MachineLearning

September 13, 2024

[6]:
```
# Importar las librerías necesarias
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score

# Cargar el dataset desde la ruta especificada
file_path = '/home/zatiel/Descargas/olimpicos/Olympic_Medal_Tally_History.csv'
df = pd.read_csv(file_path)

# Mostrar las primeras filas del dataset
df.head()

# 1. Exploración de los datos
print(df.info())
print(df.describe())

# Visualización de datos faltantes
plt.figure(figsize=(10, 6))
sns.heatmap(df.isnull(), cbar=False, cmap="YlGnBu")
plt.title('Valores nulos en el dataset')
plt.show()

# 2. Preprocesamiento de los datos
df_cleaned = df.dropna(subset=['year', 'total'])  # Eliminar filas con datos␣
 ↪faltantes en estas columnas

# Asegurarse de que la columna 'year' es numérica
df_cleaned['year'] = pd.to_numeric(df_cleaned['year'], errors='coerce')

# Convertir cualquier columna categórica a variables dummy (por ejemplo,␣
 ↪'country_noc' además de 'country')
categorical_columns = ['country', 'country_noc']  # Si hay más columnas␣
 ↪categóricas, añadirlas aquí
```

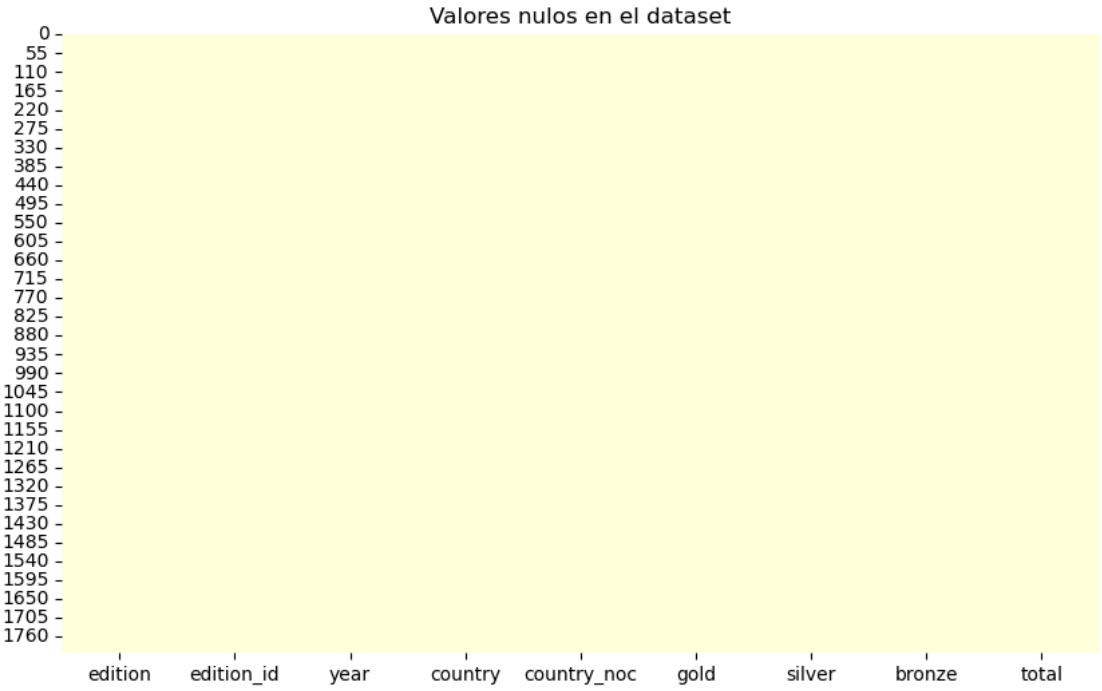
```
df_cleaned = pd.get_dummies(df_cleaned, columns=categorical_columns,␣
 ↪drop_first=True)

# Verificar si hay NaN en los datos después de las transformaciones
print("Valores NaN después de la conversión a numérico y dummies:")
print(df_cleaned.isnull().sum())

# Eliminar filas con NaN (si quedan)
df_cleaned = df_cleaned.dropna()

# Verificación de datos y dimensiones de las variables predictoras
X = df_cleaned[['year'] + [col for col in df_cleaned.columns if col.
 ↪startswith('country_') or col.startswith('country_noc_')]]  # Variables␣
 ↪predictoras
y = df_cleaned['total']  # Variable objetivo

print(f'Tamaño de X: {X.shape}')
print(f'Tamaño de y: {y.shape}')

# Dividir el dataset en conjunto de entrenamiento y prueba
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=42)

# Verificar las dimensiones después de la división
print(f'Tamaño de X_train: {X_train.shape}')
print(f'Tamaño de y_train: {y_train.shape}')

# Asegurarse de que X_train y y_train no contienen valores NaN
print("Valores NaN en X_train:", X_train.isnull().sum().sum())
print("Valores NaN en y_train:", y_train.isnull().sum())

# Asegurarse de que X_train solo contiene valores numéricos
print("Tipos de datos en X_train:")
print(X_train.dtypes)

# 3. Entrenar el modelo de regresión lineal
try:
    model = LinearRegression()
    model.fit(X_train, y_train)

    # Predicciones del modelo
    y_pred = model.predict(X_test)

    # 4. Evaluación del modelo
    mse = mean_squared_error(y_test, y_pred)
    r2 = r2_score(y_test, y_pred)
    print(f'Error Cuadrático Medio (MSE): {mse}')
```

```
    print(f'R^2 Score: {r2}')

    # 5. Visualización de los resultados

    # Gráfico de dispersión entre el año y el total de medallas predichas
    plt.figure(figsize=(10, 6))
    plt.scatter(X_test['year'], y_test, color='blue', label='Medallas Reales')
    plt.scatter(X_test['year'], y_pred, color='red', label='Medallas␣
 ↪Predichas', alpha=0.6)
    plt.xlabel('Año')
    plt.ylabel('Total de Medallas')
    plt.title('Total de Medallas Reales vs Predichas')
    plt.legend()
    plt.show()

    # Gráfico de distribución de errores
    plt.figure(figsize=(10, 6))
    sns.histplot(y_test - y_pred, bins=30, kde=True)
    plt.title('Distribución de los errores (y_test - y_pred)')
    plt.show()
except ValueError as e:
    print(f"Error al entrenar el modelo: {e}")
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1807 entries, 0 to 1806
Data columns (total 9 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   edition      1807 non-null   object
 1   edition_id   1807 non-null   int64
 2   year         1807 non-null   int64
 3   country      1807 non-null   object
 4   country_noc  1807 non-null   object
 5   gold         1807 non-null   int64
 6   silver       1807 non-null   int64
 7   bronze       1807 non-null   int64
 8   total        1807 non-null   int64
dtypes: int64(6), object(3)
memory usage: 127.2+ KB
None
        edition_id         year         gold       silver       bronze  \
count  1807.000000  1807.000000  1807.000000  1807.000000  1807.000000
mean     31.635307  1979.744328     3.737133     3.721638     3.971223
std      18.472012    32.726372     7.554092     6.411636     6.169554
min       1.000000  1896.000000     0.000000     0.000000     0.000000
25%      17.000000  1960.000000     0.000000     1.000000     1.000000
50%      25.000000  1988.000000     1.000000     2.000000     2.000000
```

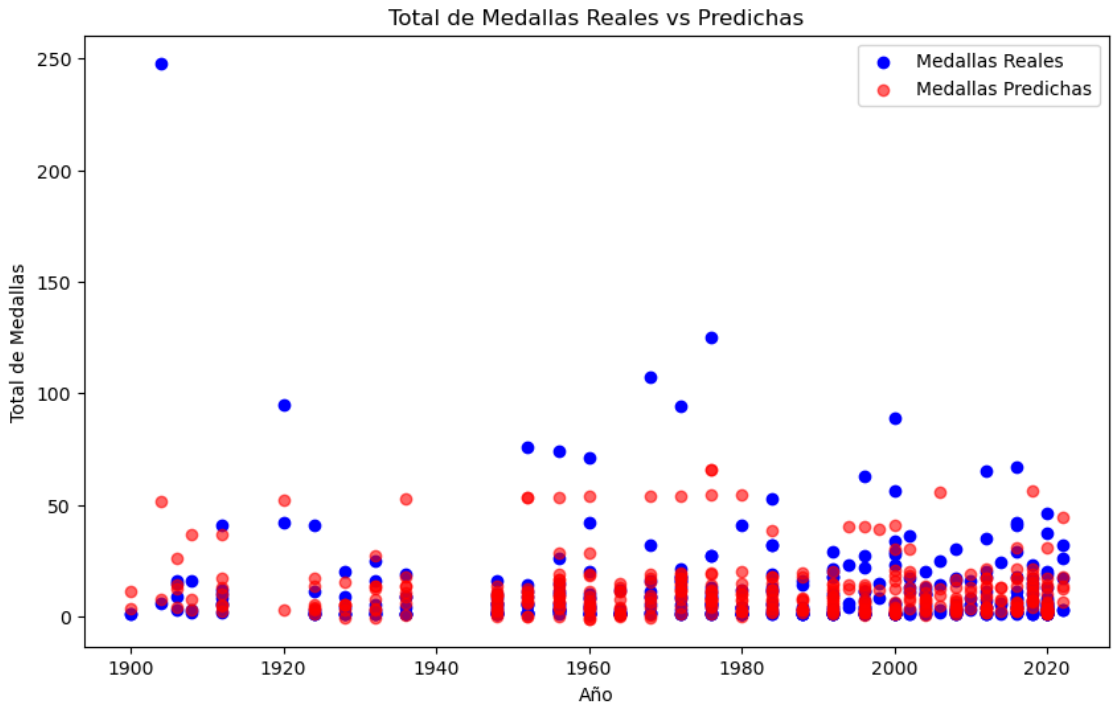
```
75%      53.000000  2008.000000     4.000000     4.000000     5.000000
max      62.000000  2022.000000    83.000000    85.000000    83.000000
```

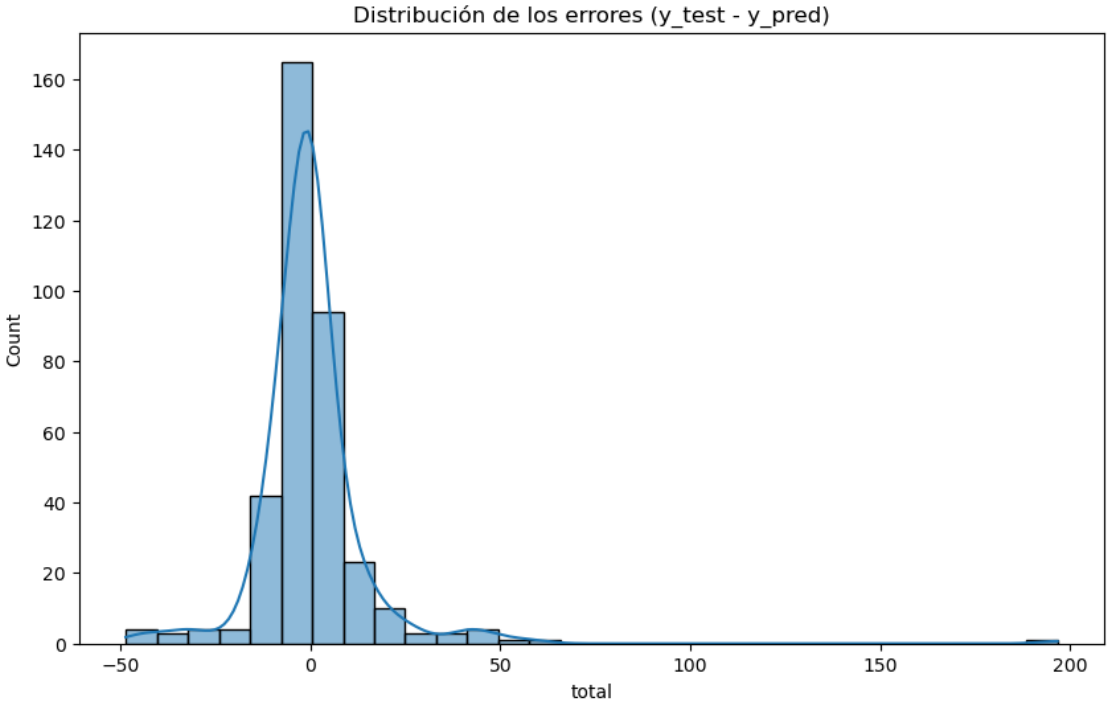
```
              total
count  1807.000000
mean     11.429994
std      19.423201
min       1.000000
25%       2.000000
50%       5.000000
75%      13.000000
max     248.000000
```




```
Valores NaN después de la conversión a numérico y dummies:
edition            0
edition_id         0
year               0
gold               0
silver             0
                  ..
country_noc_VIE    0
country_noc_WIF    0
country_noc_YUG    0
country_noc_ZAM    0
country_noc_ZIM    0
```

```
Length: 313, dtype: int64
Tamaño de X: (1807, 307)
Tamaño de y: (1807,)
Tamaño de X_train: (1445, 307)
Tamaño de y_train: (1445,)
Valores NaN en X_train: 0
Valores NaN en y_train: 0
Tipos de datos en X_train:
year                    int64
country_Algeria          bool
country_Argentina        bool
country_Armenia          bool
country_Australasia      bool
                         ...
country_noc_VIE          bool
country_noc_WIF          bool
country_noc_YUG          bool
country_noc_ZAM          bool
country_noc_ZIM          bool
Length: 307, dtype: object
Error Cuadrático Medio (MSE): 258.10075358033373
R^2 Score: 0.4041757509257128
```




Distribución de los errores (y_test - y_pred)
Count
total
0
20
40
60
80
100
120
140
160
−50
0
50
100
150
200

[ ]: