TRABAJO ACADÉMICO GRUPAL - TF

ADMINISTRACIÓN DE LA INFORMACIÓN

Docente: Reyes Silva, Patricia Daniela

Sección: CC51

Grupo: 2

Integrantes:

Galindo Alvarez, Franco - U202010807

Goyas Ayllon, Leonardo Andre - U202010206

Villafuerte Ramirez, Diego Tomas - U202010546

Índice

1. Objetivos del Proyecto

Objetivo General

- Desarrollar un Análisis Exploratorio de Datos (EDA) donde se creará conocimientos a partir del uso de datos.

Objetivos específicos

- Resolver un problema básico de Modelación de Datos
- Conocer las tendencias de los videos de Youtube en Estados Unidos y sus requerimientos
- Responder a cada unos de los requerimientos de nuestro país asignado
- Hacer uso de las herramientas aprendidas durante el curso de Administración de la Información
- Respetar la estructura del entregable, así como su entrega y publicación en la fecha establecida

2. Caso de Análisis

Los datos que se utilizaran para el trabajo provienen de la página Kaggle, se pueden encontrar como “Trending YouTube Video Statistics”. Estos fueron publicados en 2018 por el usuario Mitchell J, proveniente del Reino Unido.

Este conjunto de datos se pueden emplear para técnicas como *Sentiment Analysis,* entrenar algoritmos de *Machine Learning*, realizar un Análisis Exploratorio de Datos (EDA), entre otras aplicaciones. De esta forma se puede obtener información sobre los factores que afectan a la viralidad de los videos en Youtube. Esta información, es potencialmente útil para empresas que busquen beneficiarse de grandes visitas o impacto en la plataforma, tales como: Empresas de Marketing digital, Agencias de influencers (*“youtubers”*) e influencers independientes.

3. Conjunto de Datos (Data Set)

La base de datos “raw” tiene las siguientes variables:

Nombre de la variable	**Tipo de dato**	**Descripción**
video_id	object	Identificador único del video.
trending_date	object	Fecha y hora en la que el video fue *“trending”*.
title	object	Título que aparece del video en YouTube.
channel_title	object	Título del canal que subió el video a YouTube.
category_id	int64	Identificador de la categoría, el cual se relaciona con el *“US_category_id.json”*.
publish_time	object	Fecha en que se publicó el video.
tags	object	Serie de cadenas con las que se etiquetó el video.
views	int64	Cantidad de visitas que tuvo el video al momento de su *“trend”*.
likes	int64	Cantidad de likes que tuvo el video al momento de su *“trend”*.
dislikes	int64	Cantidad de dislikes que tuvo el video al momento de su *“trend”*.
comment_count	int64	Cantidad de comentarios que tuvo el video al momento de su *“trend”*.
thumbnail_link	object	Link de la imagen que se usó como *“thumbnail”*.
comments_disabled	bool	Booleano para determinar si los comentarios están deshabilitados.

ratings_disabled	bool	Booleano para determinar si los likes y dislikes están deshabilitados.
video_error_or_removed	bool	Booleano para determinar si el video está caído o presenta errores actualmente.
description	object	Descripción que el canal proporcionó sobre el video.
state	object	Estado de EEUU en que fue viral el video.
lat	float64	Coordenadas de la localización donde el video fue viral.
lon	float64	
geometry	object	Lista que contiene la longitud y latitud de las coordenadas.

Cabe destacar que las variables de tipo object funcionan como strings (cadenas de texto). Además consideramos que todavía hay algunas variables que podemos cambiar para mejorar el proceso de análisis, el cual se identificarán y cambiarán en el siguiente punto del proyecto.

4. Análisis Exploratorio de los Datos

Cargar Datos

Para cargar el dataset, empleamos la función read_csv(), de la librería pandas.

```
yt_data = pd.read_csv("../data/USvideos_cc50.csv")
```

Después de ello, necesitamos agregar un dataset donde podamos leer las categorías, para más adelante agregar el título como una columna en el data frame inicial.

```
#Creating a dataset to read the category types
categories = pd.read_json("../data/US_category_id.json")
categories["items"].to_json("../data/categories.json")
items = pd.read_json("../data/categories.json",orient="index")
items["snippet"].to_json("../data/items.json")
snippet = pd.read_json("../data/items.json",orient='index')
df2 = pd.DataFrame()
df2["id"] = items["id"]
df2["title"] = snippet["title"]
df2
```

Inspeccionar Datos

Una vez cargada la data, inspeccionamos los tipos de datos de cada columna.

```
yt_data.dtypes
```

```
video_id                  object
trending_date             object
title                     object
channel_title             object
category_id                int64
publish_time              object
tags                      object
views                      int64
likes                      int64
dislikes                   int64
comment_count              int64
thumbnail_link            object
comments_disabled           bool
ratings_disabled            bool
video_error_or_removed      bool
description               object
state                     object
lat                      float64
lon                      float64
geometry                  object
dtype: object
```

También nos interesa saber la cantidad de filas, esto lo obtenemos gracias a la función *"len"*.

```
print(len(yt_data))
```

```
40949
```

Después queremos obtener la cantidad de valores que sean *"NULL"* o esten vacios

```
yt_data_u[yt_data_u == "NULL"].sum(skipna=True)
yt_data_u[yt_data_u == ""].sum(skipna=True)
```

```
video_id                    0
trending_date               0
title                       0
channel_title               0
category_id               0.0
publish_time                0
tags                        0
views                     0.0
likes                     0.0
dislikes                  0.0
comment_count             0.0
thumbnail_link              0
comments_disabled           0
ratings_disabled            0
video_error_or_removed      0
description                 0
state                       0
lat                       0.0
lon                       0.0
geometry                    0
dtype: object
```

En adición, buscamos los datos con valor *"NaN"*

```
yt_data_u.isna().sum()
```

```
video_id                    0
trending_date               0
title                       0
channel_title               0
category_id                 0
publish_time                0
tags                        0
views                       0
likes                       0
dislikes                    0
comment_count               0
thumbnail_link              0
comments_disabled           0
ratings_disabled            0
video_error_or_removed      0
description               570
state                       0
lat                         0
lon                         0
geometry                    0
dtype: int64
```

Asimismo, verificamos que todos los *"video_id"* sean de 11 caracteres.

```
for i in yt_data_u["video_id"]:
    if(len(i)!=11):
        print("Error")
```

```
#Verificar que todos los id sean de 11 caracteres
for i in yt_data_u["video_id"]:
    if(len(i)!=11):
        print("Error")
#No hay ningun error en video_id
```

[10] ✓ 0.2s Python

Pre-Procesar los Datos

Aplicando las técnicas de Limpieza de Datos, eliminamos los registros duplicados. Borrando en total solo 1 fila.

```
yt_data_u = yt_data.drop_duplicates()

print(len(yt_data_u))
```

```
40948
```

Luego, aplicamos una técnica de Reducción de Datos, borrando la columna *"description"*, ya que es la única que posee valores de tipo "*NaN*" y no es necesario para el análisis de este proyecto.

```
yt_data_u.drop(columns=["description"], inplace=True, axis = 1)
```

Después, aplicamos una técnica de Transformación de Datos, para cambiar los datos de las columnas "*trending_date*" y "*publish_time*" de tipo object a fechas. Este proceso se tendrá que repetir siempre que se cargue la nueva base de datos.

```
yt_data_u["trending_date"] = pd.to_datetime(yt_data_u["trending_date"],
format="%y.%d.%m")

yt_data_u["publish_time"] = pd.to_datetime(yt_data_u["publish_time"],
format="%Y-%m-%dT%H:%M:%S.%fZ")
```

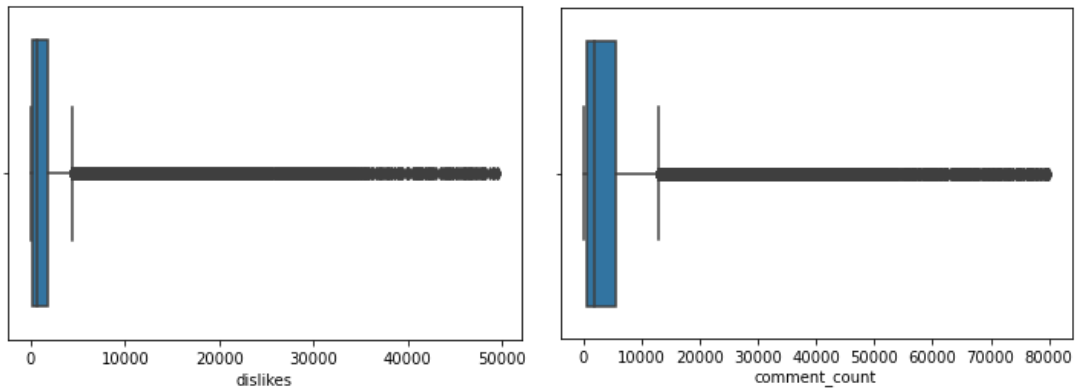
Asimismo, creamos una nueva columna para guardar los nombres de las categorías, usando el id que tiene el dataset de YouTube como indicador para seleccionar del dataset de categorías.

```
yt_data_u["category_name"] =
yt_data_u["category_id"].map(df2.set_index("id")["title"])

yt_data_u["category_name"]
```

```
0          People & Blogs
1           Entertainment
2                  Comedy
3           Entertainment
4           Entertainment
                  ...
40944       Pets & Animals
40945       People & Blogs
40946        Entertainment
40947    Film & Animation
40948              Gaming
```

Antes de terminar, eliminamos los valores atípicos de nuestra base de datos, para obtener resultados más generales y nos permite hacer mejores análisis.

```
yt_data_u = yt_data_u[yt_data_u["dislikes"] <= 50000]

sns.boxplot(yt_data_u["dislikes"])

plt.show()

yt_data_u = yt_data_u[yt_data_u["comment_count"] <= 80000]

sns.boxplot(yt_data_u["comment_count"])
```



Finalmente, revisamos que el dataset se encuentre en buen estado y lo guardamos en un csv para realizar el análisis y los modelos.

```
print(len(yt_data_u))

print((len(yt_data)-len(yt_data_u))/len(yt_data)*100,'% borrados')

yt_data_u.head()
```

```
40305
1.5726879777283937 % borrados
```

```
yt_data_u.to_csv("../data/yt_data.csv", index=False)
```

Visualizar Datos

▪ Por Categoría de Videos

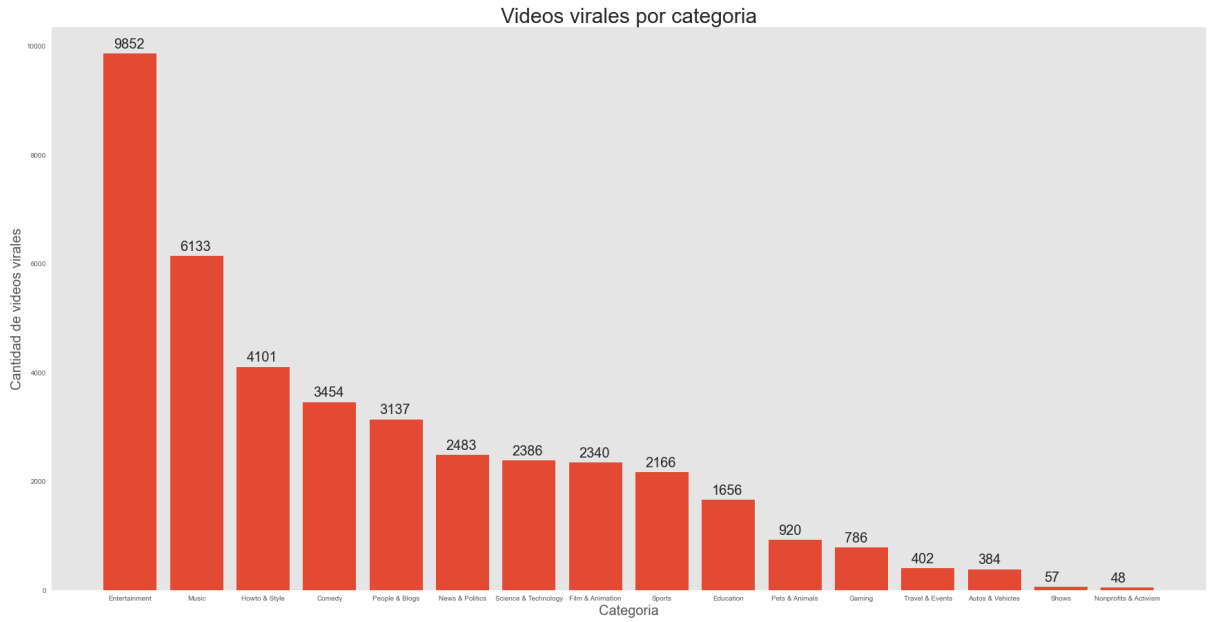
1. ¿Qué categorías de videos son las de mayor tendencia?

Creamos un dataset con los datos del nuevo csv y creamos un nuevo dataset con la frecuencia de esto.

```
data["category_name"].value_counts()

print(data["category_name"].value_counts().head())
```

Luego usamos matplotlib para graficar estos valores en un modelo estadístico. De la siguiente gráfica se puede reconocer que las categorías con mayor tendencia son de:

1. *"Entertainment"*
2. *"Music"*
3. *"Howto & Style"*
4. *"Comedy"*
5. *"People & Blogs"*



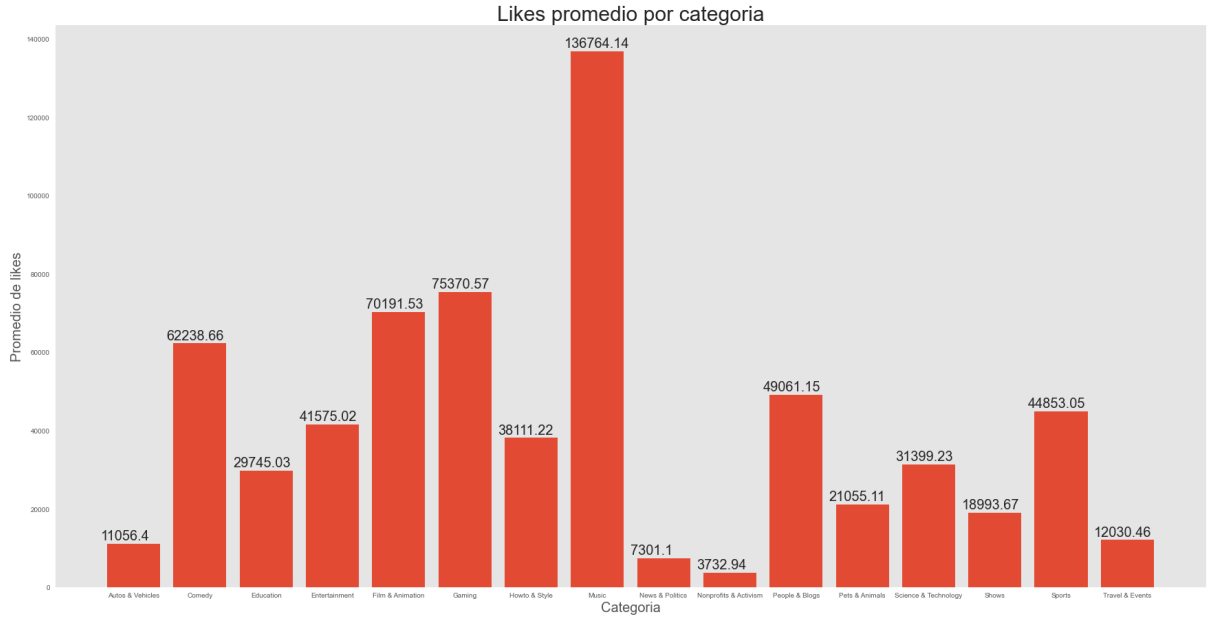
Categoría	**Cantidad de videos virales**
Entertainment	9852
Music	6133
Howto & Style	4101
Comedy	3454
People & Blogs	3137
News & Politics	2483
Science & Technology	2386
Film & Animation	2340
Sports	2166
Education	1656
Pets & Animals	920
Gaming	786
Travel & Events	402
Autos & Vehicles	384
Shows	57
Nonprofits & Activism	48

2. ¿Qué categorías de videos son los que más gustan? ¿Y las que menos gustan?

Utilizamos el dataset y lo agrupamos por categoría. Usando esta agrupación, sacamos el promedio de likes por categoría.

```
print(data.groupby("category_name")["likes"].mean())

print("Mas likes:",data.groupby("category_name")["likes"].mean().max())
```

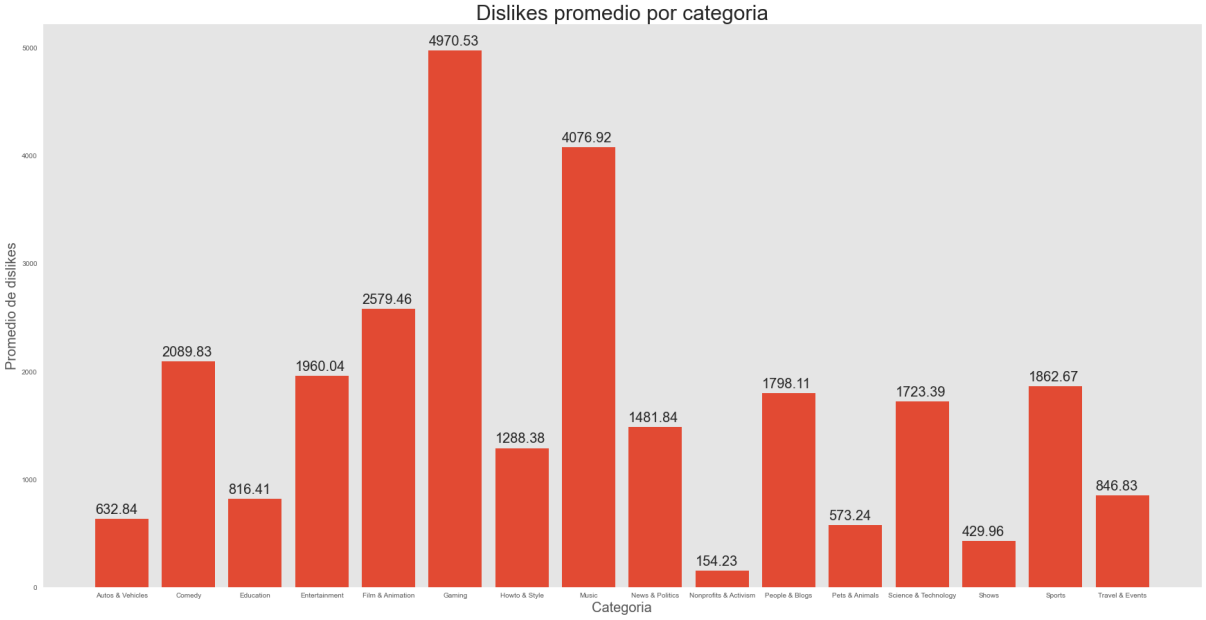
Usando matplotlib graficamos los resultados, y de este podemos determinar que la categoría con más frecuencia de tener likes es la de *"Music"*.



Categoría	**Promedio de likes**
Music	136764.14
Gaming	75370.57
Film & Animation	70191.53
Comedy	62238.66
People & Blogs	49061.15
Sports	44853.05
Entertainment	41575.02
Howto & Style	38111.22
Science & Technology	31399.23
Education	29745.03
Pets & Animals	21055.11
Shows	18993.67

Travel & Events	12030.46
Autos & Vehicles	11056.40
News & Politics	7301.10
Nonprofits & Activism	3732.94

Y luego, usando el mismo método, sacamos los datos de los dislikes y los graficamos con matplotlib. De estos datos podemos concluir que la categoría *"Gaming"* recibe más dislikes con más frecuencia.



Categoría	**Promedio de dislikes**
Gaming	4970.53
Music	4076.92
Film & Animation	2579.46
Comedy	2089.83
Entertainment	1960.04
Sports	1862.67
People & Blogs	1798.11

Science & Technology	1723.39
News & Politics	1481.84
Howto & Style	1288.38
Travel & Events	846.83
Education	816.41
Autos & Vehicles	632.84
Pets & Animals	573.24
Shows	429.96
Nonprofits & Activism	154.23

3. ¿Qué categorías de videos tienen la mejor proporción (ratio) de “Me gusta” / “No me gusta”?

Primero necesitamos crear un arreglo que nos muestre la suma de likes y otro con la de dislikes, por categoría. Con esto, creamos un nuevo arreglo el cual tenga la proporción de los Likes/Dislikes.

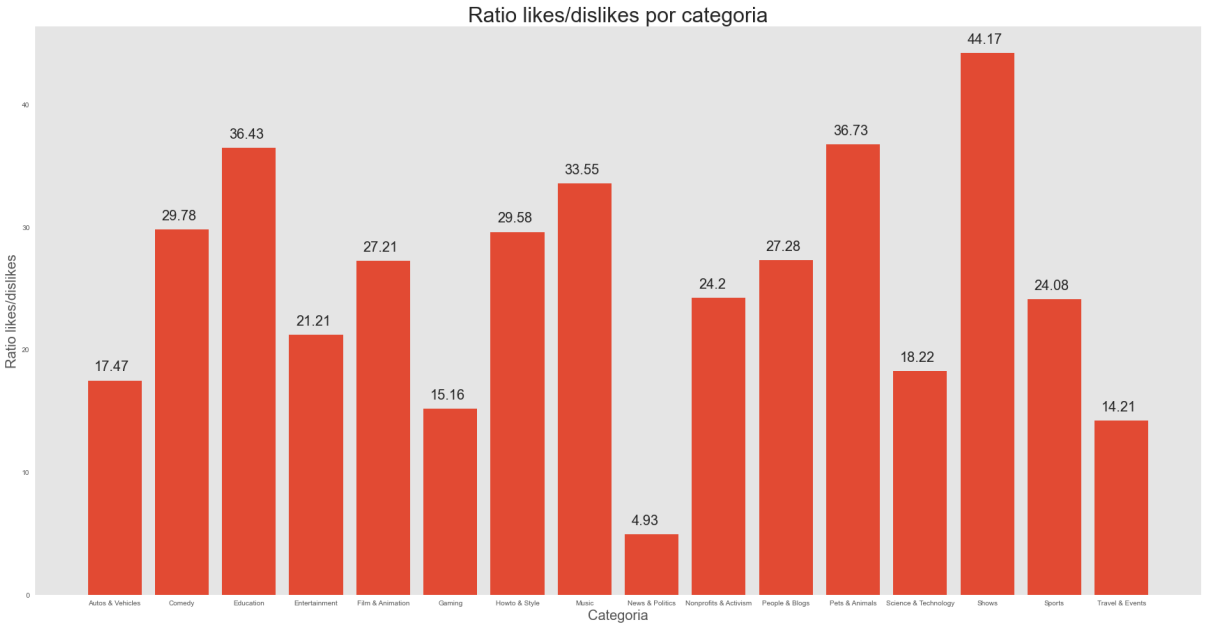
```
likesByCategory = data.groupby("category_name")["likes"].sum()

dislikesByCategory = data.groupby("category_name")["dislikes"].sum()

likeDisRatio = likesByCategory/dislikesByCategory
```

Luego graficamos los datos de nuestro nuevo arreglo likeDisRatio y, considerando que el mejor ratio son mayores, ya que significa que contienen más Likes que Dislikes, las mejores categorías son las siguientes:

1. *“Shows”*
2. *“Pets & Animals”*
3. *“Education”*

4. *“Music”*

5. *“Comedy”*



Categoría	**Ratio likes/dislikes**
Shows	44.17
Pets & Animals	36.73
Education	36.43
Music	33.55
Comedy	29.78
Howto & Style	29.58
People & Blogs	27.28
Film & Animation	27.21
Nonprofits & Activism	24.20
Sports	24.08
Entertainment	21.21
Science & Technology	18.22
Autos & Vehicles	17.47

Gaming	15.16
Travel & Events	14.21
News & Politics	4.93

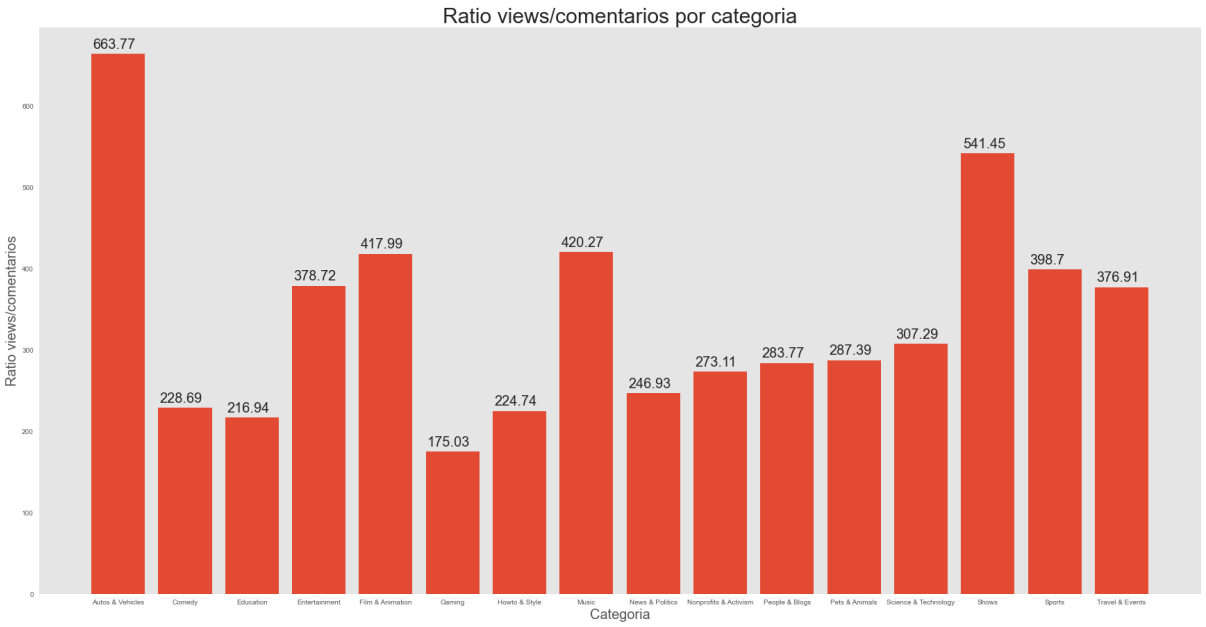
4. ¿Qué categorías de videos tienen la mejor proporción (ratio) de “Vistas” / “Comentarios”?

Utilizando la misma lógica de la pregunta anterior, conseguimos los datos de las vistas y número de comentarios.

```
viewsByCategory = data.groupby("category_name")["views"].sum()

commentsByCategory                                                =
data.groupby("category_name")["comment_count"].sum()

viewComRatio = viewsByCategory/commentsByCategory
```

Usando la gráfica podemos determinar que, considerando que los mejores ratios son menores, ya que significan más comentarios en razón de sus vistas, las mejores categorías son las siguientes:

1. *“Gaming”*
2. *“Education”*
3. *“Howto & Style”*
4. *“Comedy”*
5. *“News & Politics”*

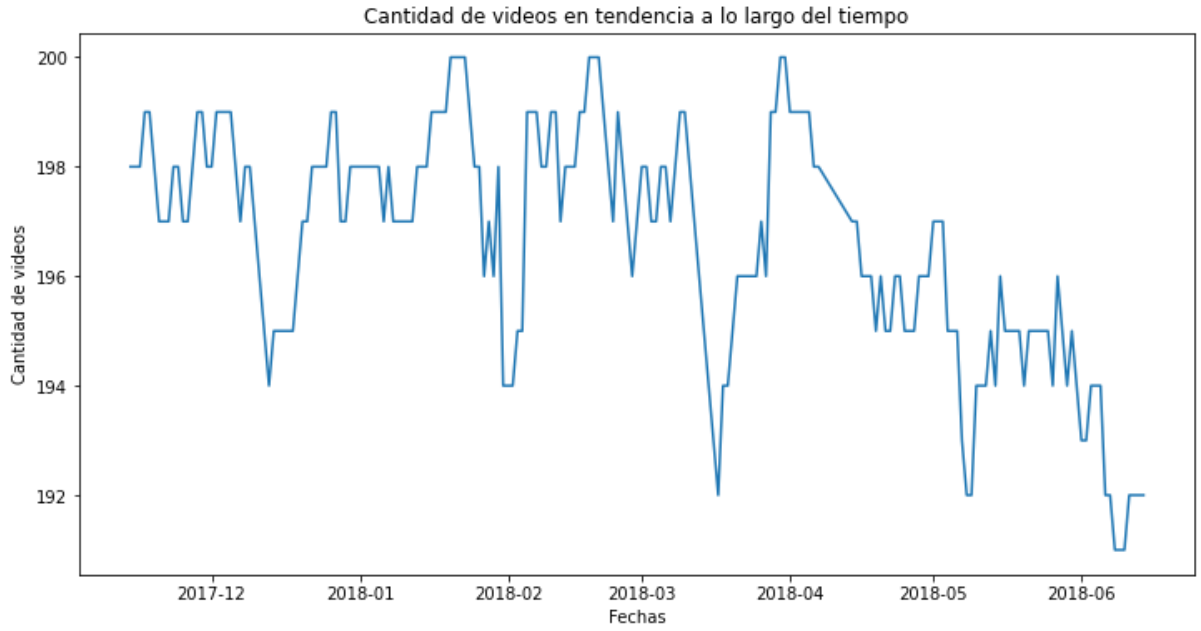


Categoría	**Ratio views/comentarios**
Gaming	175.03
Education	216.94
Howto & Style	224.74
Comedy	228.69
News & Politics	246.93
Nonprofits & Activism	273.11
People & Blogs	283.77
Pets & Animals	287.39
Science & Technology	307.29
Travel & Events	376.91
Entertainment	378.72
Sports	398.70
Film & Animation	417.99
Music	420.27
Shows	541.45
Autos & Vehicles	663.77

▪ **Por el tiempo transcurrido**

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

Empleando matplotlib, podemos realizar un diagrama de línea el cual nos permitirá visualizar con mayor claridad la cantidad de videos en tendencia a lo largo del tiempo. Obtenemos el siguiente gráfico.

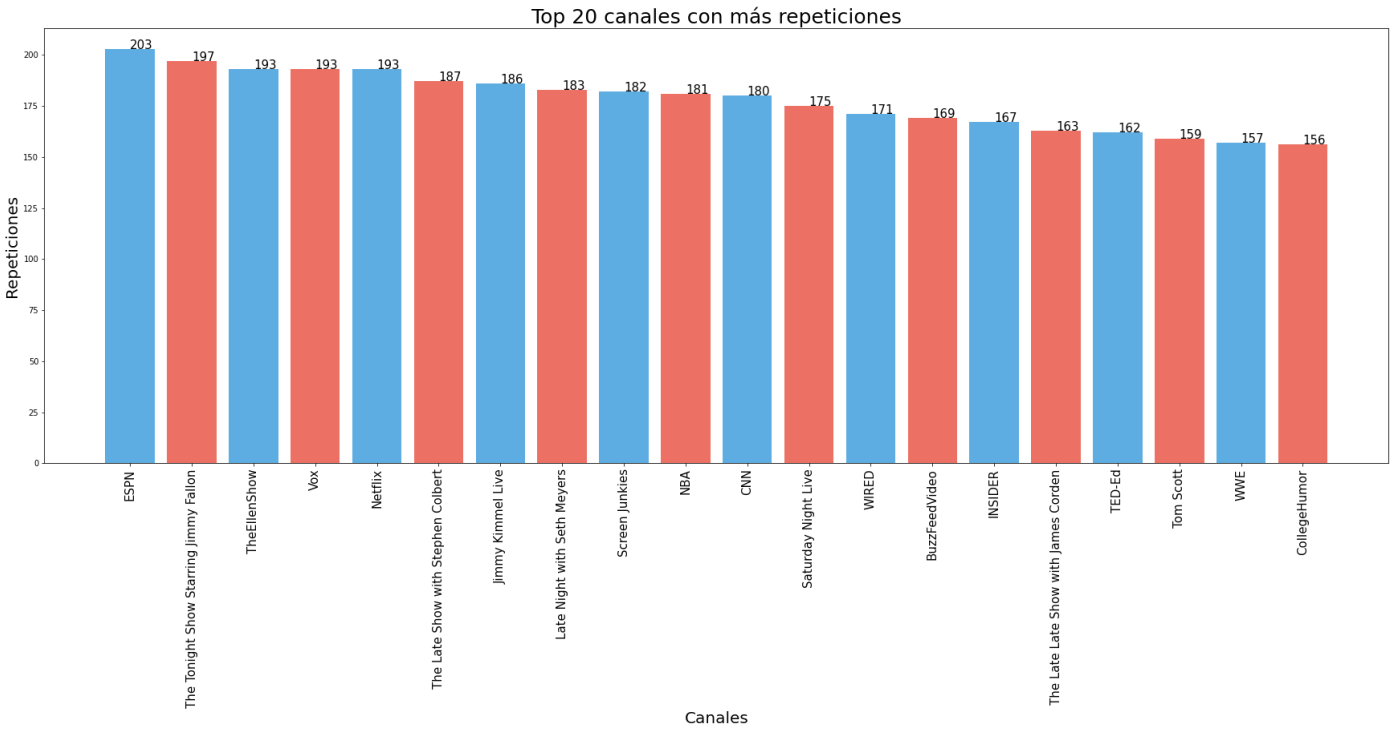


Podemos observar cómo a lo largo del tiempo, la cantidad de videos en tendencia presenta en diversos momentos unos altibajos. Sin embargo, podemos notar, cómo a partir de abril de 2018 el volumen disminuye considerablemente.

▪ **Por Canales de YouTube**

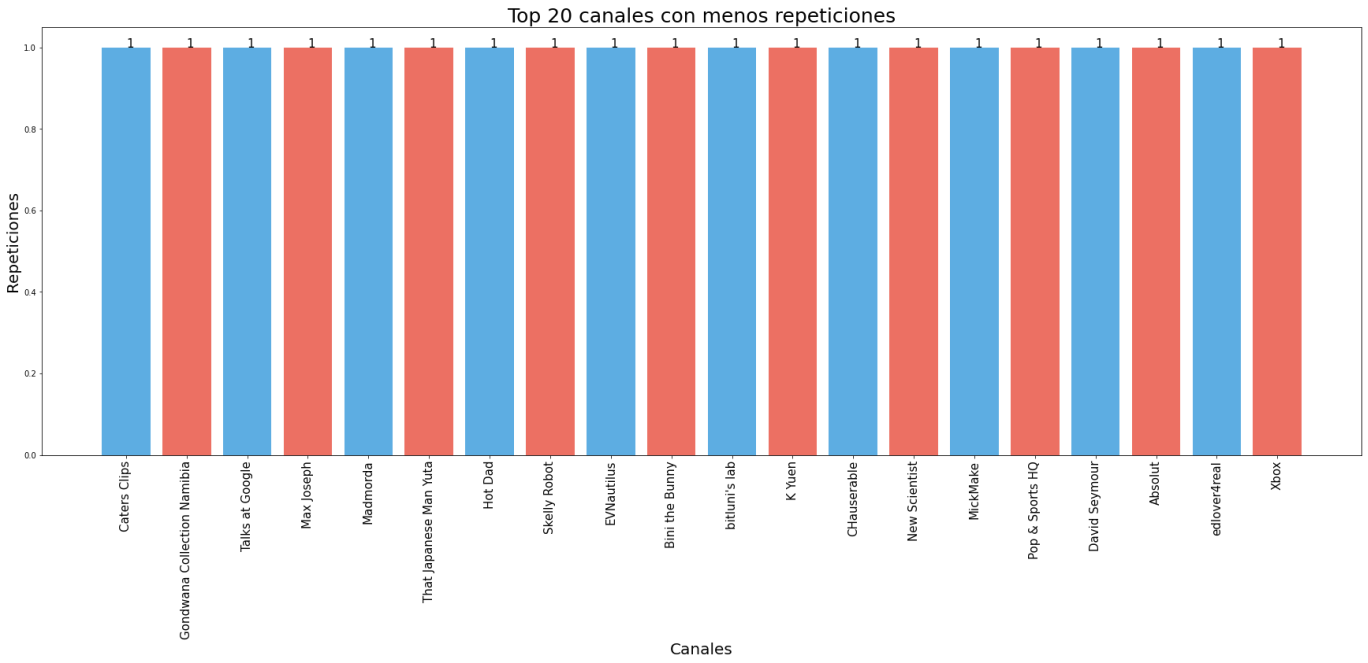
6. ¿Qué Canales de YouTube son tendencia más frecuentemente? ¿Y cuáles con menos frecuencia?

Para responder correctamente al enunciado, se decidió emplear un gráfico de barras, ya que este nos permite visualizar claramente cuáles canales presentan una frecuencia de tendencia mayor. Se optó por mostrar tanto los 20 canales que son tendencia más frecuentemente, como los 20 con menos.



Canales	**Repeticiones**
ESPN	203
The Tonight Show Starring Jimmy Fallon	197
TheEllenShow	193
Vox	193

Netflix	193
The Late Show with Stephen Colbert	187
Jimmy Kimmel Live	186
Late Night with Seth Meyers	183
Screen Junkies	182
NBA	181
CNN	180
Saturday Night Live	175
WIRED	171
BuzzFeedVideo	169
INSIDER	167
The Late Late Show with James Corden	163
TED-Ed	162
Tom Scott	159
WWE	157
CollegeHumor	156



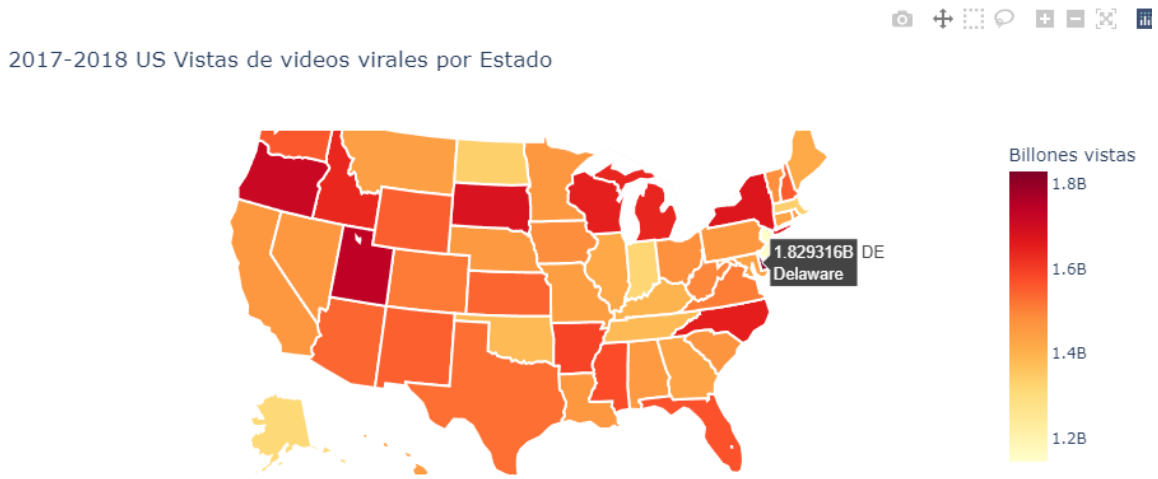
Canales	**Repeticiones**
Caters Clips	1
Gondwana Collection Namibia	1
Talks at Google	1
Max Josheph	1
Madmorda	1
That Japanese Man Yuta	1
Hot Dad	1
Skelly Robot	1
EVNautilus	1
Bini the Bunny	1
bitluni’s lab	1
K Yuen	1

CHauserable	1
New Scientist	1
MickMake	1
Pop & Sports HQ	1
David Seymour	1
Absolut	1
edlover4real	1
Xbox	1

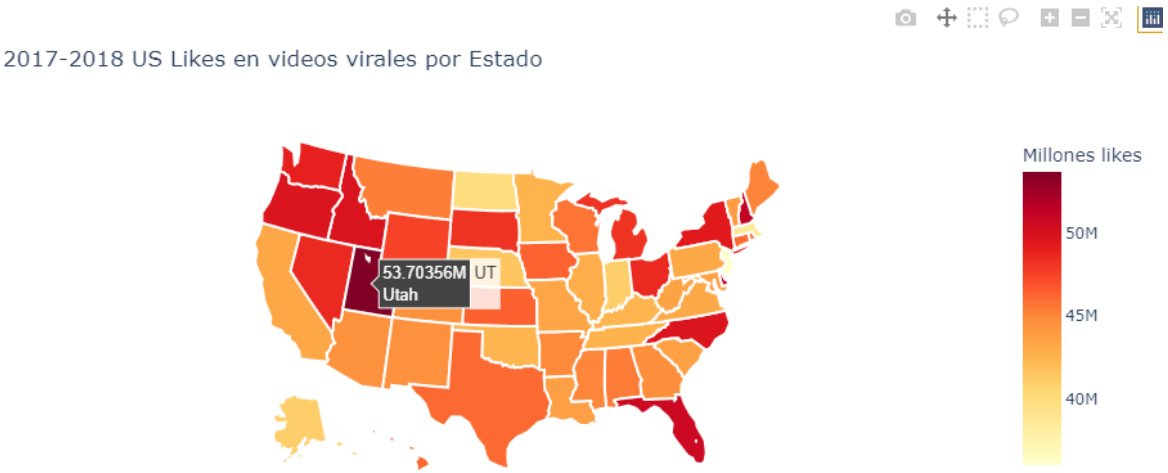
▪ **Por la geografía del país**

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

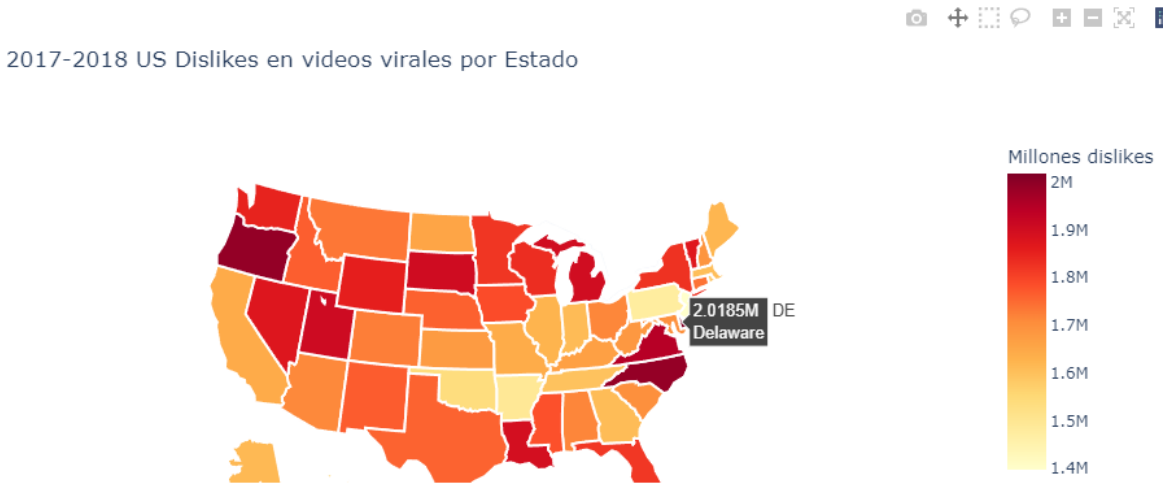
Para responder a esta incógnita se decidió hacer unos mapas de coropletas, los cuales reflejan las cantidades de cada variable a contar por estado.



En el caso de las “Vistas”que tienen los videos, se obtuvo como resultado que el estado de Delaware cuenta con el mayor conteo, con 1.83 Billones.



En el caso de los “Me gusta”, el estado de Utah tiene el cómputo mayor, con 53.7 Millones.



Finalmente se obtuvo que el estado de Delaware cuenta con 2.02 Millones de “No me gusta”, ocupando la primera posición.

Adicionalmente, al cliente le gustaría conocer si:

- ¿Es factible predecir el número de “Vistas” o “Me gusta” o “No me gusta”?

Para estas predicciones, se utilizaron modelos de regresión lineal, para una de las variables. De esta forma, se puede estimar el crecimiento de las.“Vistas” o “Me gusta” o “No me gusta”, según el transcurso después de la publicación del video.

En el caso de las *“Vistas”*, se obtuvieron los siguientes resultados, los cuales indican que por cada aumento en 1 de la variable, manteniendo las demás fijas, las vistas aumentan en el coeficiente.

	Coeficiente
likes	27.284816
dislikes	333.534504
comment_count	-97.200981

Para el caso de los *“Me gusta”*, se obtuvieron los siguientes coeficientes.

	Coeficiente
views	0.013350
dislikes	-1.402255
comment_count	6.191716

Finalmente para los *“No me gusta”*, estos son los coeficientes resultantes.

	Coeficiente
views	0.000580
likes	-0.004984
comment_count	0.203002

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

Para el correcto análisis de esta incógnita es necesario los datos de los comentarios, para de esta forma ejecutar la técnica de *Sentiment Analysis* para de esa forma clasificar los comentarios como positivos o negativos y contarlos.

Como alternativa, se decidió hacer un conteo de los videos que cuentan con una mayor cantidad de *"Me gusta"* en comparación de *"No me gusta"*.

Se creó una columna adicional para verificar lo anterior mencionado.

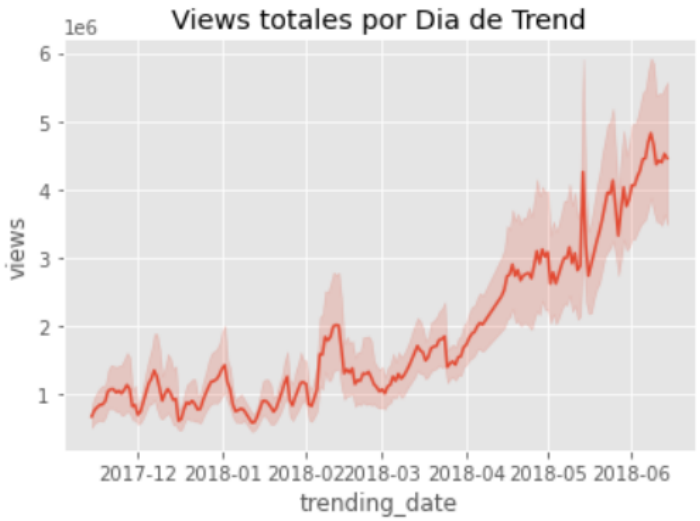
```
df['more_likes'] = np.where(df['likes'] > df['dislikes'], True,
False)
```

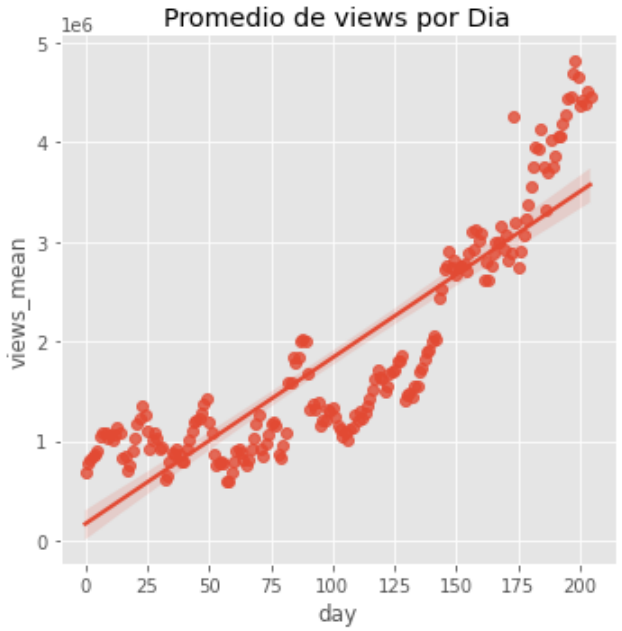
```
df['more_likes'].value_counts()
✓ 0.7s

True     39580
False      725
Name: more_likes, dtype: int64
```

Se aprecia en el cómputo que la proporción de vídeos con un impacto positivo en comparación a los de impacto negativo es muy superior. De esta forma, se puede afirmar que los videos en tendencia son los que cuentan con una recepción positiva.

5. Modelizar y Evaluar los Datos

Después de visualizar y analizar los datos, hemos llegado a una propuesta de modelo. Usando los días en que los videos fueron virales y la cantidad de vistas se obtuvieron los siguientes gráficos, siendo el primero las views totales por Día del Trend y el segundo el promedio de vistas por día, según un índice:





Este segundo gráfico nos permite observar la posibilidad de encontrar un modelo de Regresión Lineal. Sabiendo esto, utilizamos los algoritmos de la librería *"sklearn"* para conseguir el coeficiente y término independiente de la ecuación lineal.

```
    print("Las pendientes 'w1' del modelo son: ", slope)
    print("El término independiente de la recta 'w0' es: ", intercept)
 ✓  0.7s
Las pendientes 'w1' del modelo son:  [16279.59936329]
El término independiente de la recta 'w0' es:  161428.2685234337
```

Utilizando estos datos podemos formular una ecuación la cual permita predecir el promedio de views de videos virales para fechas después del 15/11/2017.

$$y = 16,279.60 * x + 161,428.27$$

$$y = promedio\ de\ views$$

$$x = cantidad\ de\ días\ del\ 15/11/2017\ al\ día\ actual$$

6. Conclusiones del Proyecto

Utilizando las herramientas y técnicas enseñadas en el curso, hemos llegado a las siguientes conclusiones:

- La categoría de vídeos con más apariciones en tendencias es *"Entertaiment"*
- La categoría de vídeos con más "Me gusta" es *"Music"*
- La categoría de vídeos con un mejor ratio entre "Me gusta" y "No me gusta" es *"Shows"*
- A partir de abril del 2018 el volumen de videos en tendencia disminuyó considerablemente.
- El canal con más videos que aparecieron en tendencias es ESPN.
- El estado con la mayor cantidad de vistas es Delaware.
- Gracias a un modelo de regresión lineal. si es factible predecir la cantidad de *"Vistas"*, *"Me gusta"* y *"No me gusta"*.
- Finalmente, gracias a la propuesta de modelo brindada, se facilita el hallar el promedio de vistas para fechas posteriores al 15/11/2017

7. Archivar y Publicar

La resolución del presente proyecto se puede contemplar en el siguiente repositorio en **Github**: [EB-2022-1-CC51 (github.com)]