr/local/lib/python3.10/dist-packages (from
nbconvert) (4.9.4)
Requirement already satisfied: beautifulsoup4 in /usr/local/lib/python3.10/dist-packa
ges (from nbconvert) (4.12.3)
Requirement already satisfied: bleach in /usr/local/lib/python3.10/dist-packages (fro
m nbconvert) (6.1.0)
Requirement already satisfied: defusedxml in /usr/local/lib/python3.10/dist-packages
(from nbconvert) (0.7.1)
Requirement already satisfied: entrypoints>=0.2.2 in /usr/local/lib/python3.10/dist-p
ackages (from nbconvert) (0.4)
Requirement already satisfied: jinja2>=3.0 in /usr/local/lib/python3.10/dist-packages
(from nbconvert) (3.1.4)
Requirement already satisfied: jupyter-core>=4.7 in /usr/local/lib/python3.10/dist-pa
ckages (from nbconvert) (5.7.2)
Requirement already satisfied: jupyterlab-pygments in /usr/local/lib/python3.10/dist-
packages (from nbconvert) (0.3.0)
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.10/dist-pack
ages (from nbconvert) (2.1.5)
Requirement already satisfied: mistune<2,>=0.8.1 in /usr/local/lib/python3.10/dist-pa
ckages (from nbconvert) (0.8.4)
Requirement already satisfied: nbclient>=0.5.0 in /usr/local/lib/python3.10/dist-pack
ages (from nbconvert) (0.10.0)
Requirement already satisfied: nbformat>=5.1 in /usr/local/lib/python3.10/dist-packag
es (from nbconvert) (5.10.4)
Requirement already satisfied: packaging in /usr/local/lib/python3.10/dist-packages
(from nbconvert) (24.1)
Requirement already satisfied: pandocfilters>=1.4.1 in /usr/local/lib/python3.10/dist
-packages (from nbconvert) (1.5.1)
Requirement already satisfied: pygments>=2.4.1 in /usr/local/lib/python3.10/dist-pack
ages (from nbconvert) (2.16.1)
Requirement already satisfied: tinycss2 in /usr/local/lib/python3.10/dist-packages (f
rom nbconvert) (1.3.0)
Requirement already satisfied: traitlets>=5.0 in /usr/local/lib/python3.10/dist-packa
ges (from nbconvert) (5.7.1)
Requirement already satisfied: platformdirs>=2.5 in /usr/local/lib/python3.10/dist-pa
ckages (from jupyter-core>=4.7->nbconvert) (4.2.2)
Requirement already satisfied: jupyter-client>=6.1.12 in /usr/local/lib/python3.10/di
st-packages (from nbclient>=0.5.0->nbconvert) (6.1.12)
Requirement already satisfied: fastjsonschema>=2.15 in /usr/local/lib/python3.10/dist
-packages (from nbformat>=5.1->nbconvert) (2.20.0)
Requirement already satisfied: jsonschema>=2.6 in /usr/local/lib/python3.10/dist-pack
ages (from nbformat>=5.1->nbconvert) (4.23.0)
Requirement already satisfied: soupsieve>1.2 in /usr/local/lib/python3.10/dist-packag
es (from beautifulsoup4->nbconvert) (2.6)
Requirement already satisfied: six>=1.9.0 in /usr/local/lib/python3.10/dist-packages
(from bleach->nbconvert) (1.16.0)
```

```
Requirement already satisfied: webencodings in /usr/local/lib/python3.10/dist-package
s (from bleach->nbconvert) (0.5.1)
Requirement already satisfied: attrs>=22.2.0 in /usr/local/lib/python3.10/dist-packag
es (from jsonschema>=2.6->nbformat>=5.1->nbconvert) (24.2.0)
Requirement already satisfied: jsonschema-specifications>=2023.03.6 in /usr/local/li
b/python3.10/dist-packages (from jsonschema>=2.6->nbformat>=5.1->nbconvert) (2023.12.
1)
Requirement already satisfied: referencing>=0.28.4 in /usr/local/lib/python3.10/dist-
packages (from jsonschema>=2.6->nbformat>=5.1->nbconvert) (0.35.1)
Requirement already satisfied: rpds-py>=0.7.1 in /usr/local/lib/python3.10/dist-packa
ges (from jsonschema>=2.6->nbformat>=5.1->nbconvert) (0.20.0)
Requirement already satisfied: pyzmq>=13 in /usr/local/lib/python3.10/dist-packages
(from jupyter-client>=6.1.12->nbclient>=0.5.0->nbconvert) (24.0.1)
Requirement already satisfied: python-dateutil>=2.1 in /usr/local/lib/python3.10/dist
-packages (from jupyter-client>=6.1.12->nbclient>=0.5.0->nbconvert) (2.8.2)
Requirement already satisfied: tornado>=4.1 in /usr/local/lib/python3.10/dist-package
s (from jupyter-client>=6.1.12->nbclient>=0.5.0->nbconvert) (6.3.3)
```

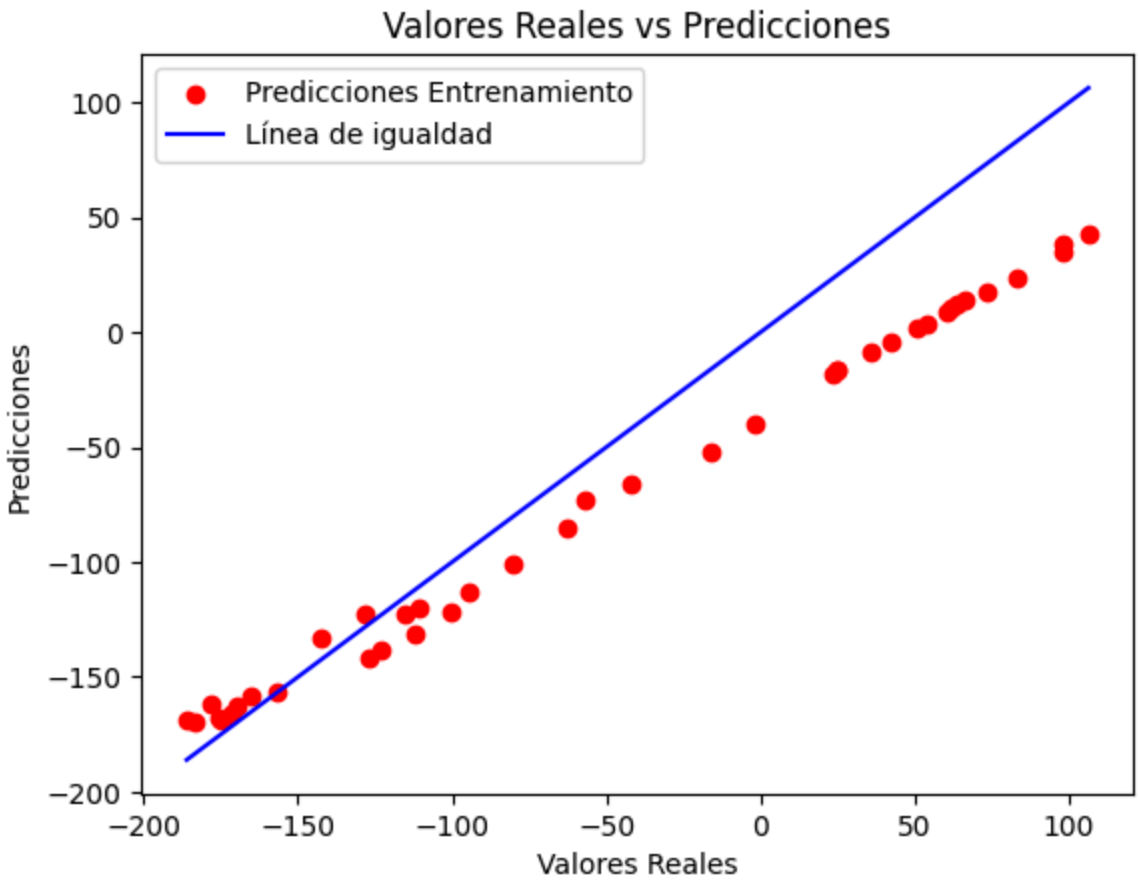
In [ ]:
```
import pandas as pd
import math
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import SGDRegressor
from sklearn.metrics import  mean_squared_error
import numpy as np

oCelsiusValhalla = pd.read_csv('/content/Valhalla23.csv')
```

In [ ]:
```
# Dividir los datos en conjuntos de entrenamiento y prueba
oCelsiusValhallaTrain, oTest = train_test_split(oCelsiusValhalla, test_size=0.2)
oTrain, oValidacion = train_test_split(oCelsiusValhallaTrain, test_size=0.5)

print("Tamaño Train: ", len(oTrain))
print("Tamaño Validacion: ", len(oValidacion))
print("Tamaño Test: ", len(oTest))
# Crear el modelo de regresión lineal
oModelo1 = SGDRegressor(eta0=1E-4,max_iter=1000000,random_state=5194)

oModelo1.fit(oTrain[["Celsius"]], oTrain["Valks"])

oYPredTrain = oModelo1.predict(oTrain[["Celsius"]])
oYPredValidacion = oModelo1.predict(oValidacion[["Celsius"]])
oYPredTest = oModelo1.predict(oTest[["Celsius"]])
```

```
Tamaño Train:  40
Tamaño Validacion:  40
Tamaño Test:  20
```

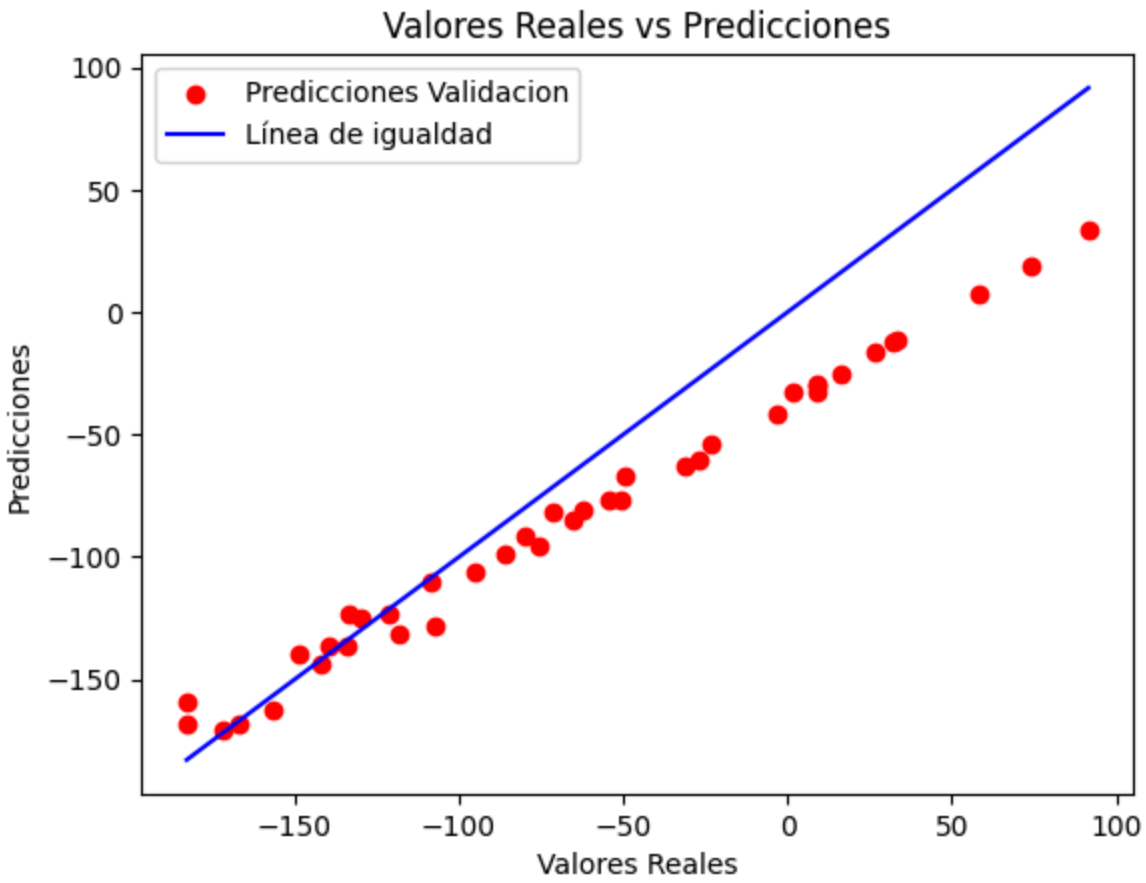
In [ ]:
```
fTrainMSE = mean_squared_error(oTrain["Valks"], oYPredTrain)
print("Mean Squared Error Training: ", fTrainMSE)

plt.scatter(oTrain["Valks"], oYPredTrain, color='red', label='Predicciones Entrenamier
plt.plot([oTrain["Valks"].min(), oTrain["Valks"].max()], [oTrain["Valks"].min(), oTrai
plt.xlabel('Valores Reales')
plt.ylabel('Predicciones')
plt.title('Valores Reales vs Predicciones')
plt.legend()
plt.show()
```

```
Mean Squared Error Training:  1271.507141428822
```



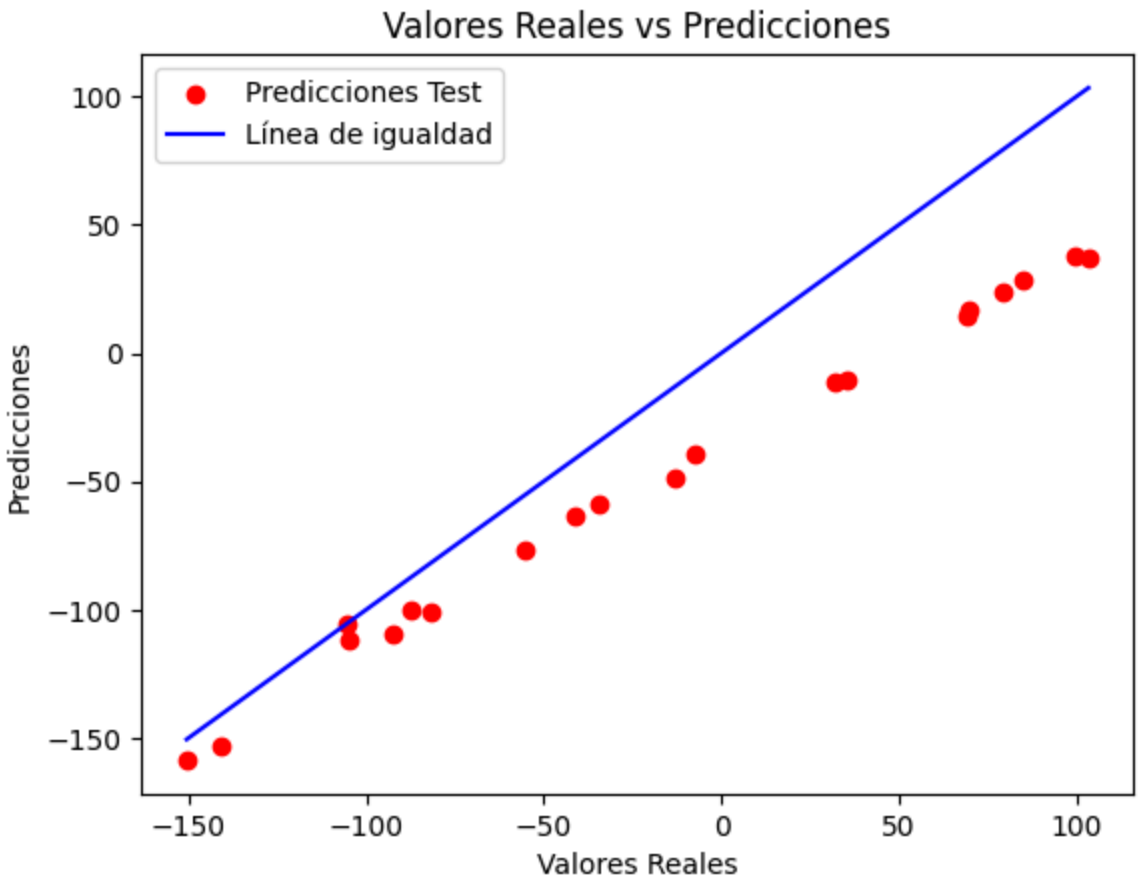
In [ ]:
```
fValidacionMSE = mean_squared_error(oValidacion["Valks"], oYPredValidacion)
print("Mean Squared Error Validacion: ", fValidacionMSE)

plt.scatter(oValidacion["Valks"], oYPredValidacion, color='red', label='Predicciones V
plt.plot([oValidacion["Valks"].min(), oValidacion["Valks"].max()], [oValidacion["Valks
plt.xlabel('Valores Reales')
plt.ylabel('Predicciones')
plt.title('Valores Reales vs Predicciones')
plt.legend()
plt.show()
```

```
Mean Squared Error Validacion:  793.0406433319511
```



In [ ]:
```
fTestMSE = mean_squared_error(oTest["Valks"], oYPredTest)
print("Mean Squared Error Test: ", mse)

plt.scatter(oTest["Valks"], oYPredTest, color='red', label='Predicciones Test')
plt.plot([oTest["Valks"].min(), oTest["Valks"].max()], [oTest["Valks"].min(), oTest["\
plt.xlabel('Valores Reales')
plt.ylabel('Predicciones')
plt.title('Valores Reales vs Predicciones')
plt.legend()
plt.show()
```

```
Mean Squared Error Test:  793.0406433319511
```



In [ ]:
```
oInstances = np.random.choice(range(3, 40), 20, replace=False)
oInstances[0] = 2 # Para que siempre tenga el valor 2
oInstances = sorted(oInstances) # Ordenamos las instancias
print(oInstances)
```

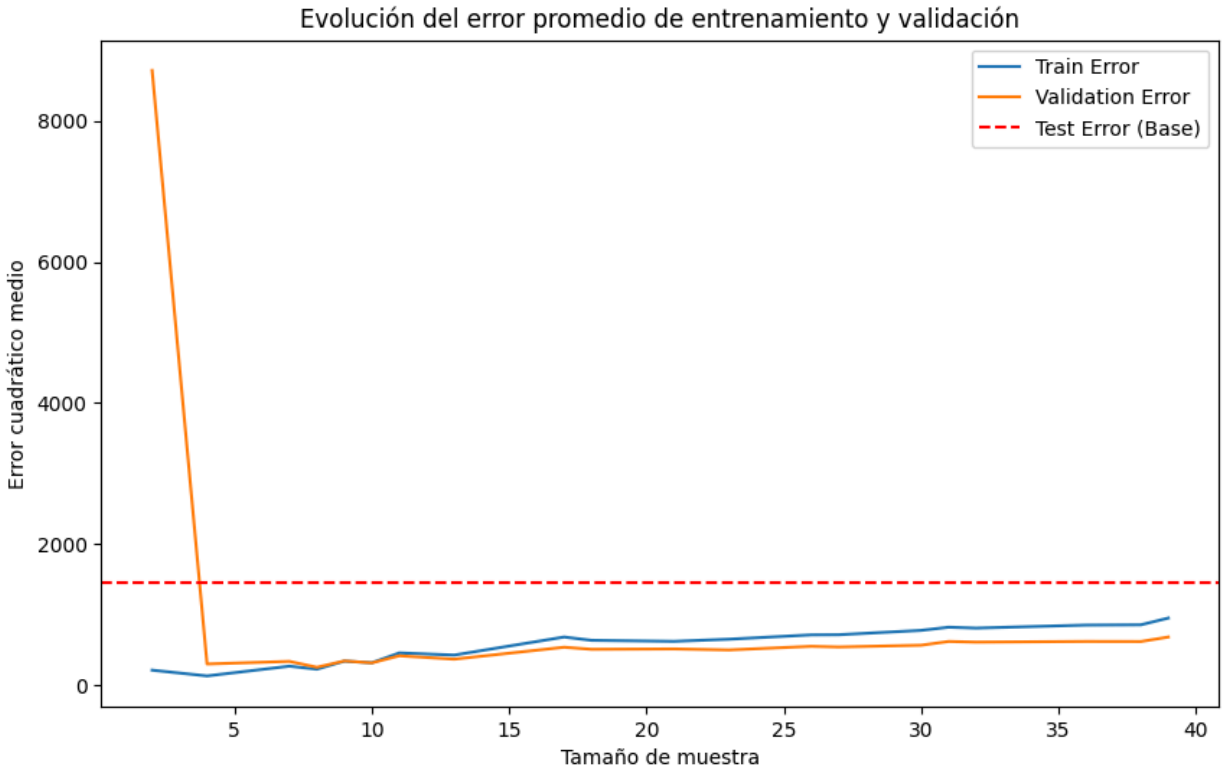
```
[2, 4, 7, 8, 9, 10, 11, 13, 17, 18, 21, 23, 26, 27, 30, 31, 32, 36, 38, 39]
```

In [ ]:
```
oTrainErrors = []
oValErrors = []

for instance in oInstances:
    oTrainMSETemp = []
    oValMSETemp = []
    for _ in range(100):

        oTrainSubset = oCelsiusValhalla.sample(n=instance)
        oModeloTemp = SGDRegressor(eta0=1E-4, max_iter=1000000)
        oModeloTemp.fit(oTrainSubset[['Celsius']], oTrainSubset['Valks'])

        # Cálculo del MSE
        fTrainingMSE = mean_squared_error(oTrainSubset['Valks'], oModeloTemp.predict(o
        fValidacionMSE = mean_squared_error(oValidacion['Valks'], oModeloTemp.predict(
        oTrainMSETemp.append(fTrainingMSE)
        oValMSETemp.append(fValidacionMSE)

    # Almacenar los errores promedios
    oTrainErrors.append(np.mean(oTrainMSETemp))
    oValErrors.append(np.mean(oValMSETemp))
```

In [ ]:
```
print(len(oTrainErrors))
print(len(oValErrors))
```

```
20
20
```

In [ ]:
```
plt.figure(figsize=(10, 6))
plt.plot(oInstances , oTrainErrors, label='Train Error')
plt.plot(oInstances , oValErrors, label='Validation Error')
plt.axhline(y=fTestMSE, color='red', linestyle='--', label='Test Error (Base)')
plt.xlabel('Tamaño de muestra')
plt.ylabel('Error cuadrático medio')
plt.title('Evolución del error promedio de entrenamiento y validación')
plt.legend()
plt.show()
```



En esta grafica se puede ver que el caso de validacion empezo con un error muy grande y fue disminuyendo mientras más modelos iban siendo ejecutados. En el caso de los datos de Training el tamaño de la muestra era muy importante para el error cuadratico, ya que mientras el tamaño de la muestra iba aumentando el Error Cuadratico tambien lo hacia.

Despues de ver la grafica el tamaño de muestra adecuado podemo dejarlo a 20 muestas para tener suficiente información sin necesidad de aumentar mucho el error cuadratico

In [ ]:
```
iTamañoOptimo = 20
oTrainOptimal = oCelsiusValhalla.sample(n=iTamañoOptimo)
oModeloOptimo = SGDRegressor( eta0=1E-4, max_iter=1000000)
oModeloOptimo.fit(oTrainOptimal[['Celsius']], oTrainOptimal['Valks'])

oYPredTrainOpt = oModeloOptimo.predict(oTrain[["Celsius"]])
oYPredValidacionOpt = oModeloOptimo.predict(oValidacion[["Celsius"]])
oYPredTestOpt = oModeloOptimo.predict(oTest[["Celsius"]])
```

In [ ]:
```
fTrainOptMSE = mean_squared_error(oTrain["Valks"], oYPredTrainOpt)
fValidacionOptMSE = mean_squared_error(oValidacion["Valks"], oYPredValidacionOpt)
fTestOptMSE = mean_squared_error(oTest["Valks"], oYPredTestOpt)

print("Modelo 1")
print("Mean Squared Error Training: ", fTrainMSE)
print("Mean Squared Error Validation: ", fValidacionMSE)
print("Mean Squared Error Test: ", fTestMSE)

print("Modelo Optimo")
print("Mean Squared Error Training: ", fTrainOptMSE)
print("Mean Squared Error Validation: ", fValidacionOptMSE)
print("Mean Squared Error Test: ", fTestOptMSE)
```

```
Modelo 1
Mean Squared Error Training:  1271.507141428822
Mean Squared Error Validation:  793.0406433319511
Mean Squared Error Test:  1459.7224788768303
Modelo Optimo
Mean Squared Error Training:  87.82765616177105
Mean Squared Error Validation:  73.54517172970229
Mean Squared Error Test:  75.09330504002091
```

El modelo 1 tiene un overfit ya que entrenamos con la mayoria de datos, esto genera un sesgo ya que es el modelotermina siendo muy ajustado solo para coincidir con los datos entrenados. Tambien podemos identificar este overfit con los modelos entrenados con muchos datos, como decia arriba el valor que mejoraba la precision del modelo era el modelo que era entrenado con la mitad de los datos.

Con estos datos se puede confirmar que el error, en validacion, si bajo mucho con el numero de muestra, asi que con esto podemos confirmar que con un numero adecuado de muestras la prediccion puede ser mucho mejor sin necesidad de agregar todos los datos

In [ ]:
```
from google.colab import files
f = files.upload()

# Convert ipynb to html
import subprocess
file0 = list(f.keys())[0]
_ = subprocess.run(["pip", "install", "nbconvert"])
_ = subprocess.run(["jupyter", "nbconvert", file0, "--to", "html"])

# download the html
files.download(file0[:-5]+"html")
```

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving Challenge3_A00835194.ipynb to Challenge3_A00835194.ipynb
```