Prueba Intertrimestral

Nombre:

Apellidos:

Tiempo de la prueba: 2 Horas

Asignatura: Desarrollo de Aplicaciones para la Visualización de Datos

Fecha: 18 de octubre de 2023

Instrucciones:

- Escribe código limpio y autoexplicativo.
- Se eliminará 0.5 puntos por usar Seaborn o Matplotlib.
- Se pueden utilizar los materiales de clase.
- Se puede utilizar internet para búsqueda de dudas y documentación.
- No se puede utilizar ningún tipo de LLM.
- No se puede utilizar mensajería instantánea.
- Sube tus resultados a tu repositorio de Github.
- Imprime una versión en PDF en A3 y Portrait del notebook.
- Envialo tus resultados a dmartincorral@icai.comillas.edu (mailto:dmartincorral@icai.comillas.edu) adjuntando el PDF y la url del notebook subido al repositorio de Github.

Inicialización de librerías

Carga aquí todas las librerías que vayas a utilizar.

In [109]:
```
import pandas as pd
import numpy as np


from sklearn.datasets import fetch_california_housing
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split


# Regresión
from sklearn.linear_model import ElasticNet
from sklearn.metrics import (
    mean_squared_error,
    mean_absolute_error,
    mean_absolute_percentage_error)



# Random Forest
from sklearn.ensemble import RandomForestRegressor



import plotly.graph_objects as go
import plotly.express as px
import statsmodels.api as sm
```

Ejercicio 1 (2 puntos):

a) Crea una función que calcule y devuelva el factorial de un número entero. **(0.6 puntos)**

b) Crea una función que verifique si un número es primo o no. **(0.6 puntos)**

c) Muestra en un dataframe los 50 primeros números positivos, si es primo y su factorial utilizando las funciones anteriores. **(0.6 puntos)**

d) ¿Cómo se podría programar en una clase las tres operaciones anteriores? **(0.2 puntos)**

In [4]:
```
def factorial(n):
    if n <= 0:
        return 1
    factorial = 1
    while n > 0:
        factorial = factorial * n
        n -= 1
    return factorial

factorial(5)
```

In [7]:
```
def is_primo(n):
  for i in range(2,n):
    if (n%i) == 0:
      return False
  return True

print(is_primo(73))
print(is_primo(72))
```

```
True
False
```

In [23]:
```
df = pd.DataFrame(columns=["Numero", "Factorial", "Primo"])

df["Numero"] = np.linspace(1, 50, 50)


df["Factorial"] =df["Numero"].map(lambda x: factorial(x))
df["Primo"] = df["Numero"].map(lambda x: is_primo(int(x)))

df.head(20)
```

Out[23]:

	Numero	Factorial	Primo
0	1.0	1.000000e+00	True
1	2.0	2.000000e+00	True
2	3.0	6.000000e+00	True
3	4.0	2.400000e+01	False
4	5.0	1.200000e+02	True
5	6.0	7.200000e+02	False
6	7.0	5.040000e+03	True
7	8.0	4.032000e+04	False
8	9.0	3.628800e+05	False
9	10.0	3.628800e+06	False
10	11.0	3.991680e+07	True
11	12.0	4.790016e+08	False
12	13.0	6.227021e+09	True
13	14.0	8.717829e+10	False
14	15.0	1.307674e+12	False
15	16.0	2.092279e+13	False
16	17.0	3.556874e+14	True
17	18.0	6.402374e+15	False
18	19.0	1.216451e+17	True
19	20.0	2.432902e+18	False

In [37]:
```
class my_df():
    def __init__(self):
        self.dataframe = pd.DataFrame(columns=["Numero", "Factorial", "Primo"])

    def factorial(self, n):
        if n <= 0:
            return 1
        factorial = 1
        while n > 0:
            factorial = factorial * n
            n -= 1
        return factorial


    def is_primo(self, n):
      for i in range(2,n):
        if (n%i) == 0:
          return False
      return True

    def create_dataframe(self):
        df = pd.DataFrame(columns=["Numero", "Factorial", "Primo"])
        df["Numero"] = np.linspace(1, 50, 50)


        df["Factorial"] =df["Numero"].map(lambda x: factorial(x))
        df["Primo"] = df["Numero"].map(lambda x: is_primo(int(x)))
        self.dataframe = df


    def see_dataframe(self):
        print(self.dataframe.head(20))
```

In [38]:
```
df = my_df()
df.see_dataframe()
```

```
Empty DataFrame
Columns: [Numero, Factorial, Primo]
Index: []
```

Ejercicio 2 (4 puntos):

a) Extrae de sklearn el conjunto de datos **California Housing dataset** y transfórmalo a dataframe de pandas **(0.25 puntos)**

b) Construye una función que muestra la estructura del dataset, el número de NAs, tipos de variables y estadísticas básicas de cada una de las variables. **(0.5 puntos)**

c) Construye una **Regresión lineal** y un **Random forest** que predigan el **Median house value** según los datos disponibles. **(0.75 puntos)**

d) Visualiza cuales son las variables (coeficientes) más importantes en cada uno de los modelos. **(1.25 puntos)**

e) Decide a través de las métricas que consideres oportunas, cuál de los dos modelos es mejor, por qué y explica el proceso que has realizado para responder en los puntos anteriores. **(1.25 puntos)**

A. Extraer Datos

In [57]:
```
dataset = fetch_california_housing()
df_data = pd.DataFrame(data = dataset["data"], columns= dataset["feature_names"])
df_target = pd.DataFrame(data = dataset["target"], columns = dataset["target_names"])
```

B. Mostrar Datos

In [96]:
```
#### Checkear datos

def check_data(df_data):
    print("These are the types of the variables:")
    print(df_data.dtypes)

    print("\n These are the NAs in the Datasets")
    print(df_data.isna().sum())

    print("\n This a preview of the Dataset")
    print(df_data.head(10))

    print("\n Here we describe the variables of the dataset")
    print(df_data.describe())
    ## target
    print("\n Here we count the distribution of the target variable")
    print(df_target.value_counts())

check_data(df_data)
```

```
These are the types of the variables:
MedInc        float64
HouseAge      float64
AveRooms      float64
AveBedrms     float64
Population    float64
AveOccup      float64
Latitude      float64
Longitude     float64
dtype: object

 These are the NAs in the Datasets
MedInc        0
HouseAge      0
AveRooms      0
AveBedrms     0
Population    0
AveOccup      0
Latitude      0
Longitude     0
dtype: int64

 This a preview of the Dataset
   MedInc  HouseAge  AveRooms  AveBedrms  Population  AveOccup  Latitude  \
0  8.3252      41.0  6.984127   1.023810       322.0  2.555556     37.88
1  8.3014      21.0  6.238137   0.971880      2401.0  2.109842     37.86
2  7.2574      52.0  8.288136   1.073446       496.0  2.802260     37.85
3  5.6431      52.0  5.817352   1.073059       558.0  2.547945     37.85
4  3.8462      52.0  6.281853   1.081081       565.0  2.181467     37.85
5  4.0368      52.0  4.761658   1.103627       413.0  2.139896     37.85
6  3.6591      52.0  4.931907   0.951362      1094.0  2.128405     37.84
7  3.1200      52.0  4.797527   1.061824      1157.0  1.788253     37.84
8  2.0804      42.0  4.294118   1.117647      1206.0  2.026891     37.84
9  3.6912      52.0  4.970588   0.990196      1551.0  2.172269     37.84

   Longitude
0    -122.23
1    -122.22
2    -122.24
```

```
3    -122.25
4    -122.25
5    -122.25
6    -122.25
7    -122.25
8    -122.26
9    -122.25

 Here we describe the variables of the dataset
             MedInc      HouseAge      AveRooms     AveBedrms    Population  \
count  20640.000000  20640.000000  20640.000000  20640.000000  20640.000000
mean       3.870671     28.639486      5.429000      1.096675   1425.476744
std        1.899822     12.585558      2.474173      0.473911   1132.462122
min        0.499900      1.000000      0.846154      0.333333      3.000000
25%        2.563400     18.000000      4.440716      1.006079    787.000000
50%        3.534800     29.000000      5.229129      1.048780   1166.000000
75%        4.743250     37.000000      6.052381      1.099526   1725.000000
max       15.000100     52.000000    141.909091     34.066667  35682.000000

            AveOccup      Latitude     Longitude
count  20640.000000  20640.000000  20640.000000
mean       3.070655     35.631861   -119.569704
std       10.386050      2.135952      2.003532
min        0.692308     32.540000   -124.350000
25%        2.429741     33.930000   -121.800000
50%        2.818116     34.260000   -118.490000
75%        3.282261     37.710000   -118.010000
max     1243.333333     41.950000   -114.310000

 Here we count the distribution of the target variable
MedHouseVal
5.00001        965
1.37500        122
1.62500        117
1.12500        103
1.87500         93
              ...
3.83000           1
3.83100           1
3.83200           1
0.36600           1
0.56300           1
Length: 3842, dtype: int64
```

C y D. Modelos

In [60]:
```
# Dividir train-test
X = dataset["data"].copy()
y = dataset["target"].copy()
X_train, X_test, y_train, y_test = train_test_split(X, y ,test_size = 0.3, random_state = 123)
```

Regresion

In [63]:
```
# Regresion Lineal
reg = ElasticNet()
reg.fit(X_train,y_train)

predictions = reg.predict(X_test)
predictions_train = reg.predict(X_train)
```

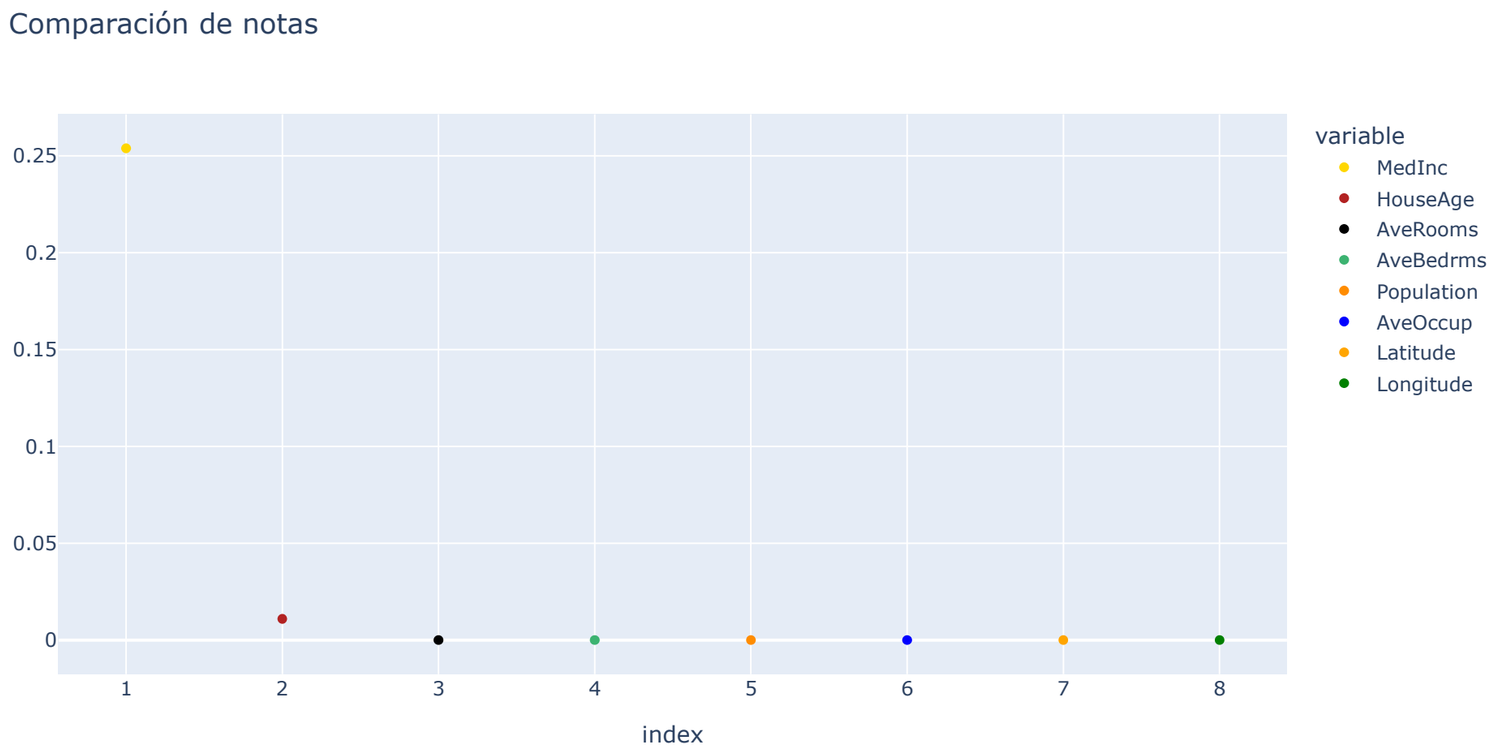
In [64]:
```
# Metricas de evaluación
rmse_train = np.sqrt(mean_squared_error(y_train,predictions_train))
mae_train = mean_absolute_error(y_train, predictions_train)
mape_train = mean_absolute_percentage_error(y_train, predictions_train)

rmse_test = np.sqrt(mean_squared_error(y_test,predictions))
mae_test = mean_absolute_error(y_test, predictions)
mape_test = mean_absolute_percentage_error(y_test, predictions)

print("El RMSE de train del modelo es: {}".format(rmse_train))
print(f"El MAE de train del modelo es: {mae_train}")
print(f"El MAPE de train del modelo es: {100 * mape_train} %")

print("")

print("El RMSE de test del modelo es: {}".format(rmse_test))
print(f"El MAE de test del modelo es: {mae_test}")
print(f"El MAPE de test del modelo es: {100*mape_test} %")
```

```
El RMSE de train del modelo es: 0.877980259023968
El MAE de train del modelo es: 0.6807095985024406
El MAPE de train del modelo es: 45.20172523874712 %

El RMSE de test del modelo es: 0.8720194772729227
El MAE de test del modelo es: 0.6785162737312243
El MAPE de test del modelo es: 46.262056633536005 %
```

In [133]:
```
print(df_data.columns)
print(reg.coef_)
```

```
Index(['MedInc', 'HouseAge', 'AveRooms', 'AveBedrms', 'Population', 'AveOccup',
       'Latitude', 'Longitude'],
      dtype='object')
[ 2.53912734e-01  1.09735267e-02  0.00000000e+00 -0.00000000e+00
  1.11830749e-05 -0.00000000e+00 -0.00000000e+00 -0.00000000e+00]
```

In [134]:
```
# Existe alguna correlación entre las notas de matemáticas y escritura
my_df = pd.DataFrame(columns = ["index","variable", "coef"])
my_df["index"] = [1, 2, 3, 4, 5 ,6 ,7 ,8]
my_df["variable"] = df_data.columns
my_df["coef"] = my_coef

fig = px.scatter(my_df, x="index", y="coef", color="variable", title = "Comparación de notas",
                 color_discrete_map={"MedInc": "gold", "HouseAge": "firebrick",
                                     "AveRooms": "black", "AveBedrms": "mediumseagreen",
                                     "Population": "darkorange", "AveOccup": "blue", "Latitude": "orange",
                                     "Longitude": "green"},
                 trendline="ols")
fig.show()
```



Aquí podemos ver que la variable que tiene impacto en el modelo es MedInc, es decir, la renta media. La segunda variable que más impacta que es la edad del inmueble tiene un impacto mucho menor que MedInc.

In [ ]:
```
# Create a variable for the best model
best_rf = rand_search.best_estimator_

# Print the best hyperparameters
print('Best hyperparameters:',  rand_search.best_params_)
```

Random Forest

In [73]:
```
rf = RandomForestRegressor()
rf.fit(X_train, y_train)

predictions = rf.predict(X_test)
predictions_train = rf.predict(X_train)
```

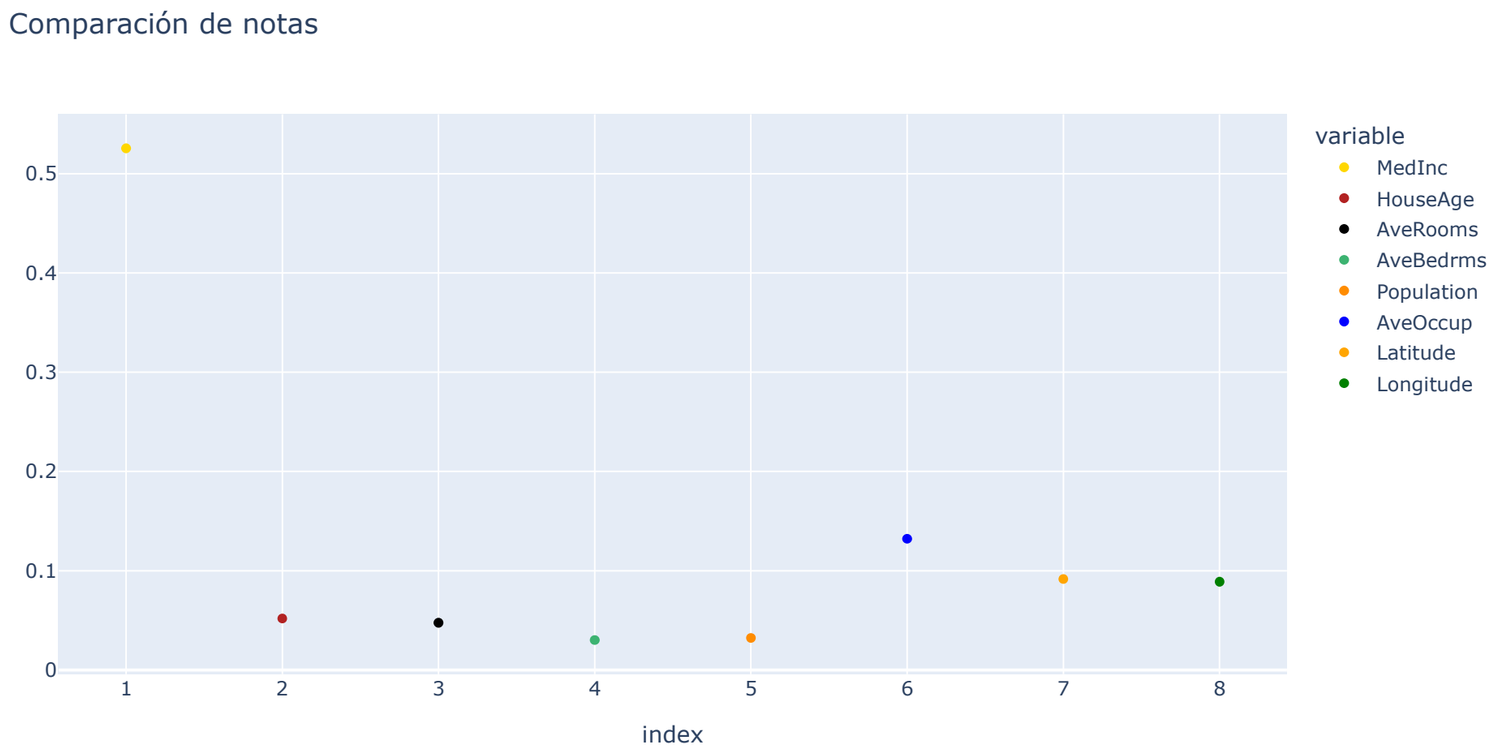
In [74]:
```
# Metricas de evaluación
rmse_train = np.sqrt(mean_squared_error(y_train,predictions_train))
mae_train = mean_absolute_error(y_train, predictions_train)
mape_train = mean_absolute_percentage_error(y_train, predictions_train)

rmse_test = np.sqrt(mean_squared_error(y_test,predictions))
mae_test = mean_absolute_error(y_test, predictions)
mape_test = mean_absolute_percentage_error(y_test, predictions)

print("El RMSE de train del modelo es: {}".format(rmse_train))
print(f"El MAE de train del modelo es: {mae_train}")
print(f"El MAPE de train del modelo es: {100 * mape_train} %")

print("")

print("El RMSE de test del modelo es: {}".format(rmse_test))
print(f"El MAE de test del modelo es: {mae_test}")
print(f"El MAPE de test del modelo es: {100*mape_test} %")
```

```
El RMSE de train del modelo es: 0.18964018882828046
El MAE de train del modelo es: 0.12404517482004468
El MAPE de train del modelo es: 6.961807918946687 %

El RMSE de test del modelo es: 0.4966975488103967
El MAE de test del modelo es: 0.325577445268088
El MAPE de test del modelo es: 18.62177836418584 %
```

In [135]:
```
rf.feature_importances_
```

Out[135]:
```
array([0.52557323, 0.05183313, 0.04757895, 0.03010838, 0.0322937 ,
       0.13215042, 0.09158491, 0.08887729])
```

In [136]:
```
# Existe alguna correlación entre las notas de matemáticas y escritura
my_df = pd.DataFrame(columns = ["index","variable", "coef"])
my_df["index"] = [1, 2, 3, 4, 5 ,6 ,7 ,8]
my_df["variable"] = df_data.columns
my_df["coef"] = rf.feature_importances_

fig = px.scatter(my_df, x="index", y="coef", color="variable", title = "Comparación de notas",
                 color_discrete_map={"MedInc": "gold", "HouseAge": "firebrick",
                                     "AveRooms": "black", "AveBedrms": "mediumseagreen",
                                     "Population": "darkorange", "AveOccup": "blue", "Latitude": "orange",
                                     "Longitude": "green"},
                 trendline="ols")
fig.show()
```



Aquí podemos ver que la variable MedInc vuelve a ser la más importante, pero, las últimas tres variables (Ocupación y el lugar geográfico tienen cierto impacto ~ 5 veces menor que MedInc)

E. Evaluación

Aunque el random forest parece sacar mayor provecho a las variables para explicar el comportamiento, se puede por las métricas que este funciona mucho peor que el modelo de regresión.

- El RMSE del modelo de regresión es del 87% en test
- El RMSE del modelo random forest es del 50% en test

El primer paso que hemos realizado ha sido extraer los datos y crear un dataframe. Segundo hemos sacado las métricas de los datos para estudiar las variables. Aquí hemos visto que todas las variables eran numericas continuas y que, efectivamente, debíamos usar modelos de regresión. Tercero, hemos entrenado dos modelos de regresión y random forest. Cuarto hemos comparado la importancia de los coeficientes y la metrica

MSRE dónde hemos visto un desempeño mejor en el modelo regresivo

Ejercicio 3 (4 puntos):

Consideremos el dataset que contiene **The Most Streamed Spotify Songs 2023** que se encuentra en el respositorio.

Información de las variables:

- track_name: Name of the song
- artist(s)_name: Name of the artist(s) of the song
- vartist_count: Number of artists contributing to the song
- released_year: Year when the song was released
- released_month: Month when the song was released
- release_day: Day of the month when the song was released
- in_spotify_playlists: Number of Spotify playlists the song is included in
- in_spotify_charts: Presence and rank of the song on Spotify charts
- streams: Total number of streams on Spotify
- in_apple_playlists: Number of Apple Music playlists the song is included in
- in_apple_charts: Presence and rank of the song on Apple Music charts
- in_deezer_playlists: Number of Deezer playlists the song is included in
- in_deezer_charts: Presence and rank of the song on Deezer charts
- in_shazam_charts: Presence and rank of the song on Shazam charts
- bpm: Beats per minute, a measure of song tempo
- key: Key of the song
- mode: Mode of the song (major or minor)
- danceability_%: Percentage indicating how suitable the song is for dancing
- valence_%: Positivity of the song's musical content
- energy_%: Perceived energy level of the song
- acousticness_%: Amount of acoustic sound in the song
- instrumentalness_%: Amount of instrumental content in the song
- liveness_%: Presence of live performance elements
- speechiness_%: Amount of spoken words in the song

Para las respuestas b, c, d, e, f y g es imperativo acompañarlas respuestas con una visualización.

a) Lee el fichero en formato dataframe, aplica la función del ejercicio 2.b, elimina NAs y convierte a integer si fuera necesario. **(0.25 puntos)**

b) ¿Cuántos artistas únicos hay? **(0.25 puntos)**

c) ¿Cuál es la distribución de reproducciones? **(0.5 puntos)**

d) ¿Existe una diferencia signitificativa en las reproducciones entre las canciones de un solo artista y las de más de uno? **(0.5 puntos)**

e) ¿Cuáles son las propiedades de una canción que mejor correlan con el número de reproducciones de una canción? **(0.5 puntos)**

f) ¿Cuáles son las variables que mejor predicen las canciones que están por encima el percentil 50? **(1 puntos)**

Nota: Crea una variable binaria (Hit/No Hit) en base a 3.c, crea una regresión logística y visualiza sus coeficientes.

g) Agrupa los 4 gráficos realizados en uno solo y haz una recomendación a un sello discográfico para producir un nuevo hit. **(1 puntos)**

A.

In [89]:
```
df = pd.read_csv("spotify-2023.csv", encoding = 'iso-8859-1')
df.head()
```

Out[89]:

	track_name	artist(s)_name	artist_count	released_year	released_month	released_day	in_spotify_playlists	in_spotify_charts	streams	in_apple_playlists	...	bpm	key	mode	danceability_%	valence_%	energy_%	ac
0	Seven (feat. Latto) (Explicit Ver.)	Latto, Jung Kook	2	2023	7	14	553	147	141381703	43	...	125	B	Major	80	89	83	
1	LALA	Myke Towers	1	2023	3	23	1474	48	133716286	48	...	92	C#	Major	71	61	74	
2	vampire	Olivia Rodrigo	1	2023	6	30	1397	113	140003974	94	...	138	F	Major	51	32	53	
3	Cruel Summer	Taylor Swift	1	2019	8	23	7858	100	800840817	116	...	170	A	Major	55	58	72	
4	WHERE SHE GOES	Bad Bunny	1	2023	5	18	3133	50	303236322	84	...	144	A	Minor	65	23	80	

5 rows × 24 columns

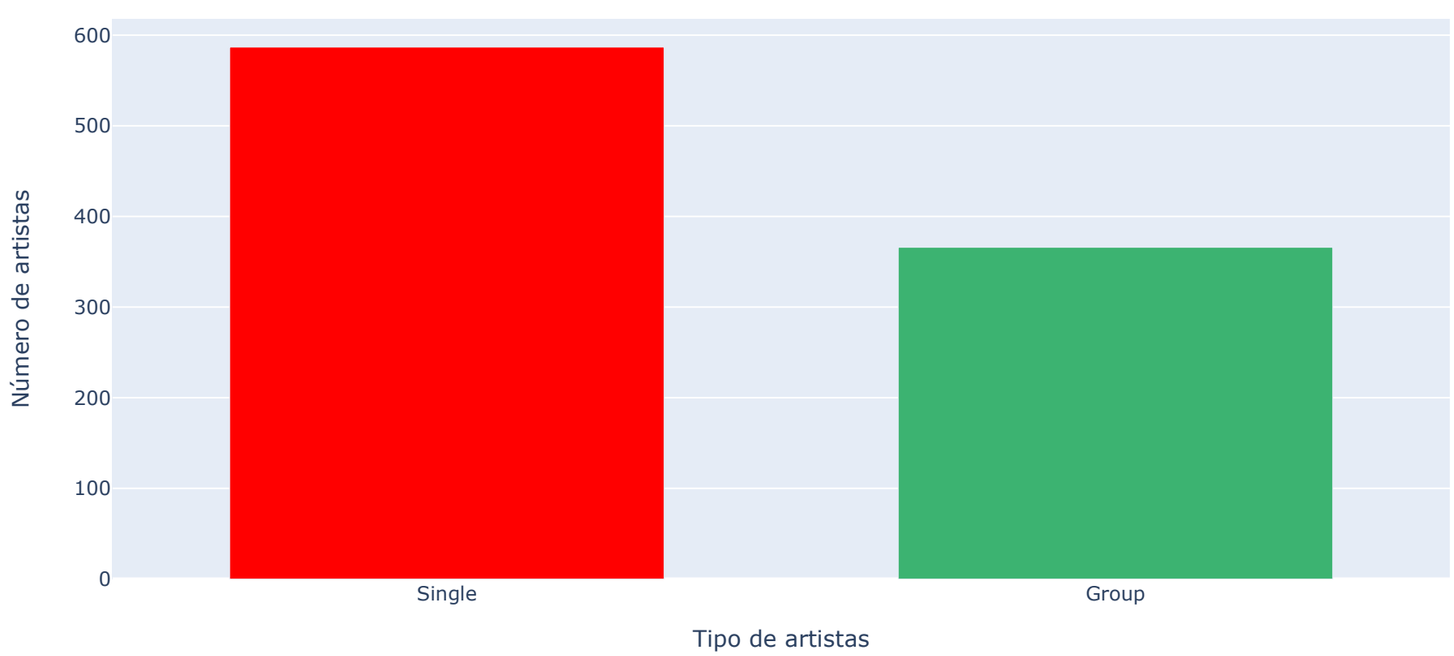
In [97]:
```
check_data(df)
```

```
These are the types of the variables:
track_name              object
artist(s)_name          object
artist_count             int64
released_year            int64
released_month           int64
released_day             int64
in_spotify_playlists     int64
in_spotify_charts        int64
streams                 object
in_apple_playlists       int64
in_apple_charts          int64
in_deezer_playlists     object
in_deezer_charts         int64
in_shazam_charts        object
bpm                      int64
key                     object
mode                    object
danceability_%           int64
```

B.

In [111]:
```
df["artist_count"].value_counts()
single_artist = df[df["artist_count"] == 1]
group_artist = df[df["artist_count"] > 1]
df["artist_group"] = df["artist_count"].map(lambda x: "Single" if x == 1 else "Group")
```

In [113]:
```
# Cómo es la distribución de la educacion de los padres los padres?
level_count = df["artist_group"].value_counts()

data = [
    go.Bar(
        x = level_count.index,
        y = level_count,
        name = "Artist Group",
        marker_color = ["red","mediumseagreen"],
        width= np.repeat(0.65,len(level_count)) # Cambiar la anchura de cada barra
    )
]

layout = go.Layout(title = "Numero de artistas individuales y grupales", xaxis_title = "Tipo de artistas",
                   yaxis_title = "Número de artistas",
                  )

fig = go.Figure(data = data, layout = layout)

fig.show()

level_count
```



Out[113]:
```
Single    587
Group     366
Name: artist_group, dtype: int64
```

C.

In [ ]:

In [ ]: