Universidad Peruana de Ciencias Aplicadas

UPC

TF

Administración De La Información

Profesora: Patricia Daniela Reyes Silva

Ciclo: 2021 - 2

Integrantes:

- Renzo Paredes Villagra - u20191b335
- Sergio Andres Flores Ñahuis - u201915474

1. Objetivos del proyecto

En este proyecto final, se va a realizar un Análisis Exploratorio de Datos y , con ello, ofrecer una solución a un problema de Modelización de datos. En este sentido, el proyecto nos pide evaluar y predecir las tendencias de los videos de Youtube de un determinado país. Con el conocimiento generado a partir de un conjunto de datos que recopila todos los registros diarios de los videos con mayor tendencia de Youtube, se atenderá la necesidad de una importante empresa de marketing digital que desea responder a diversos requerimientos de información.

2. Caso de análisis

El conjunto de datos que se va a analizar se llama Trending Youtube Video Statistics y almacena una lista de los vídeos de mayor tendencia en la plataforma. Fue creado por el desarrollador de software y experto en Kaggle Mitchell J y fue actualizado por última vez el 2 de junio del 2019. Este dataset contiene diferentes tipos de información que usa Youtube para determinar los vídeos más populares del año, tales como número de visitas, veces que fueron compartidos, comentarios y likes. Además, incluye muchos meses de información de los videos más populares para distintas regiones de Estados Unidos, Gran Bretaña, Alemania, Canadá, Francia, Rusia, México, India y Corea del Sur.

Este conjunto de datos podría ser aplicado en diferentes casos, tales como categorizar videos de Youtube usando comentarios y estadística, análisis de las opiniones y reacciones de los usuarios, entrenar algoritmos de machine learning como algoritmos de redes neuronales para generar comentarios en Youtube, examinar qué factores determinan el nivel de popularidad de un video, entre otros.

3. Conjunto de datos

a. Videos:

Item	Variable	Descripción
1	**video_id**	Número ID de cada video
2	**trending_date**	Fecha en que el video tuvo gran tendencia
3	**title**	Título del video

4	**channel_title**	Título del canal que subió el video
5	**category_id**	Número ID de la categoría del video
6	**publish_time**	Hora de publicación del video
7	**tags**	Etiquetas usadas en el video
8	**views**	Número de visitas de cada video
9	**likes**	Número de likes de cada video
10	**dislikes**	Número de dislikes de cada video
11	**comment_count**	Cantidad de comentarios
12	**thumbnail_link**	Link de la imagen usada para la miniatura del video
13	**comment_disabled**	Indica si los comentarios fueron desactivados en el video
14	**ratings_disabled**	Indica si las puntuaciones fueron desactivadas en el video
15	**video_error_or_removed**	Indica si hubo un error en el video o si fue removido
16	**description**	Descripción del video
17	**state**	Estado donde el video tuvo gran tendencia
18	**lat**	Latitud geográfica de ubicación del Estado
19	**long**	Longitud geográfica de ubicación del Estado

20	**geometry**	Contiene las coordenadas de las geometrías donde se encuentra el Estado en el planeta

b. Categorías:

Item	Variable	Descripción
1	**kind**	Tipo video categoría
2	**etag**	Enlace del tag de la categoría
3	**id**	ID de la categoría
4	**snippet**	Unidad de la categoría
5	**channelId**	ID del canal de la categoría
6	**title**	Título de la categoría
7	**assignable**	Estado de asignación de la categoría

4. Análisis exploratorio de datos

4.1. Cargar los datos

Antes de empezar a desarrollar el análisis, es necesario llevar a cabo una serie de procesos que van a garantizar la completitud, veracidad y calidad de los datos que se van a usar.

En este sentido, primero, vamos a guardar todos los datos en una variable para su posterior inspección.

```
import pandas as pd

dfg = pd.read_csv('INvideos_cc50_202101.csv')

dff = pd.read_json('IN_category_id.json')
```

Gracias a las funciones read_csv y read_json que provienen de la librería Pandas, somos capaces de almacenar los dos conjuntos de datos. La variable dfg guardará todos los datos crudos sobre los videos que están en tendencia diariamente en distintas zonas de la India. La variable dff contendrá todos los datos acerca los diferentes tipos de categoría para cada video del conjunto de datos anterior.

4.1. Inspeccionar los datos

Ya que hemos asignado a dos variables los dos conjuntos de datos existentes, iniciaremos una inspección para conocer más detalles sumamente fundamentales sobre ellos. Para ello, emplearemos las siguientes pasos:

dfg.info()

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 38533 entries, 0 to 38532
Data columns (total 20 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   video_id                38528 non-null  object
 1   trending_date           37617 non-null  object
 2   title                   37535 non-null  object
 3   channel_title           37422 non-null  object
 4   category_id             37389 non-null  object
 5   publish_time            37376 non-null  object
 6   tags                    37364 non-null  object
 7   views                   37357 non-null  object
 8   likes                   37352 non-null  float64
 9   dislikes                37352 non-null  float64
 10  comment_count           37352 non-null  float64
 11  thumbnail_link          37352 non-null  object
 12  comments_disabled       37352 non-null  object
 13  ratings_disabled        37352 non-null  object
 14  video_error_or_removed  37352 non-null  object
 15  description             36791 non-null  object
 16  state                   38533 non-null  object
 17  lat                     38533 non-null  float64
 18  lon                     38533 non-null  float64
 19  geometry                38533 non-null  object
dtypes: float64(5), object(15)
memory usage: 5.9+ MB
```

La variable dfg contiene un total de 20 columnas y 38533 entradas. Podemos conocer los nombres de cada columnas, así como el tipo de variable en cada una. También se observa que, en la columna video_id, hay 38528 valores non-null, lo que significa que el dataset posee filas con valores faltantes. Existen 5 columnas pertenecientes al tipo float64 y 15 columnas al tipo object. El tipo de variable object nos indica que el tipo de dato de los valores dentro de esas columnas

pueden llegar a ser una mezclas de enteros, float y string, por lo que será necesario asignar un tipo específico a cada columna más adelante

dff.info()

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 31 entries, 0 to 30
Data columns (total 3 columns):
 #   Column  Non-Null Count  Dtype 
---  ------  --------------  ----- 
 0   kind    31 non-null     object
 1   etag    31 non-null     object
 2   items   31 non-null     object
dtypes: object(3)
memory usage: 872.0+ bytes
```

En el output, se puede observar que hay un total de 3 columnas y 31 entradas en la variable dfg. Además, se muestran los nombres de cada columna y los tipos de variable para cada una. En este caso, las 3 columnas pertenecen al tipo object.

4.1. Pre-procesar los datos

Durante la captura y recopilación de datos, es bastante común que surjan errores y problemas, tales como piezas faltantes o incorrectas, en los valores que constituyen estos conjuntos de datos. Debido a ello, los datos deben atravesar una sucesión de pasos para asegurar que estos sean íntegros y de calidad para asegurar un correcto y verídico análisis. El pre-procesamiento de los datos que hemos realizado está compuesto por las siguientes instrucciones:

Con respecto al data frame dfg, utilizamos la técnica de eliminar valores na y reseteamos los índices.

```
dfg.dropna(inplace=True)
dfg.reset_index(drop=True,inplace=True)
```

Luego creamos una función para crear un video_id.

```
import random
def create_videoid():
    list=['a','b','c','d','e','f','g','h','i','j','k','l','m','n','o','p','q','r','s','t','u','v','w','x','y','z','A'
        'H','I','J','K','L','M','N','O','P','Q','R','S','T','U','V','W','X','Y','Z','0','1','2','3','4','5','6','7','
    arr=''
    for i in range(11):
        arr+=random.choice(list)
    return arr
```

Después recorremos todo el dataframe, donde primero arreglamos los valores sin sentido de la columna video_id como #NAME?, reemplazandolos por un código de números y letras usando la función create_videoid(); luego encontramos que en el category_id hay algunos valores tipo float y para que no haya problemas hemos decidido en reemplazarlos por tipo string; luego también visualizamos un problema similar con la columna views donde decidimos reemplazar los datos que tengan tipo string a float; después agregamos el string 20 a todos los datos de trending _date para no tener problemas para transformar a formato datetime.

```
for i in range(len(dfg['video_id'])):
    if(len(dfg['video_id'][i])!=11):
        dfg['video_id'][i]=create_videoid()

    #los que son float a string
    if(type(dfg['category_id'][i])==float):
        dfg['category_id'][i]=str(dfg['category_id'][i])
        dfg['category_id'][i]=dfg['category_id'][i][:-2]

    #los que son str a float
    if(type(dfg['views'][i])==str):
        dfg['views'][i]=float(dfg['views'][i])

    dfg['trending_date'][i]='20'+dfg['trending_date'][i]
```

Y por último eliminamos los duplicados, hacemos que la columna views sea tipo np.float64, transformamos las columnas trending_date y publish_time a formato datetime.

```
dfg.drop_duplicates(ignore_index = True, inplace=True)
dfg['views']=dfg['views'].astype(np.float64)
dfg['trending_date']=pd.to_datetime(dfg['trending_date'],format='%Y.%d.%m')
dfg['publish_time']=dfg['publish_time'].apply(pd.to_datetime)
```

Con respecto al data frame dff, creamos un arreglo para que guarde todos los id de cada ítem, y con ese arreglo agregarle una columna llamada id a la variable dff.

```
ercd = []
for a in dff['items']:
    ercd.append(a['id'])

dff.assign(id = '')
dff['id'] = ercd
```

Luego hemos eliminado 2 columnas por que se repetían sus valores en todas las filas y que dentro de su columna ítems se encuentran esos datos, también hemos

agregado una nueva categoría para que no hubiese problemas con el otro data frame y por último realizamos un join entre los 2 data frames usando como referencia las columnas id y category_id, y el resultado lo hemos guardado en una variable dfc.

```
dff.drop(['kind','etag'], axis = 1, inplace = True)
dff.loc[31]=[{'kind': 'youtube#videoCategory',
 'etag': '"m2yskBQFythfE4irbTIeOgYYfBU/SalkJoBWq_smSEqiAx_qyri6Wa8"',
 'id': '29',
 'snippet': {'channelId': 'UCBR8-60-B28hp2BmDPdntcQ',
  'title': 'Nonprofits & Activism',
  'assignable': True}},'29']

dfc = dfg.join(dff.set_index('id'), on='category_id')
```

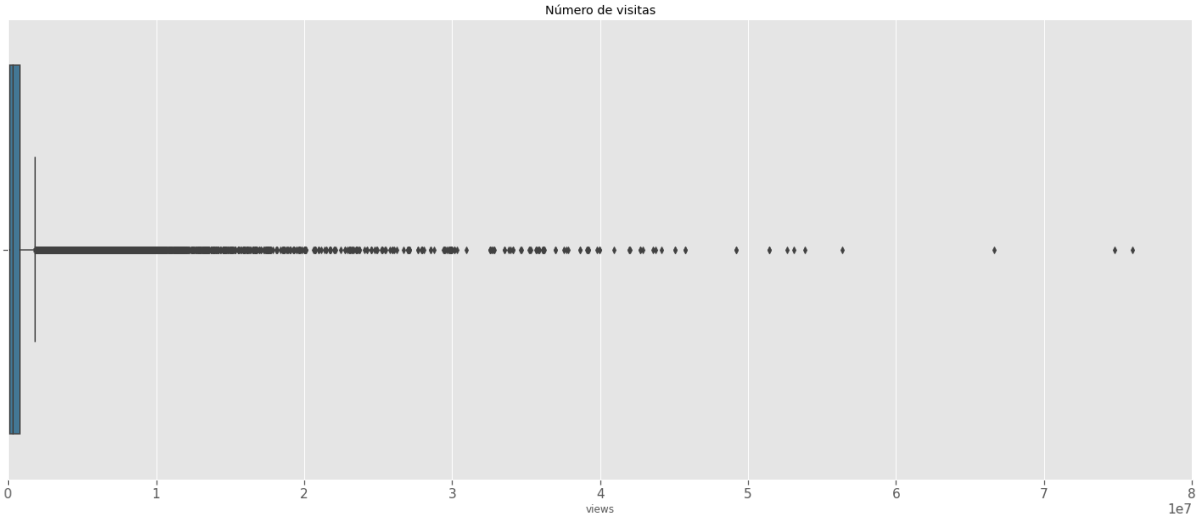
Con respecto al nuevo data frame dfc, le creamos una nueva columna llamada trending_date_month que contiene el nombre de un mes en el que cada video llegó a ser tendencia.

```
months = { 1:'Enero', 2:'Febrero', 3:'Marzo', 4:'Abril', 5:'Mayo', 6:'Junio',
          7:'Julio', 8:'Agosto', 9:'Septiembre', 10:'Octubre', 11:'Noviembre', 12:'Diciembre'}

dfc['trending_date_month'] = np.nan

for i in range(len(dfg)):
    dfc['trending_date_month'][i] = months[dfg['trending_date'][i].month]
```

Para averiguar si existen valores atípicos o no en nuestro dataset, utilizaremos los gráficos de cajas, ya que estos son ideales para mostrar todos los datos distribuidos a través de percentiles. Realizamos las siguientes instrucciones:

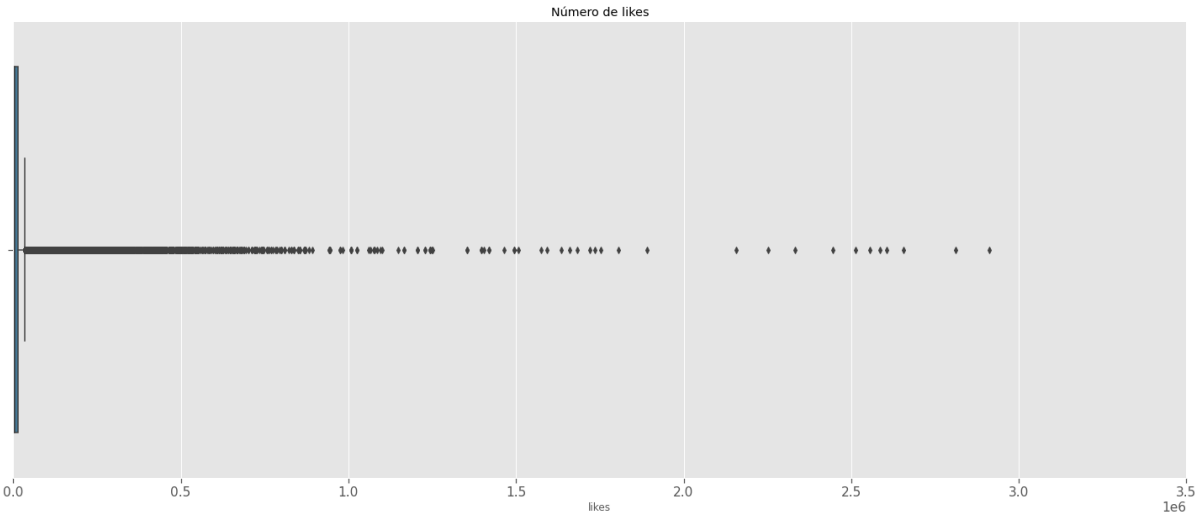
- views

```
#views
plt.style.use('ggplot')
plt.figure(figsize=(25,10))
plt.title('Número de visitas')
sns.set_context('notebook',font_scale=1.4)
ax = sns.boxplot(x='views', data=dfc, orient="h", palette='mako') #rocket
ax.set_xlim([0.0, 800*10**5])
```



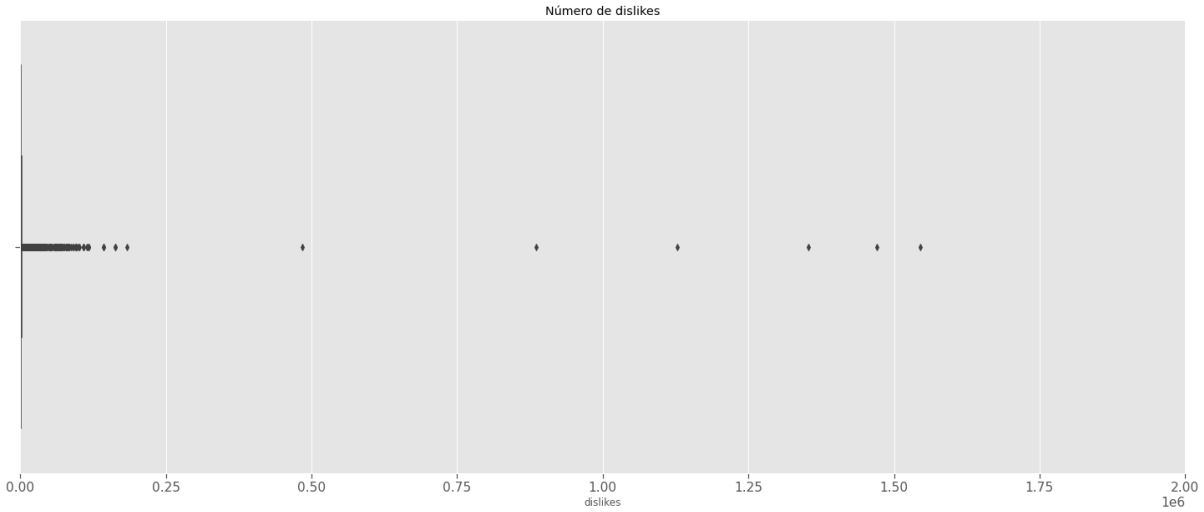
- likes

```
plt.style.use('ggplot')
plt.figure(figsize=(25,10))
plt.title('Número de likes')
sns.set_context('notebook',font_scale=1.4)
ax = sns.boxplot(x='likes', data=dfc, orient="h", palette='mako') #rocket
ax.set_xlim([0, 35*10**5])
```



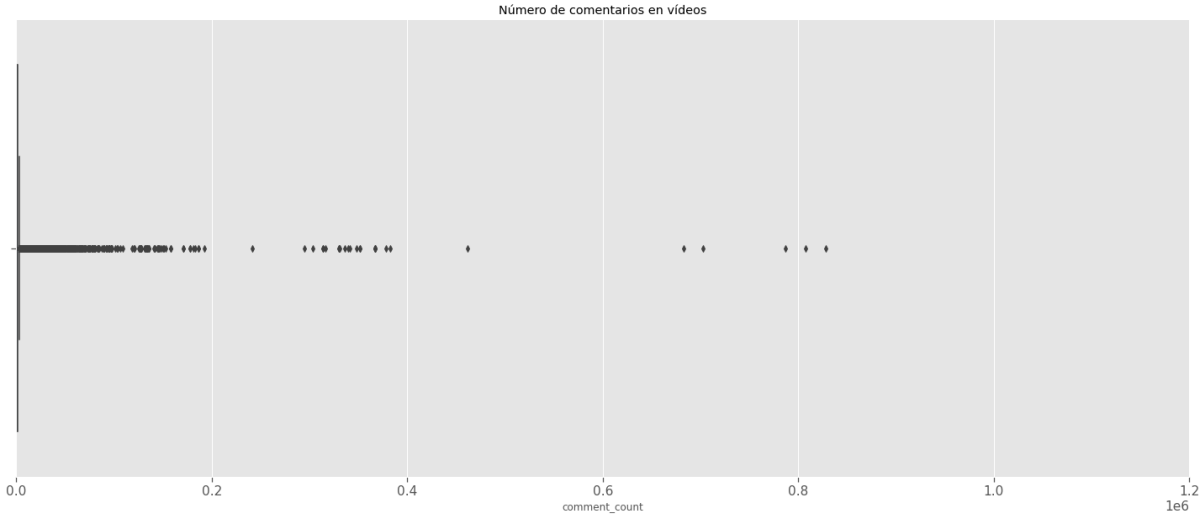
- dislikes

```
#dislikes
plt.style.use('ggplot')
plt.figure(figsize=(25,10))
plt.title('Número de dislikes')
sns.set_context('notebook',font_scale=1.4)
ax = sns.boxplot(x='dislikes', data=dfc, orient="h", palette='mako') #rocket
ax.set_xlim([0.0, 20*10**5])
```



- comment_count

```
#comment_count
plt.style.use('ggplot')
plt.figure(figsize=(25,10))
plt.title('Número de comentarios en vídeos')
sns.set_context('notebook',font_scale=1.4)
ax = sns.boxplot(x='comment_count', data=dfc, orient="h", palette='mako') #rocket
ax.set_xlim([0.0, 12*10**5])
```

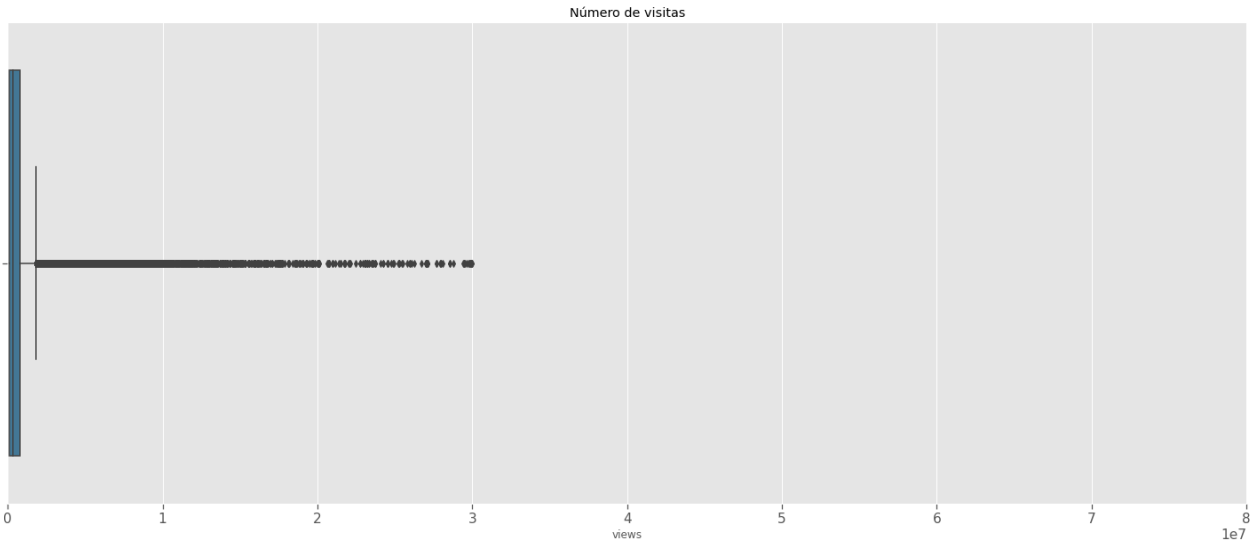


Debido a que hay una gran cantidad de valores atípicos en cada una de las columnas mostradas, no resulta muy conveniente usar el método de eliminación, así que decidimos crear una técnica que va a enmascarar ciertos outliers. Primero, creamos una copia del data frame dfc en la variable dfc2. Esta será nuestro dataset para los posteriores análisis y resolución de preguntas. Luego utilizamos nuestra técnica ya mencionada que determina que, si los datos superan un límite establecido por nosotros para cada columna, su valor se reemplazará por el valor del cap, el cual guarda el valor del percentil 95, lo que causará que, de esta manera, no se produzca una alteración muy grande en el conjunto de datos. Asimismo, los valores de las columnas indicadas que sean iguales a cero también serán reemplazados por la variable cap. Aplicando esta técnica, arreglamos los outliers y los valores que son ceros de las columnas views, likes, dislikes y comment_count.

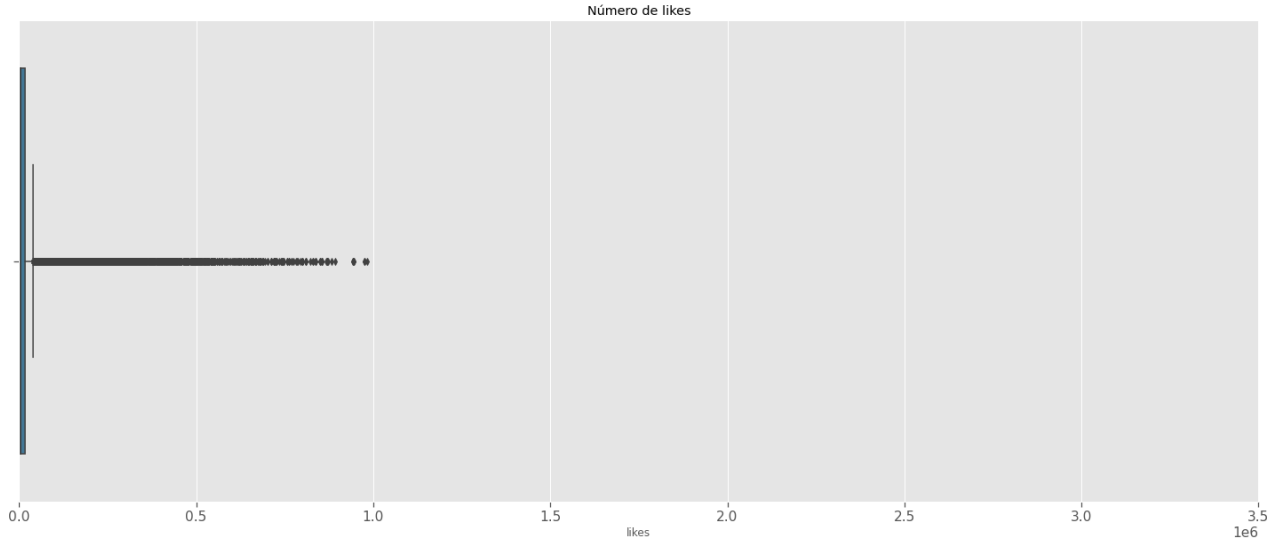
```
#arreglar valores atípicos
dfc2 = dfc.copy()
def fix_outliers(data, cols_lims):

    for col, lim in cols_lims:
        cap = data[col].quantile(.95)

        for e in range(0, len(data)):
            if data[col][e] > lim or data[col][e]==0:
                data[col][e] = cap
    return data

dfc2 = fix_outliers(dfc2, [('views', 30 * 10**6), ('likes', 10**6), ('dislikes', 0.25 * 10**6), ('comment_count', 0.2
```

Datos atípicos arreglados:

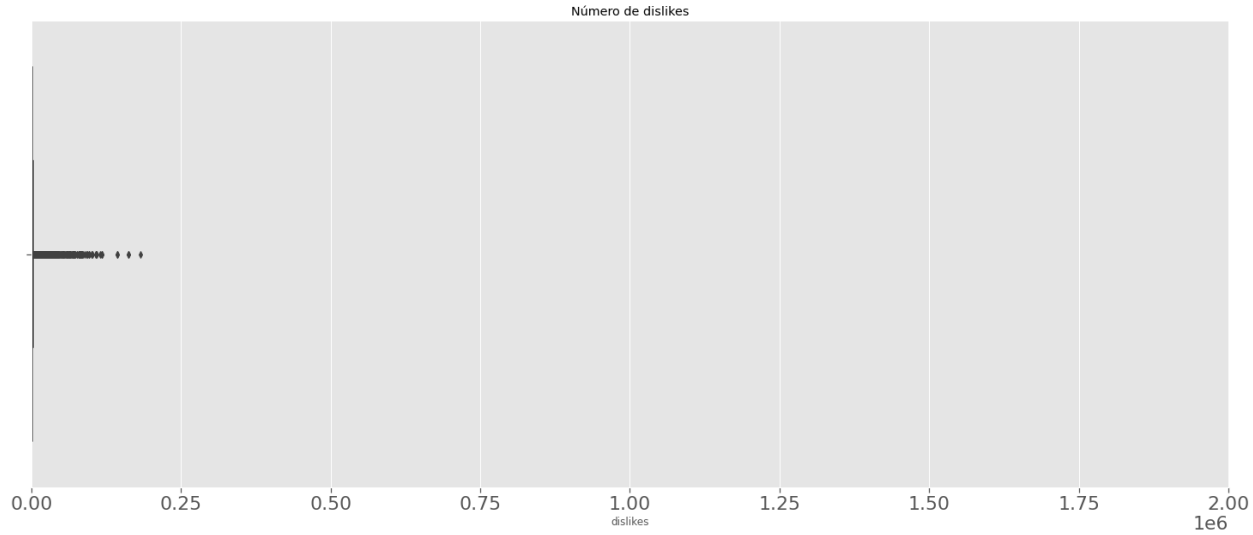
- views



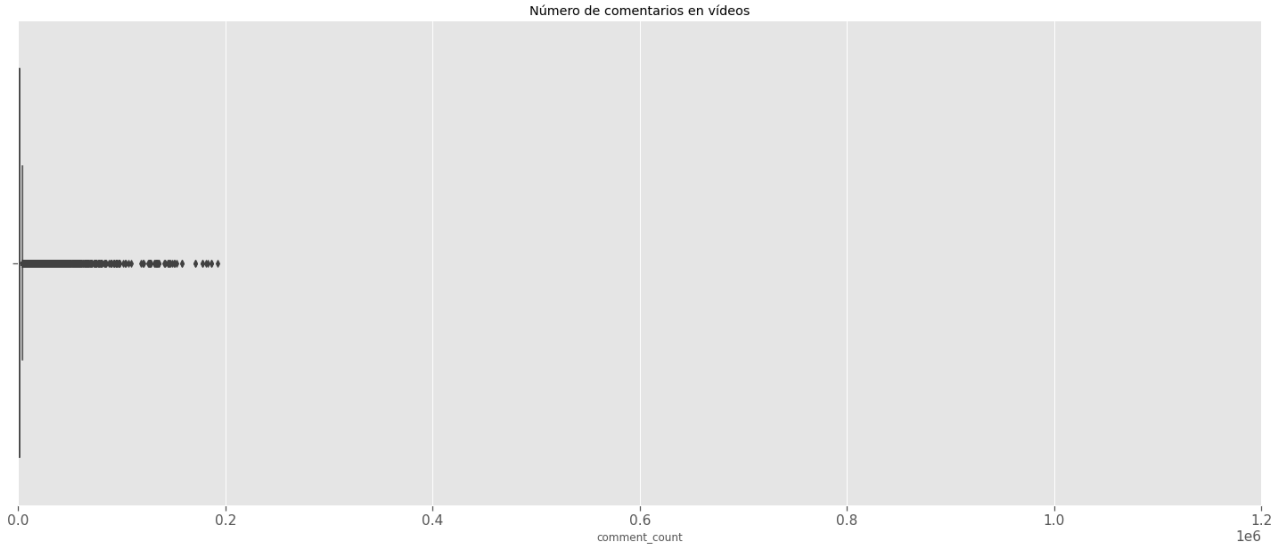
- likes



- dislikes



- comment_count



4.1. Visualizar los datos

En esta sección, comenzaremos a responder las preguntas relacionadas a varios requerimientos de información sobre los videos con mayor tendencia en India.

A. ¿Qué categorías de video son las de mayor tendencia?

Para poder calcular qué videos tienen mayor tendencia que otros, optamos por sumar la cantidad de vistas, la cantidad de likes y la cantidad de comentarios para cada video, y lo guardaremos en la variable sum_numbers.

sum_numbers = views + likes + comment_count

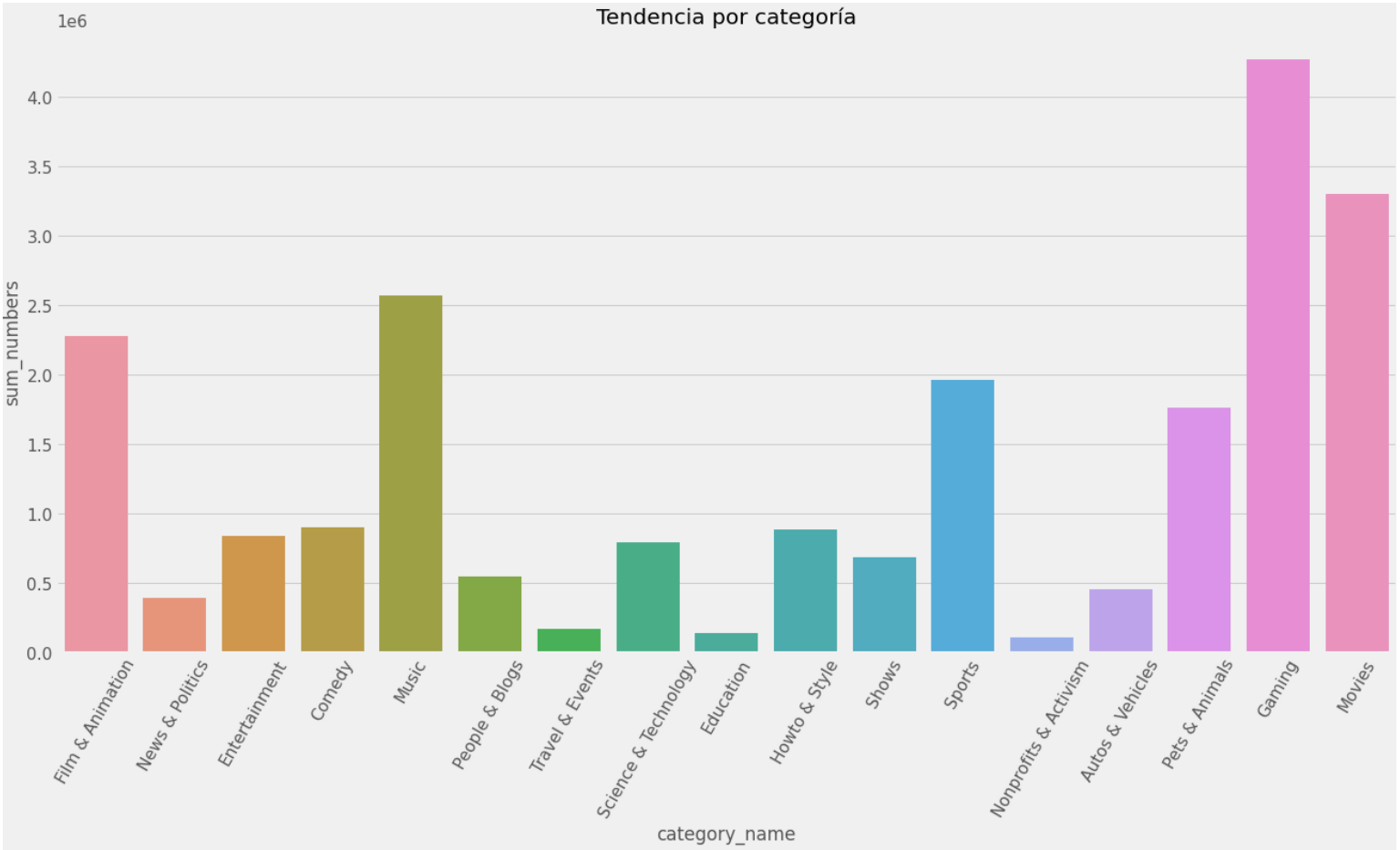
Con esto, determinaremos qué video genera una mayor interacción con los usuarios, así como el nivel de popularidad de cada categoría de video.

```
dfc2['category_name'] = np.nan
for e in range(len(dfc2)):
    dfc2['category_name'][e] = dfc2['items'][e]['snippet']['title']
```

Primero, creamos la columna category_name, que contendrá el nombre de la categoría correspondiente a cada video. Se extraerá de la columna items. Esto es con la finalidad de mostrar los datos de forma intuitiva y ordenada.

```
#REQUERIMIENTOS
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
dfc2['sum_numbers'] = dfc2['views'] + dfc2['likes'] + dfc2['comment_count']
plt.style.use('fivethirtyeight')
plt.figure(figsize=(20,15))
sns.barplot(x="category_name", y="sum_numbers", data=dfc2, estimator=np.mean, ci=None)
plt.title('Tendencia por categoría')
plt.xticks(rotation=60)
```

Se añade una columna llamada sum_numbers, que almacena la suma de número de vistas, número de likes y número de comentarios. Usamos la gráfica de barras, ya que una de las variables a incluir es categórica y queremos visualizar la tendencia de cada categoría. Además, especificamos que calcule la media para cada tipo de categoría y lo use como estimador en nuestro gráfico de barras.



B. ¿Qué categorías de video son las que más gustan? ¿Y las que menos gustan?

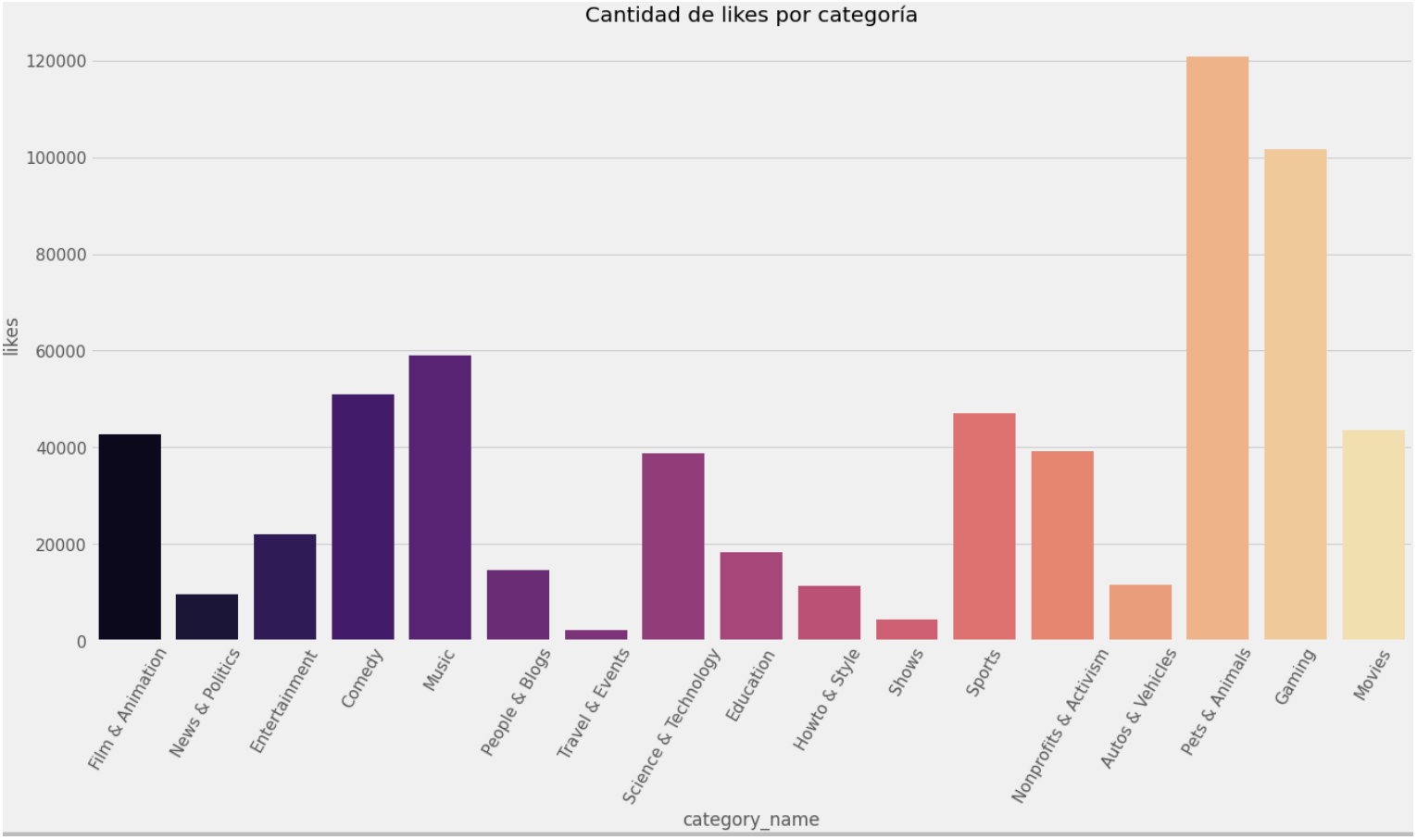
Para conocer qué categorías de video poseen más likes o dislikes, agruparemos la cantidad de likes y dislikes de cada video por categoría. Para esto, usamos la gráfica de barras.

En el caso de las categorías de vídeo que más gustan, empleamos lo siguiente:

```
plt.style.use('fivethirtyeight')
plt.figure(figsize=(20,15))
sns.barplot(x="category_name", y="likes", data=dfc2, estimator=np.mean, palette='magma', ci=None)
plt.title('Cantidad de likes por categoría')
plt.xticks(rotation=60)
```

Asignamos a la función np.mean como estimador para poder calcular la cantidad promedio de likes que hay por cada categoría en la gráfica.

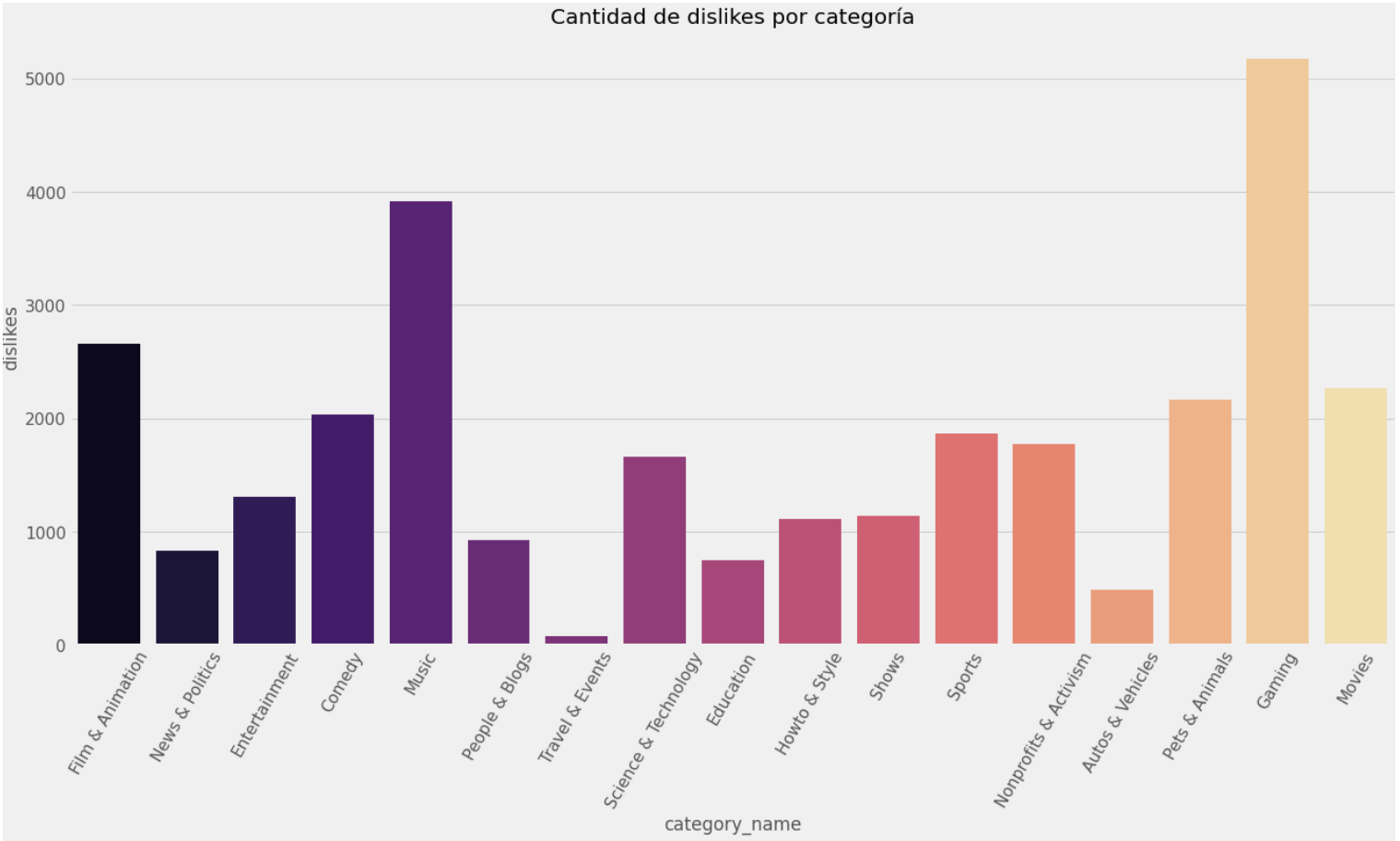
Como resultado, obtenemos



En el caso de las categorías de vídeo que menos gustan, empleamos lo siguiente:

```
plt.style.use('fivethirtyeight')
plt.figure(figsize=(20,15))
sns.barplot(x="category_name", y="dislikes", data=dfc2, estimator=np.mean, palette='magma', ci=None)
plt.title('Cantidad de dislikes por categoría')
plt.xticks(rotation=60)
```

Como resultado, obtenemos



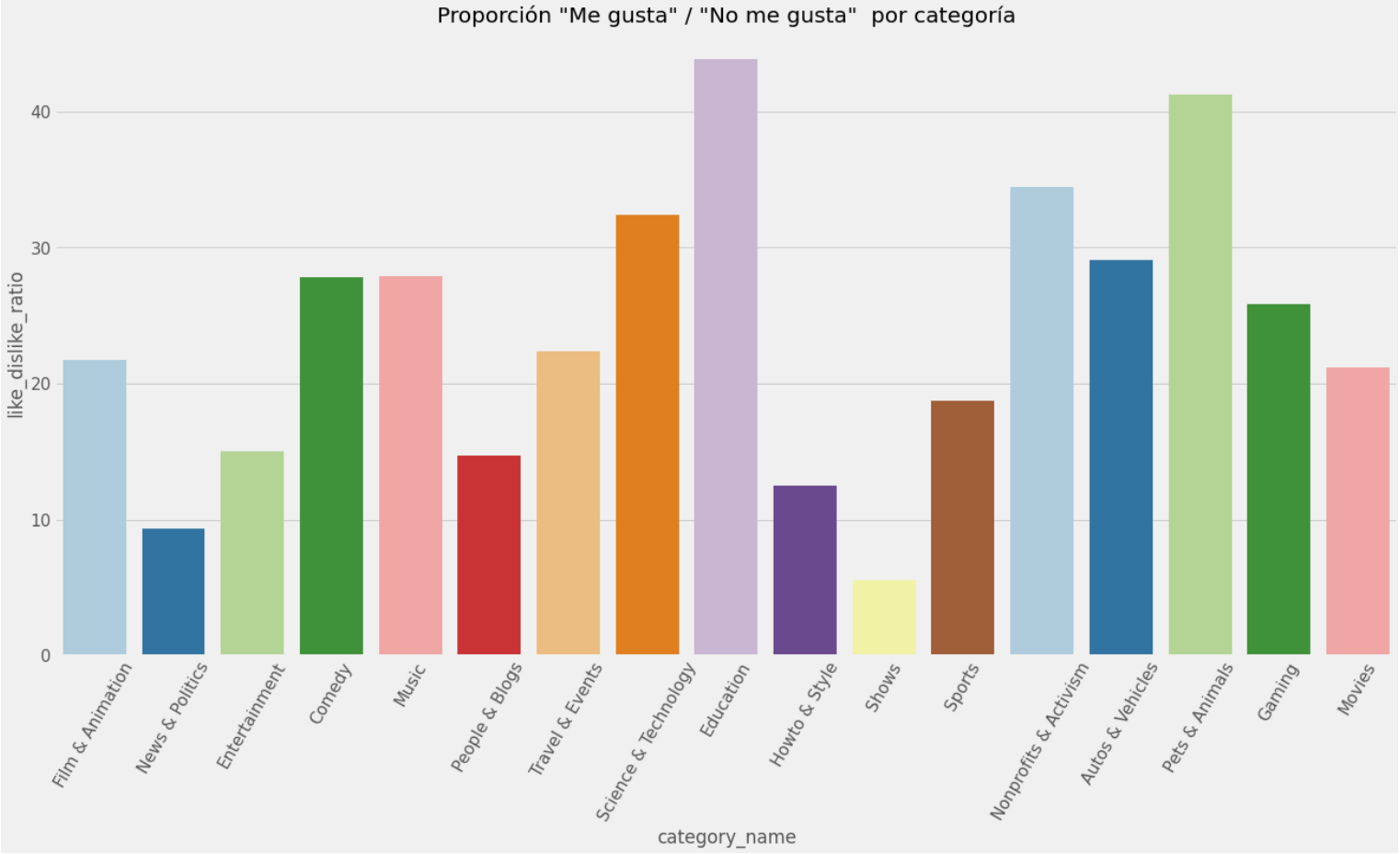
C. ¿Qué categorías de video tienen la mejor proporción (ratio) de "Me gusta" / "No me gusta"?

Para determinar la proporción "me gusta" / "no me gusta" de un video, es favorable crear una variable que guarde un valor que equivalga a la proporción pedida. En este caso, la variable like_dislike_ratio será la división de la cantidad de likes entre la cantidad de dislikes.

Usamos el siguiente código:

```
dfc2['like_dislike_ratio'] = dfc2['likes'] / dfc2['dislikes']
plt.style.use('fivethirtyeight')
plt.figure(figsize=(20,10))
g = sns.barplot(x="category_name", y="like_dislike_ratio", data=dfc2, estimator=np.mean, palette='Paired', ci=None)
plt.title('Proporción "Me gusta" / "No me gusta"  por categoría')
plt.xticks(rotation=60)
```

Utilizamos la gráfica de barras para ofrecer una mejor visualización de las proporciones para cada categoría y escogemos el método np.mean como estimador para la función barplot.



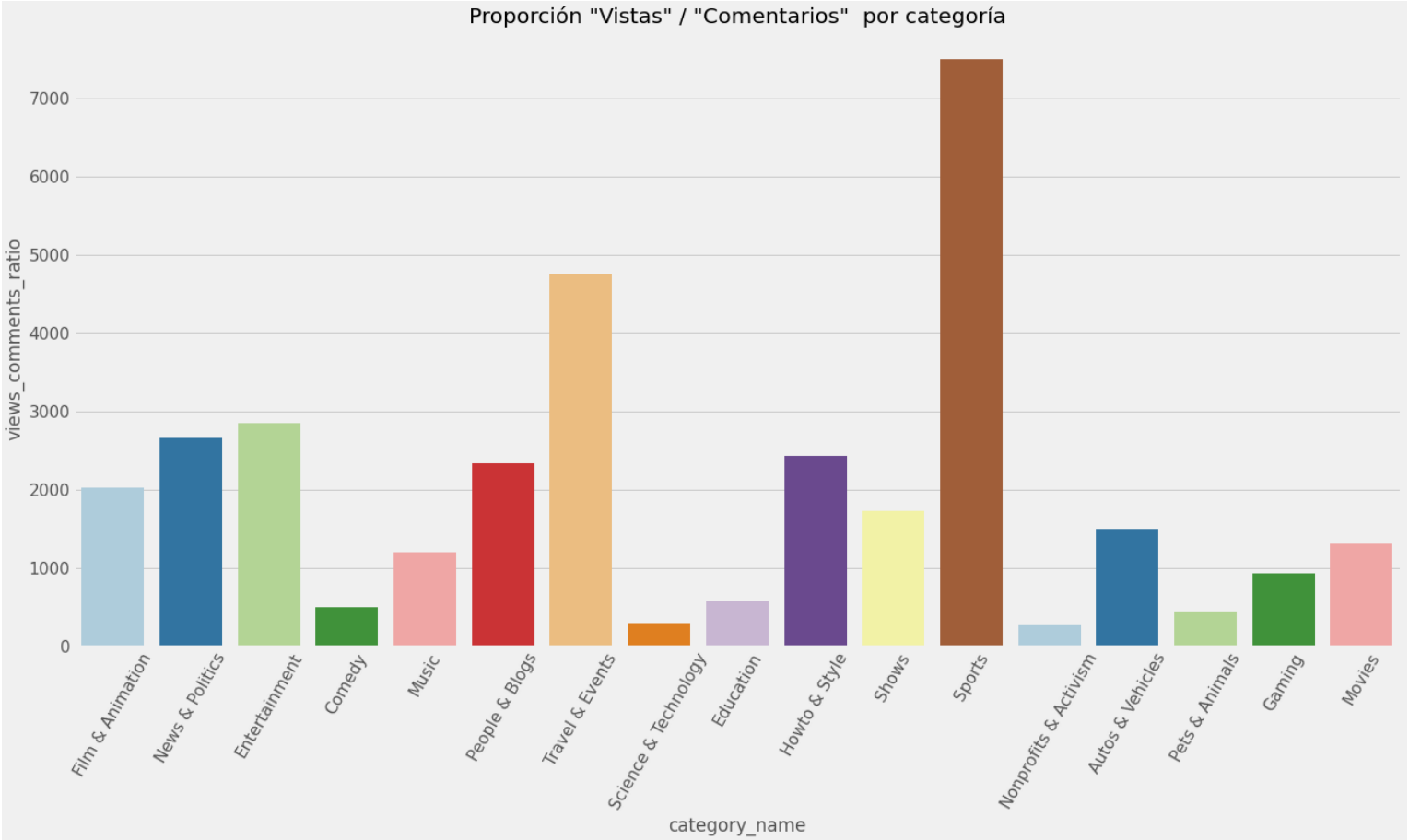
D. ¿Qué categorías de video tienen la mejor proporción (ratio) de “Vistas” / “Comentarios”?

Para calcular la proporción “Vistas” / “Comentarios” de un video, es necesario crear una variable que almacene un valor que equivalga a la proporción pedida. En este caso, la variable views_comments_ratio será la división de la cantidad de vistas entre la cantidad de comentarios por video.

Usamos el siguiente código:

```
dfc2['views_comments_ratio'] = dfc2['views'] / dfc2['comment_count']

plt.style.use('fivethirtyeight')
plt.figure(figsize=(20,10))
sns.barplot(x="category_name", y="views_comments_ratio", data=dfc2, estimator=np.mean, palette='Paired', ci=None)
plt.title('Proporción "Vistas" / "Comentarios"  por categoría')
plt.xticks(rotation=60)
```

El gráfico resultante es el siguiente:

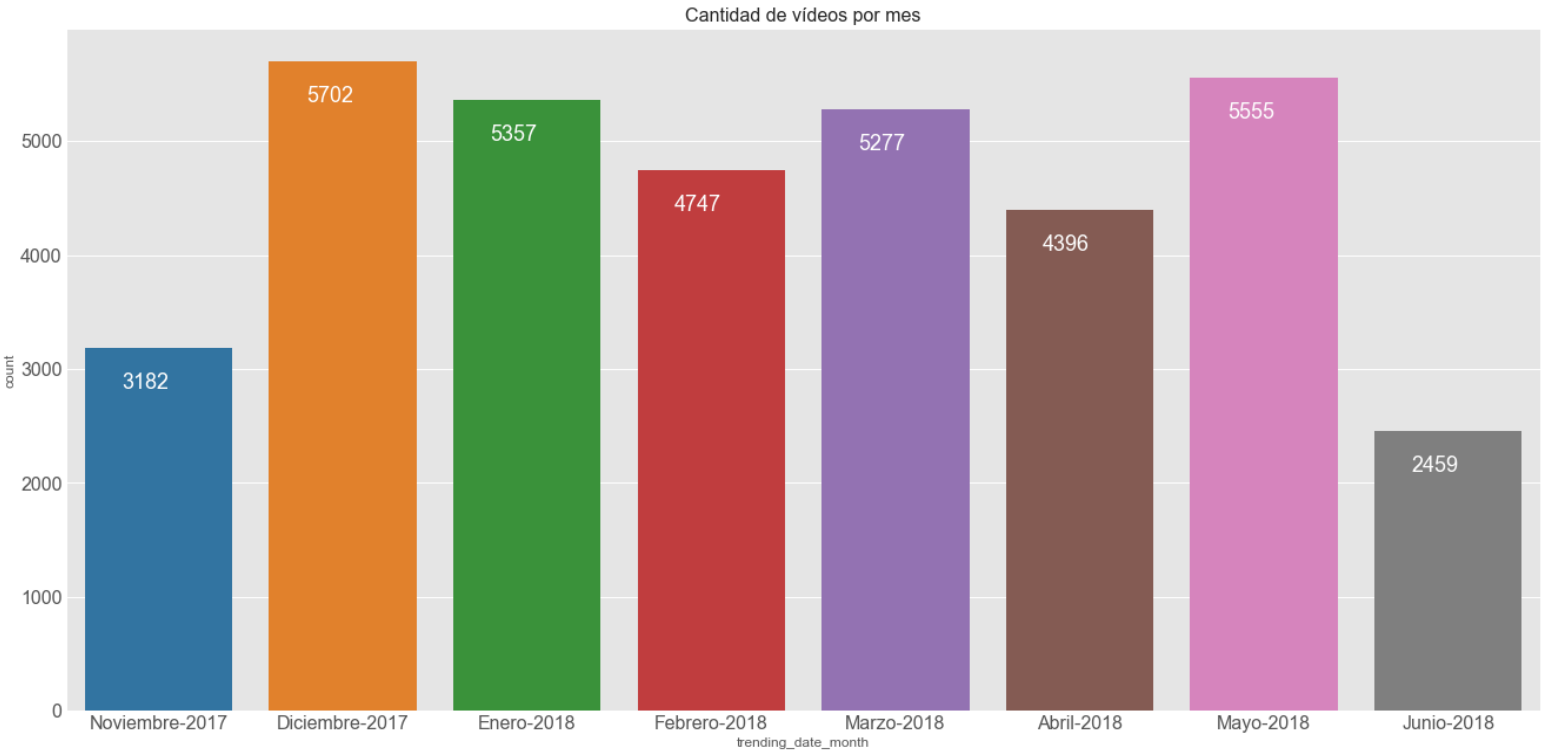


E. ¿Cómo ha cambiado el volumen de los videos en tendencia a lo largo del tiempo?

Ya que el dataset recolectado para el desarrollo de este trabajo contiene datos sobre los videos de Youtube en tendencia diariamente, vamos a juntar los videos por el mes en que llegaron a ser tendencia. Luego, contaremos el número de videos por mes y examinaremos de qué forma ha cambiado esa cantidad hallada.

```
plt.style.use('ggplot')
plt.figure(figsize=(20,10))
ax = sns.countplot(x="trending_date_month", data=dfc2, palette='tab10')
plt.title('Cantidad de vídeos por mes')
plt.tick_params(axis='both', labelsize=16)
for p in ax.patches:
    ax.annotate(f'\n{p.get_height()}', (p.get_x()+0.2, p.get_height()), ha='left', va='top', color='white', size=18)
```

Para poder visibilizar la cantidad de videos por mes, utilizamos el método countplot() proporcionada por la librería Seaborn. También agregamos el número exacto de video en tendencia por mes para lograr una mejor observación.

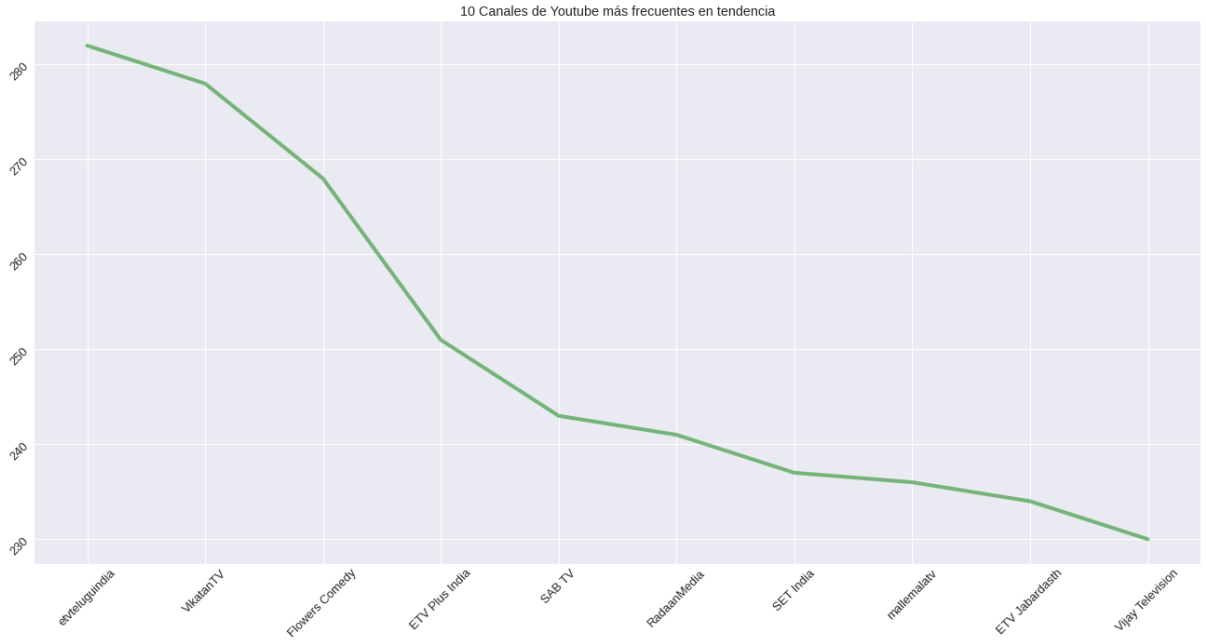


F. ¿Qué canales de Youtube son tendencias más frecuentemente? ¿Y cuáles con menos frecuencia?

Primero para responder esto hemos contado cuantas veces se repite cada canal dentro del data frame utilizando la columna channel title y el resultado de contar por canal lo hemos guardado en una variable llamada tab1.

```
tab1 = dfc2["channel_title"].value_counts()
fig, axes = plt.subplots(1,1, figsize=(20,10))
axes.plot(tab1[0:10], color='green', alpha=0.5)
axes.tick_params(axis='both', labelsize=13, labelrotation=45)
axes.title.set_text('10 Canales de Youtube más frecuentes en tendencia')
```

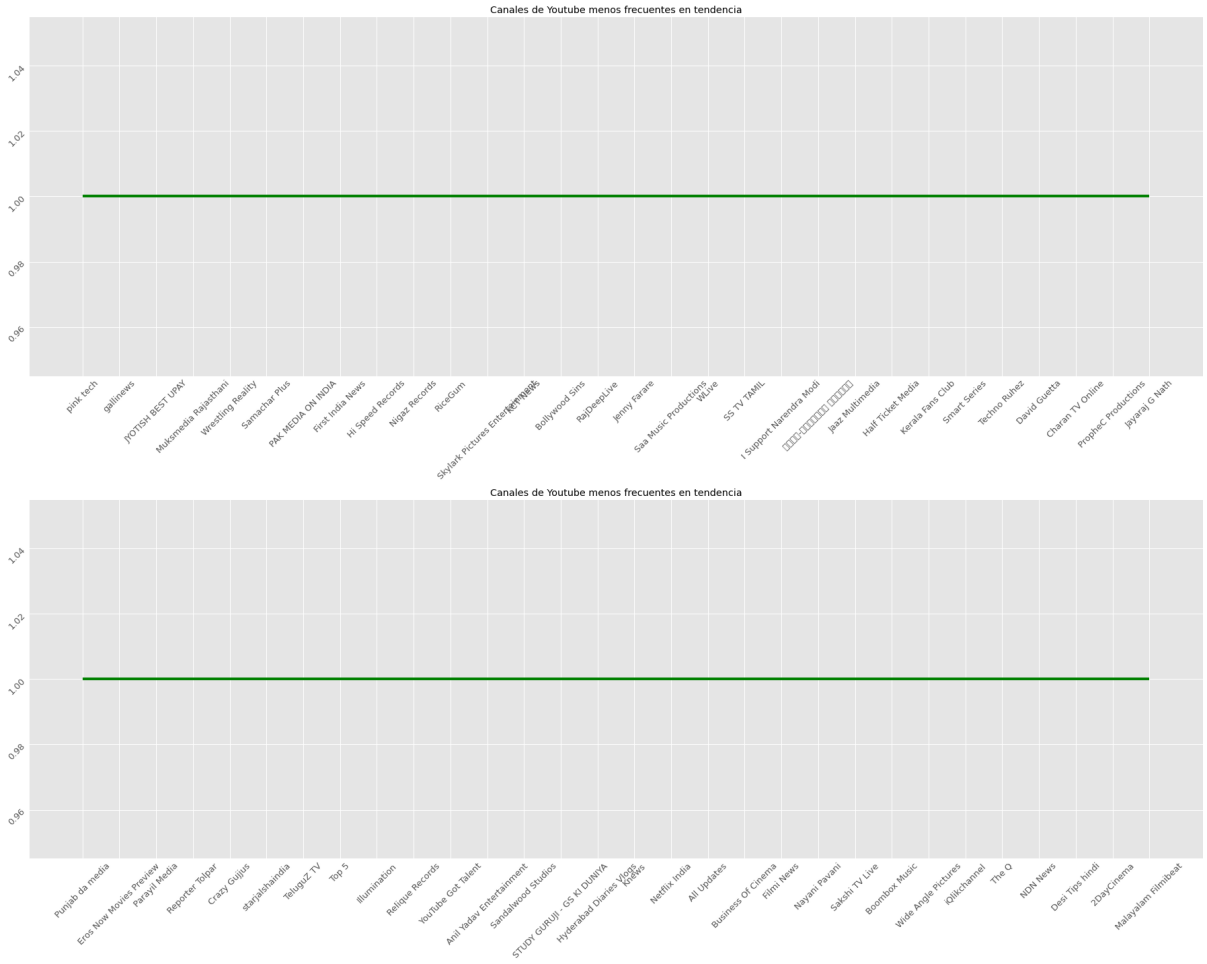
Para responder la primera utilizamos esta gráfica utilizamos los 10 primeros canales de la variable “tab1”:



A Continuación para la segunda pregunta usamos las mismas cosas excepto que acá tomamos desde el canal que se encuentra en la posición 1330 ya que a partir de este todos los canales dentro del data frame se encontraron en tendencia solo 1 vez.

```
fig, axes = plt.subplots(2,1, figsize=(25,20))
axes[0].plot(tab1[1330:1360], color='green')
axes[0].tick_params(axis='both', labelsize=13,labelrotation=45)
axes[0].title.set_text('Canales de Youtube menos frecuentes en tendencia')

axes[1].plot(tab1[1360:], color='green')
axes[1].tick_params(axis='both', labelsize=13,labelrotation=45)
axes[1].title.set_text('Canales de Youtube menos frecuentes en tendencia')
plt.tight_layout()
```

En estos 2 gráficos se pueden visualizar los canales menos frecuentes en tendencia:



G. ¿En qué estados se presenta el mayor número de “Vistas”, “Me gusta” y “No me gusta”?

Primero creamos un nuevo data frame llamado sub_df que guarda las columnas state, views, likes y dislikes, y luego en otro data frame llamado df_by_state va agrupar mediante la mediana y tomando como referencia la columna state.

```
sub_df = dfc2[['state', 'views', 'likes', 'dislikes']]
df_by_state = sub_df.groupby(['state']).mean()
df_by_state.reset_index(inplace=True)
```

Luego hemos usado otras librerías que nos permitirán visualizar de forma más detallada los gráficos de los mapas donde al mantener el mouse encima de un estado del país se podrá visualizar su promedio de views, likes y dislikes.

```
import chart_studio as py
import pandas as pd
import plotly.graph_objs as go
from plotly.offline import download_plotlyjs, init_notebook_mode, plot, iplot
init_notebook_mode(connected=True)
```

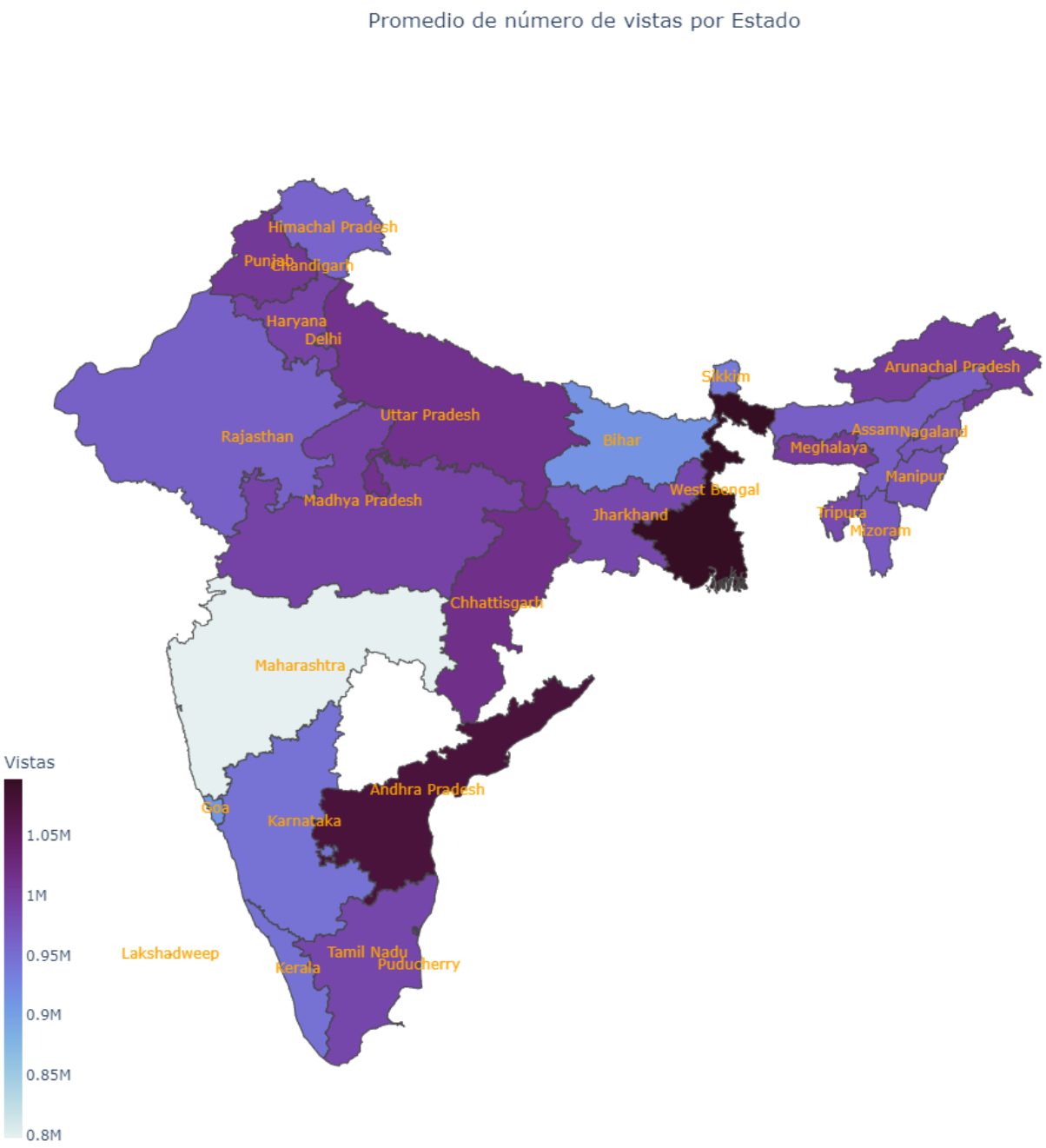
A continuación se visualizará el código necesario para visualizar los mapas de forma detallada y la imagen de dicho mapa:

- ☐ **VIEWS:**

  Código:

```
data = go.Choropleth(geojson="https://gist.githubusercontent.com/jbrobst/56c13bbbf9d97d187fea01ca62ea5112/raw/e388c4cae20aa53cb5090210a42ebb9b765c0a36/india_states.geojson",
                     featureidkey='properties.ST_NM',
                     locationmode='geojson-id',
                     locations=df_by_state['state'],
                     z=df_by_state['views'],
                     colorscale='dense',
                     colorbar=dict(title={'text': "Vistas"},
                                   thickness=15,
                                   len=0.35,
                                   xanchor='left',
                                   x=0.01,
                                   yanchor='bottom',
                                   y=0.05),
                    )
fig = go.Figure(data)
fig.update_geos(fitbounds = "locations",
                visible = False,
                lonaxis={'range': [68, 98]},
                lataxis={'range': [6, 38]})

fig.add_scattergeo(
  geojson="https://gist.githubusercontent.com/jbrobst/56c13bbbf9d97d187fea01ca62ea5112/raw/e388c4cae20aa53cb5090210a42ebb9b765c0a36/india_states.geojson",
  locations = df_by_state['state'],
  text = df_by_state['state'],
  featureidkey="properties.ST_NM",
  mode = 'text')


    fig.update_traces(textposition="middle center", textfont_color="orange", selector=dict(type='scattergeo'))

    fig.update_layout(
        title=dict(
            text="Promedio de número de vistas por Estado",
            xanchor='center',
            x=0.5,
            yref='paper',
            yanchor='bottom',
            y=1,
            pad={'b': 10}
        ),
        margin={'r': 0, 't': 30, 'l': 0, 'b': 0},
        height=1000,
        width=1000
    )
    fig.show()
```

Mapa:



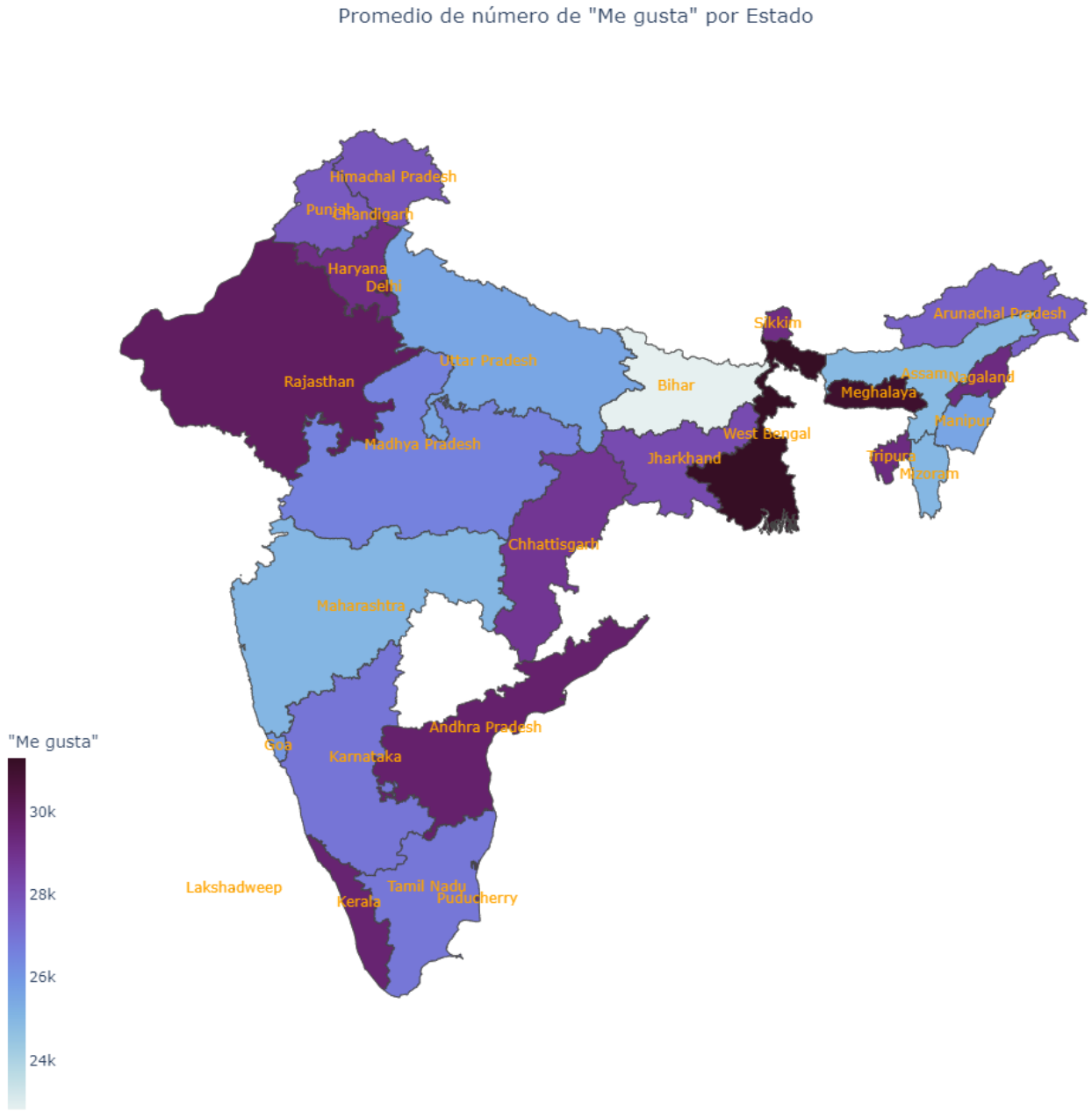
☐ **LIKES:**

Código:

```
data = go.Choropleth(geojson="https://gist.githubusercontent.com/jbrobst/56c13bbbf9d97d187fea01ca62ea5112/raw/e388c4cae20aa53cb5090210a42ebb9b765c0a36/india_states.geojson",
                     featureidkey='properties.ST_NM',
                     locationmode='geojson-id',
                     locations=df_by_state['state'],
                     z=df_by_state['likes'],
                     colorscale='dense',
                     colorbar=dict(title={'text': '"Me gusta"'},
                                   thickness=15,
                                   len=0.35,
                                   xanchor='left',
                                   x=0.01,
                                   yanchor='bottom',
                                   y=0.05),
                    )
fig = go.Figure(data)
fig.update_geos(fitbounds = "locations",
                visible = False,
                lonaxis={'range': [68, 98]},
                lataxis={'range': [6, 38]})

fig.add_scattergeo(
  geojson="https://gist.githubusercontent.com/jbrobst/56c13bbbf9d97d187fea01ca62ea5112/raw/e388c4cae20aa53cb5090210a42ebb9b765c0a36/india_states.geojson",
  locations = df_by_state['state'],
  text = df_by_state['state'],
  featureidkey="properties.ST_NM",
  mode = 'text')
```

```
fig.update_traces(textposition="middle center", textfont_color="orange", selector=dict(type='scattergeo'))

fig.update_layout(
    title=dict(
        text='Promedio de número de "Me gusta" por Estado',
        xanchor='center',
        x=0.5,
        yref='paper',
        yanchor='bottom',
        y=1,
        pad={'b': 10}
    ),
    margin={'r': 0, 't': 30, 'l': 0, 'b': 0},
    height=1000,
    width=1000
)
fig.show()
```

Mapa:



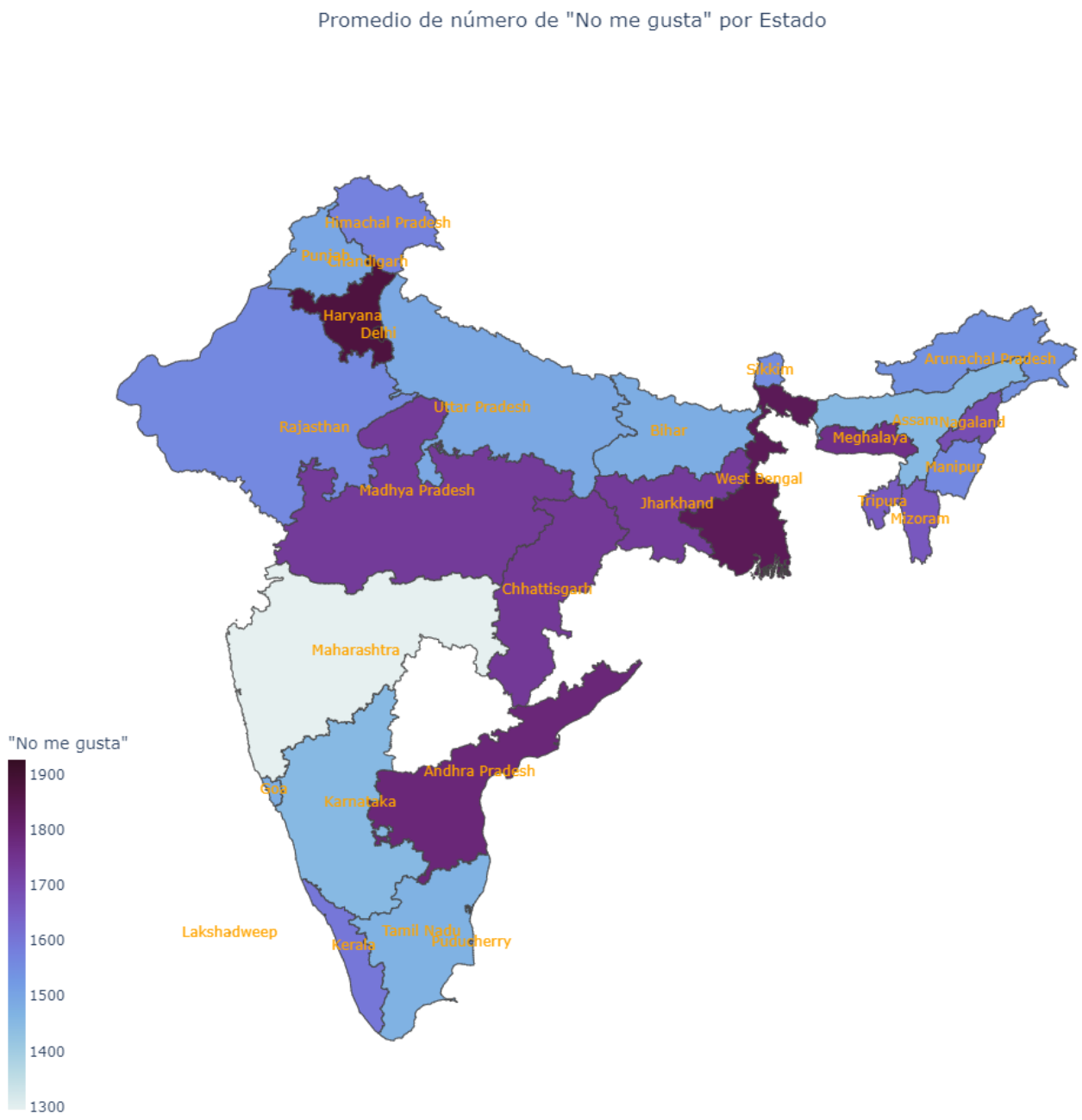
☐ **DISLIKES:**

Código:

```
data = go.Choropleth(geojson="https://gist.githubusercontent.com/jbrobst/56c13bbbf9d97d187fea01ca62ea5112/raw/e388c4cae20aa53cb5090210a42ebb9b765c0a36/india_states.geojson",
                     featureidkey='properties.ST_NM',
                     locationmode='geojson-id',
                     locations=df_by_state['state'],
                     z=df_by_state['dislikes'],
                     colorscale='dense',
                     colorbar=dict(title={'text': '"No me gusta"'},
                                   thickness=15,
                                   len=0.35,
                                   xanchor='left',
                                   x=0.01,
                                   yanchor='bottom',
                                   y=0.05),
                    )
fig = go.Figure(data)
fig.update_geos(fitbounds = "locations",
                visible = False,
                lonaxis={'range': [68, 98]},
                lataxis={'range': [6, 38]})

fig.add_scattergeo(
  geojson="https://gist.githubusercontent.com/jbrobst/56c13bbbf9d97d187fea01ca62ea5112/raw/e388c4cae20aa53cb5090210a42ebb9b765c0a36/india_states.geojson",
  locations = df_by_state['state'],
  text = df_by_state['state'],
  featureidkey="properties.ST_NM",
  mode = 'text')
```

```
fig.update_traces(textposition="middle center", textfont_color="orange", selector=dict(type='scattergeo'))

fig.update_layout(
    title=dict(
        text='Promedio de número de "No me gusta" por Estado',
        xanchor='center',
        x=0.5,
        yref='paper',
        yanchor='bottom',
        y=1,
        pad={'b': 10}
    ),
    margin={'r': 0, 't': 30, 'l': 0, 'b': 0},
    height=1000,
    width=1000
)
fig.show()
```

Mapa:



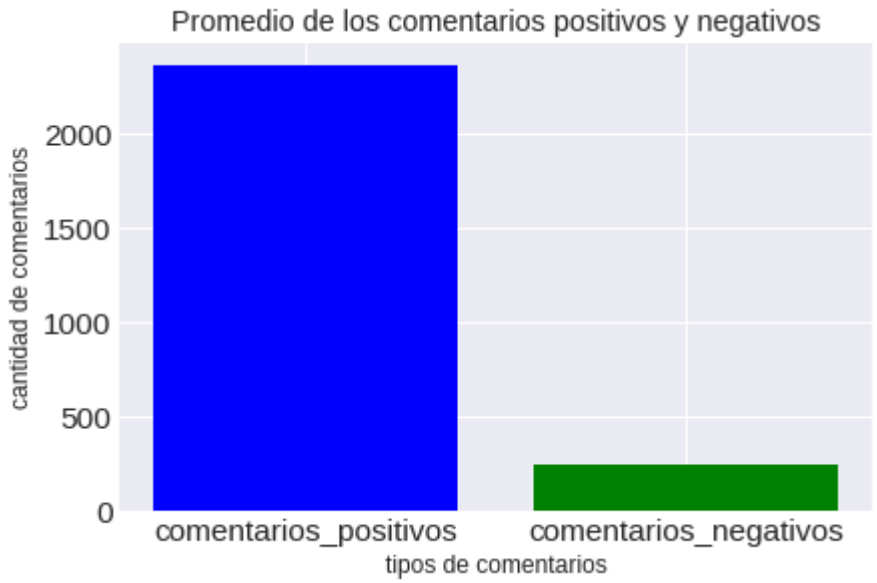
H. ¿Los videos en tendencia son los que mayor cantidad de comentarios positivos reciben?

Primero para resolver esto hemos relacionado likes, dislikes y la cantidad de comentarios para poder responder esta pregunta.

Mediante una fórmula hemos dividido la cantidad de comentarios en partes proporcionales y luego estos resultados los hemos guardado en columnas llamadas comentarios_positivos y comentarios_negativos.

```
dfc2_sddcvsd=dfc2.copy()

dfc2_sddcvsd['comentarios_positivos']=dfc2_sddcvsd['likes']/(dfc2_sddcvsd['likes']+dfc2_sddcvsd['dislikes'])*dfc2_sddcvsd['comment_count']
dfc2_sddcvsd['comentarios_negativos']=dfc2_sddcvsd['dislikes']/(dfc2_sddcvsd['likes']+dfc2_sddcvsd['dislikes'])*dfc2_sddcvsd['comment_count'

import numpy as np
import matplotlib.pyplot as plt
x1 = ['comentarios_positivos']
y1 = [int(dfc2_sddcvsd['comentarios_positivos'].mean())]
x2 = ['comentarios_negativos']
y2 = [int(dfc2_sddcvsd['comentarios_negativos'].mean())]

plt.bar(x1, y1, label="Barra azul", color='b')
plt.bar(x2, y2, label="Barra verde", color='g')

plt.plot()
plt.xlabel("tipos de comentarios")
plt.ylabel("cantidad de comentarios")
plt.title("Promedio de los comentarios positivos y negativos")
plt.show()
```

A Continuación se visualiza una gráfica del promedio total de los comentarios positivos y negativos :



5. Modelizar y evaluar los datos

El cliente, además de requerir una respuesta para cada una de las preguntas mencionadas previamente, quiere conocer si es factible predecir el número de “Vistas”, “Me gusta” y “No me gusta”. Para ello, es indispensable realizar una modelización y evaluación de los datos para descubrir un conocimiento.

En este sentido, tenemos que escoger los datos que pueden ser modelados. De acuerdo a la duda que el cliente quiere resolver, consideramos a las variables “likes”, “dislikes” y “views” como variables que son susceptibles a ser

moderadas. Asimismo, necesitamos descubrir si es posible predecir las cantidades de las variables “likes”, “dislikes” y “views”.

Para este caso, aplicaremos un modelo de regresión logística, ya que este analiza la relación entre una variable categórica dependiente y varias variables independientes calculando probabilidades con la ayuda de una función logística. Esto nos permitirá conocer el porcentaje de precisión que existe para predecir los valores de las variables deseadas.

Entonces, usaremos a las variables “category_id”, “comment_count”, “comments_disabled”, “ratings_disabled” y “state” como independientes, debido a que pueden influir significativamente en el comportamiento de las variables anteriormente nombradas. También, dependiendo de la variable que vayamos a modelar, incluiremos a “likes”, “dislikes” o “views” en el conjunto de variables de entrada para el modelo.

Ya que el modelo elegido no acepta variables categóricas en el conjunto de entradas, efectuaremos una conversión en los datos.

```
df = dfc2.copy()
for i in range(len(df)):
    if df['comments_disabled'][i] == 'VERDADERO':
        df['comments_disabled'][i] = 1
    else: df['comments_disabled'][i] = 0

    if df['ratings_disabled'][i] == 'VERDADERO':
        df['ratings_disabled'][i] = 1
    else: df['ratings_disabled'][i] = 0

df_only_state=df_by_state.copy()

ercd = []
for i in range(len(df_only_state['state'])):
    ercd.append(i)

df_only_state.assign(id = '')
df_only_state['id'] = ercd
df_only_state.drop(['views','likes','dislikes'],axis=1, inplace = True)

df = df.join(df_only_state.set_index('state'), on='state')
df.drop('state', axis=1, inplace=True)
df.rename(columns={'id':'state'}, inplace=True)
```

Creamos una copia del dataset dfc2 con el nombre df. Este nuevo conjunto de datos será utilizado para todo el proceso de modelación. Luego, ejecutamos la conversión de datos en las columnas “comments disabled” y “ratings_disabled”. Empleamos al dataframe df_by_state previamente creado para transformar los

valores de la columna “state” a número único para ser diferenciados. Por último, todas estas modificaciones son agregadas al dataset df.

Como el modelo de regresión logística solo puede predecir variables que son categóricas de 2 o más posibles clases, y no variables continuas, debemos clasificar los valores de las variables “likes”, “dislikes” o “views” usando rangos específicos para cada una.

```
#VISTAS
c = pd.cut(df[['views']].stack(), [0.0, 0.4*10**6, 1.0*10**7, 3.5*10**7], labels=['bajo', 'medio', 'alto'])
df_views = df.join(c.unstack().add_suffix('_cat'))
df_views = df_views[['category_id', 'views_cat', 'likes', 'dislikes', 'comment_count', 'comments_disabled', 'ratings_disabled', 'state']]
```

La variable “views” es catalogada en tres clases: bajo, medio y alto. Si los valores se encuentran en el rango 0 a 0.4E6, son clasificados como bajo. Si los valores se encuentran en el rango 0.4E6 a 1.0E7, son clasificados como medio. Si los valores se encuentran en el rango 1.0E7 a 3.5E7, son clasificados como alto. Estos valores transformados son almacenados en df_views.

```
#LIKES
c = pd.cut(df[['likes']].stack(), [0.0, 0.1*10**5, 0.1*10**6, 1.5*10**6], labels=['bajo', 'medio', 'alto'])
df_likes = df.join(c.unstack().add_suffix('_cat'))
df_likes = df_likes[['category_id', 'likes_cat', 'views', 'dislikes', 'comment_count', 'comments_disabled', 'ratings_disabled', 'state']]
```

La variable “likes” es catalogada en tres clases: bajo, medio y alto. Si los valores se encuentran en el rango 0 a 0.1E5, son clasificados como bajo. Si los valores se encuentran en el rango 0.1E5 a 0.1E6, son clasificados como medio. Si los valores se encuentran en el rango 0.1E6 a 1.5E6, son clasificados como alto. Estos valores transformados son almacenados en df_likes.

```
#DISLIKES
c = pd.cut(df[['dislikes']].stack(), [0.0, 0.8*10**3, 8*10**3, 200*10**3], labels=['bajo', 'medio', 'alto'])
df_dislikes = df.join(c.unstack().add_suffix('_cat'))
df_dislikes = df_dislikes[['category_id', 'dislikes_cat', 'views', 'likes', 'comment_count', 'comments_disabled', 'ratings_disabled', 'state']]
```

La variable “dislikes” es catalogada en tres clases: bajo, medio y alto. Si los valores se encuentran en el rango 0 a 0.8E3, son clasificados como bajo. Si los valores se encuentran en el rango 0.8E3 a 8.0E3, son clasificados como medio. Si los valores se encuentran en el rango 8.0E3 a 2.0E5, son clasificados como alto. Estos valores transformados son almacenados en df_dislikes.

Luego de haber preparado y ordenado los datos, procedemos a modelar los datos. Primero, inicializamos las librerías que se van a usar.

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report
```

Primero, modelamos la variable “views”.

```
X_train, X_test, y_train, y_test = train_test_split(df_views.drop('views_cat', axis=1), df_views['views_cat'], test_size=0.30, random_state=101)
logmodel = LogisticRegression(multi_class='multinomial', solver='lbfgs', max_iter=1000) #Parametros utilizados para problemas con multiclases
logmodel.fit(X_train,y_train)
predictions = logmodel.predict(X_test)
print(classification_report(y_test, predictions))
```

Se divide el conjunto de datos df_views en variables que servirán para el entrenamiento y posterior predicción del modelo. Iniciamos el modelo de regresión logística. Especificamos el atributo multi-class como “multinomial”, ya que la variable “views” presenta varias clases. También, escogemos el algoritmo “lbfgs”, pues es capaz de lidiar con la pérdida multinomial para variables de varias clases. Después, entrenamos el modelo con el método fit() y predecimos los valores con el método predict(). Por último, guardamos los resultados en la variable predictions e imprimimos un reporte con la función classfication_report() enviando los datos en y_test.

El reporte generado es el siguiente:

	precision	recall	f1-score	support
alto	0.43	0.31	0.36	144
bajo	0.74	0.93	0.83	6316
medio	0.82	0.54	0.65	4543
accuracy			0.76	11003
macro avg	0.66	0.59	0.61	11003
weighted avg	0.77	0.76	0.75	11003

A continuación, modelamos la variable “likes”.

```
X_train, X_test, y_train, y_test = train_test_split(df_likes.drop('likes_cat', axis=1), df_likes['likes_cat'], test_size=0.30, random_state=101)
logmodel = LogisticRegression(multi_class='multinomial', solver='lbfgs', max_iter=1000) #Parametros utilizados para problemas con multiclases
logmodel.fit(X_train,y_train)
predictions = logmodel.predict(X_test)
print(classification_report(y_test, predictions))
```

Se divide el conjunto de datos df_likes en variables que servirán para el entrenamiento y posterior predicción del modelo. Iniciamos el modelo de regresión logística. Especificamos el atributo multi-class como “multinomial”, ya que la variable “likes” presenta varias clases. También, escogemos el algoritmo “lbfgs”, pues es capaz de lidiar con la pérdida multinomial para variables de varias clases. Después, entrenamos el modelo con el método fit() y predecimos los valores con el método predict(). Por último, guardamos los resultados en la variable predictions e imprimimos un reporte con la función classfication_report() enviando los datos en y_test.

El reporte generado es el siguiente:

	precision	recall	f1-score	support
alto	0.74	0.61	0.67	858
bajo	0.85	0.95	0.90	7449
medio	0.67	0.50	0.57	2696
accuracy			0.81	11003
macro avg	0.75	0.69	0.71	11003
weighted avg	0.80	0.81	0.80	11003

Finalmente, modelamos la variable “dislikes”.

```
X_train, X_test, y_train, y_test = train_test_split(df_dislikes.drop('dislikes_cat', axis=1), df_dislikes['dislikes_cat'], test_size=0.30, random_state=101)
logmodel = LogisticRegression(multi_class='multinomial', solver='lbfgs', max_iter=1000) #Parametros utilizados para problemas con multiclases
logmodel.fit(X_train,y_train)
predictions = logmodel.predict(X_test)
print(classification_report(y_test, predictions))
```

Se divide el conjunto de datos df_dislikes en variables que servirán para el entrenamiento y posterior predicción del modelo. Iniciamos el modelo de regresión logística. Especificamos el atributo multi-class como “multinomial”, ya que la variable “dislikes” presenta varias clases. También, escogemos el algoritmo “lbfgs”, pues es capaz de lidiar con la pérdida multinomial para variables de varias clases. Después, entrenamos el modelo con el método fit() y predecimos los valores con el método predict(). Por último, guardamos los resultados en la variable predictions e imprimimos un reporte con la función classfication_report() enviando los datos en y_test.

El reporte generado es el siguiente:

	precision	recall	f1-score	support
alto	0.05	0.16	0.08	384
bajo	0.82	0.45	0.58	7389
medio	0.37	0.66	0.47	3230
accuracy			0.50	11003
macro avg	0.41	0.42	0.38	11003
weighted avg	0.66	0.50	0.53	11003

6. Conclusiones

- **Por Categoría de Videos**

1. ¿Qué categorías de videos son las de mayor tendencia?
Las categorías de videos con mayor tendencia son Gaming, Movies y Music.

2. ¿Qué categorías de videos son los que más gustan? ¿Y las que menos gustan?
Las categorías de videos que más gustan con respecto a los likes son Pets & Animals, Gaming y Music; y los que menos gustan son Travel & Events, Shows y News & Politics.
Y con respecto a los dislikes los videos que más gustan son Travel & Events, Autos & Vehicles y Shows y News & Politics; y los que menos gustan son Gaming, Music y Film & Animation.

3. ¿Qué categorías de videos tienen la mejor proporción (ratio) de "Me gusta" / "No me gusta"?
Las categorías de videos que tienen la mejor proporción de "Me gusta" / "No me gusta" son Education, Pets & Animals y Nonprofits & Activism.

4. ¿Qué categorías de videos tienen la mejor proporción (ratio) de "Vistas" / "Comentarios"?
Las categorías de videos que tienen la mejor proporción de "Vistas" / "Comentarios" son Sports, Travel & Events y Entertainment.

▪ Por el tiempo transcurrido

5. ¿Cómo ha cambiado el volumen de los videos en tendencia a lo largo del tiempo?
Con respecto a la cantidad de videos en tendencia, tuvo un gran aumento en el mes de Diciembre de 2017 luego se mantuvo con aproximadamente la misma cantidad hasta el mes de Mayo de 2018, y por último al siguiente mes disminuyó bastante la cantidad de videos en tendencia.

▪ Por Canales de YouTube

6. ¿Qué canales de YouTube son tendencia más frecuentemente? ¿Y cuáles con menos frecuencia?
Los canales de YouTube que son tendencia más frecuentemente son etvteluguindia, VikatanTV y Flowers Comedy, y con respecto a los que son tendencia menos frecuentes se encuentran bastantes canales, ya que estos tan solo cuentan con tan solo una aparición en tendencia y los más posible que estos nunca vuelvan a estar

en tendencia de nuevo, algunos de estos canales son Trending Today , Business Of Cinema, Bollywood Sins y Ilumination.

▪ Por la geografía del país

7. ¿En qué Estados se presenta el mayor número de "Vistas", "Me gusta" y "No me gusta"?

Los Estados que presentan el mayor número de "Vistas" son West Bengal, Andhra Pradesh y Chandigarh.
Los Estados que presentan el mayor número de "Me gusta" son West Bengal, Nagaland y Haryana.
Los Estados que presentan el mayor número de "No me gusta" son Chandigarh, Andhra Pradesh y West Bengal.

▪ Adicionales

8.¿Es factible predecir el número de "Vistas" o "Me gusta" o "No me gusta"?

Luego de haber efectuado un modelo de regresión logística para medir la factibilidad de predicción del número de "Vistas" o "Me gusta" o "No me gusta", se descubrió lo siguiente:
En el caso de "Vistas", hubo una precisión del 43% para identificar correctamente los valores de la clase "alto", una precisión del 82% para identificar correctamente los valores de la clase "medio" y una precisión del 74% para identificar correctamente los valores de la clase "bajo" de la variable "views". Adicionalmente, se detectó una precisión ponderada total del 77%. Por lo tanto, podemos afirmar que es posible predecir a qué clase puede pertenecer la cantidad de "Vistas" de un video en tendencias.

En el caso de "Me gusta", hubo una precisión del 74% para identificar correctamente los valores de la clase "alto", una precisión del 67% para identificar correctamente los valores de la clase "medio" y una precisión del 85% para identificar correctamente los valores de la clase "bajo" de la variable "views". Adicionalmente, se detectó una precisión ponderada total del 80% . Por tal razón, podemos afirmar que es posible predecir a qué clase puede pertenecer la cantidad de "Vistas" de un video en tendencias.

En el caso de "No me gusta", hubo una precisión del 5% para identificar correctamente los valores de la clase "alto", una precisión del 37% para identificar correctamente los valores de la clase "medio" y una precisión del 82% para

identificar correctamente los valores de la clase “bajo” de la variable “views”. Adicionalmente, se detectó una precisión ponderada total del 66%, lo cual no resulta ser un porcentaje alto o aceptable de aciertos que produjo nuestro modelo. Por ese motivo, no es muy adecuado predecir a qué clase puede pertenecer la cantidad de “Vistas” de un video en tendencias.

9.¿Los videos en tendencia son los que mayor cantidad de comentarios positivos reciben?

Si, los videos en tendencia son los que mayor cantidad de comentarios positivos reciben.

7. Link del Github:

https://github.com/sergio185678/EB-2021-1-CC51