Examen Final

Profesora: Patricia Reyes Silva

Integrantes del grupo: Natalia Maury Castañeda (u201816996)
Oscar Flores Palermo (u201716498)
Carlos Eduardo Iparraguirre Marujo (u201810601)

Fecha de entrega: 4 de julio del 2021
Ciclo: 2021-1

Contenido

Objetivo

El objetivo de este examen final es aplicar los conocimientos aprendidos durante todo el ciclo para obtener conocimiento de los datos luego de hacer un análisis exploratorio y resolver un problema de modelización de datos. Para ello, se tratará este trabajo como un proyecto de analítica. Las herramientas de software a utilizar son python y R/RStudio, así se aplican todos los conocimientos del curso. El dataset a utilizar es Trending YouTube Video Statistics.

Caso de análisis

Sobre el Dataset

El dataset a analizar es el de Trending YouTube Video Statistics, la versión original es de kaggle.com pero se ha modificado con la adición de 4 columnas adicionales (state, lat, lon, geometry) para este trabajo. No obstante, el origen de los datos es el mismo y tiene al mismo creador del conjunto de datos. El dataset contiene información sobre los videos subidos a YouTube para conocer diversas características de ellos respecto a su ubicación geográfica por países. Estas características son la cantidad de vistas que tienen, cantidad de me gusta y no me gusta, categoría del video, etc.

Este dataset tiene dos versiones, la primera fue creada en el 2017 y la segunda es en el 2019. La versión del 2017 contiene todas las características mencionadas anteriormente pero solo contiene los datos de Estados Unidos e Inglaterra (Mitchell; 2017). La versión del 2019 es la misma que la del 2017 (Mitchell; 2019), pero se le agregó el contenido de más países como México, Rusia, Japón, etc. Ambos datasets le pertenecen y fueron creados por la misma persona y es Mitchell Jolly.

El proceso de recolección de datos fue de muchos meses donde incluía los datos diarios de los videos top trending en youtube. Para poder tener acceso a estos datos, Mitchell usó el API de YouTube. Debido a que en su momento de creación del dataset él estudiaba en la universidad de Glasgow, se puede decir que su país de publicación es Inglaterra (Mitchell; 2019).

Sobre el autor

La persona que creó este dataset es Mitchell Jolly, un estudiante de Computing Science en la universidad de Glasgow en Inglaterra, donde pertenece a la facultad de ingeniería (Mitchell; S/F).

En el perfil de Kaggle de Mitchell, se puede observar que ha participado en competencias de contribución, de Datasets Experts y en discusiones de contribuidores. En las competencias de contribución no tiene ningún ranking (unranked) pero ha participado en 3 competencias. En la de Datasets Experts, su ranking actual es de 23 de 41646, y su mejor ranking fue de 6to lugar. Ha ganado 3 medallas de oro por sus publicaciones de Datasets. En las

discusiones de contribuidores, no tiene ningún ranking (unranked) pero tiene 10 medallas de bronce (Mitchell; 2017, Kaggle; N/A).

Casos de uso

El proyecto de analítica tiene como alcance conocer las tendencias de los videos para una consultora internacional con sede en Lima, por ende, los casos de uso serán relacionados a esto. Con el análisis del dataset, la consultora podrá clasificar los videos de diversas formas y saber con qué creador le conviene hacer negocios (publicidad, etc) basado en sus comentarios y estadísticas. Para una campaña de marketing, podrían usar este análisis para saber qué factores podrían interferir para hacer que un video esté en trending. Si luego de publicar algún video o se desea saber las reacciones a cierto contenido, se puede realizar un análisis emocional. La consultora podría tener más posibles casos de uso como saber en qué países les conviene publicar cierto contenido, etc, pero se han nombrado los principales casos de uso al análisis de este dataset.

Conjunto de datos

Nr	Variable	Descripción	Tipo de dato	Posibles valores de los datos
1	video_id	ID del video	String	Combinación de caracteres
2	trending_date	Fecha de tendencia del video	Date	Desde el año 2017 hasta el 2018
3	title	Titulo del video	String	Nombre aleatorio que seleccionó el youtuber video

4	channel_title	Nombre del canal del video	String	Nombre aleatorio que seleccionó el youtuber para su canal
5	category_id	ID de la categoría	String	Entre las 43 categorías
6	publish_time	Dia de la publicación del video	Date	Desde el año 2017 hasta el 2018
7	tags	etiquetas del video	String	Etiquetas que los youtubers escogen
8	views	Vistas del video	Número	Más de miles de vistas
9	likes	Me gustas del video	Número	más de miles de me gusta
10	dislikes	No me gustas del video	Número	más de miles no me gusta
11	comment_count	Cantidad de los comentarios	Número	cantidad de comentarios según su popularidad
12	thumbnail_link	enlace de la miniatura	String	Combinación de caracteres
13	comments_disabled	comentarios deshabilitados	Bool	verdadero o falso

14	ratings_disabled	clasificaciones deshabilitadas	Bool	verdadero o falso
15	video_error_or_removed	error del video o video removido	Bool	verdadero o falso
16	description	Descripcion del video	String	Descripción que el youtuber le puso al video
17	state	estado del video	String	Estados de México
18	lat	latitud de la ubicación de donde se publicó el video	Número	latitud de una parte de México
19	lon	longitud de la ubicación de donde se publicó el video	Número	longitud de una parte de México
20	geometry	Punto geográfico de donde se publicó el video	Número	Punto geográfico en México

Análisis exploratorio de datos

Para poder analizar los datos, primero se deben de cargar, inspeccionar , pre procesar y luego visualizar

Cargar datos

Para cargar los datos, se descargaron los archivos usando el link brindado por el enunciado del trabajo final. Debido a que uno de ellos tiene el formato json se utiliza la librería jsonlite de R. Luego se incluyen los archivos a leer en el proyecto y se usa el comando “read.csv” para el archivo csv y fromJSON para el archivo json. Para comprobar que se cargaron los datos correctamente, se utiliza el comando “View” para revisar los archivos.

```
#Carga de Datos
install.packages("jsonlite")
library(jsonlite)
DFVideos_MX <- read.csv("MXvideos_cc50_202101.csv", header = TRUE, stringsAsFactors = FALSE)
View(DFVideos_MX)
#Carga de Categorías
Categoria_MX <- fromJSON("MX_category_id.json")
id <- (Categoria_MX[["items"]][["id"]])
titulo <- (Categoria_MX[["items"]][["snippet"]][["title"]])
DFCategoria <- data.frame(id, titulo)
View(DFCategoria)
```

En este caso se decidió añadir las categorías según su id antes de guardar el primer dataset, pero debido a que el id 29 no se encontró en el json de México, se descargó el json de Estados Unidos para completarlo.

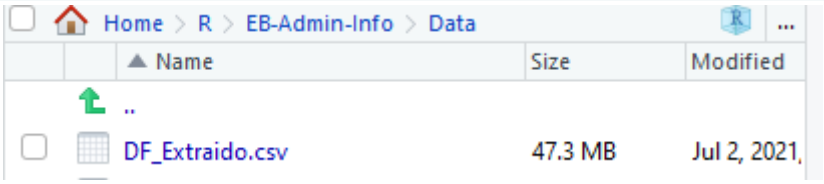
```
#No hay id 29 en DFCategoría, por lo que se carga el archivo US
Categoria_US <- fromJSON("US_category_id.json")
id <- (Categoria_US[["items"]][["id"]])
titulo <- (Categoria_US[["items"]][["snippet"]][["title"]])
DFCategoriaEx <- data.frame(id, titulo)
View(DFCategoriaEx)
```

Cuando ya se tienen todas las categorías, se usa una función para insertarlas en el dataframe original. Después se introduce las que tienen id 29 y las que no tenían id se clasificaron como “Undefined”. Además, se cambió el nombre de la columna video_id para mayor facilidad al referenciarla.

```
#Añadiendo Categorías al Dataframe
DFVideos_MX$category_id[is.na(DFVideos_MX$category_id)] <- 0
DFVideos_MX$category <- DFVideos_MX$category_id
n = 1
for (id in DFCategoria$id){
  DFVideos_MX$category <- replace(DFVideos_MX$category, DFVideos_MX$category == id, DFCategoria$titulo[n])
  n = n+1
}
DFVideos_MX$category[DFVideos_MX$category == "29"] <- DFCategoriaEx$titulo[17]
DFVideos_MX$category[DFVideos_MX$category == "0"] <- 'Undefined'
names(DFVideos_MX)[names(DFVideos_MX) == "ï..video_id"] <- "video_id"
```

Por último, se guarda el dataframe extraído tanto en formato csv como formato Rdata para su uso tanto en Python como R.

```
#Guardando el Dataframe Extraído en Rdata y csv
save(DFVideos_MX, file = "~/R/EB-Admin-Info/Data/DF_Extraido.RData")
write.csv(DFVideos_MX, file = "~/R/EB-Admin-Info/Data/DF_Extraido.csv", row.names = FALSE)
```



Inspeccionar datos

Primero se carga el dataset extraído mediante la carga de datos. Utilizando las funciones “nrow”, “ncol” y “colnames” podemos ver las dimensiones del dataset y el nombre de cada

atributo que columna que tiene para revisar qué datos incluye. Los resultados nos muestran que hay 44043 filas, 21 columnas y el nombre de cada una de ellas.

```
#Inspección de Datos
DFVideos_MX <- read.csv("~/R/EB-Admin-Info/Data/DF_Extraido.csv")
nrow(DFVideos_MX)
ncol(DFVideos_MX)
colnames(DFVideos_MX)
```

```
> DFVideos_MX <- read.csv("~/R/EB-Admin-Info/Data/DF_Extraido.csv")
> nrow(DFVideos_MX)
[1] 44043
> ncol(DFVideos_MX)
[1] 21
> colnames(DFVideos_MX)
 [1] "video_id"               "trending_date"          "title"                  "channel_title"
 [5] "category_id"            "publish_time"           "tags"                   "views"
 [9] "likes"                  "dislikes"               "comment_count"          "thumbnail_link"
[13] "comments_disabled"      "ratings_disabled"       "video_error_or_removed" "description"
[17] "state"                  "lat"                    "lon"                    "geometry"
[21] "category"
```

Para saber más de la estructura de cada tipo de dato se usa la función “summary” y “str”. Con “str” se ve una pequeña muestra de los datos, mientras que con “summary” se ve información más precisa en cuanto a datos numéricos.

```
> str(DFVideos_MX)
'data.frame':   44043 obs. of  21 variables:
 $ video_id              : chr  "SbOwzAl9ZfQ" "klOV6Xh-DnI" "6L2ZF7Qzsbk" "hcY52MFWMDM" ...
 $ trending_date         : chr  "17.14.11" "17.14.11" "17.14.11" "17.14.11" ...
 $ title                 : chr  "CapÃtulo 12 | MasterChef 2017" "ALEXA EX-INTEGRANTE DEL GRUPO TIMBIRICHE RENUN
CIA A Â¨La Voz Mexico 7Â¨TELEVISA 11/11/2017" "LOUIS CKAGÃ“ - EL PULSO DE LA REPÃšBLICA" "Sismo de 6.7 sacude C
osta Rica 12 Noviembre 2017" ...
 $ channel_title         : chr  "MasterChef 2017" "Micky Contreras Martinez" "El Pulso De La RepÃºblica" "Casan
are" ...
 $ category_id           : int  24 22 25 25 26 10 22 25 24 24 ...
 $ publish_time          : chr  "13/11/2017 1:06" "13/11/2017 0:11" "13/11/2017 12:00" "12/11/2017 22:47" ...
 $ tags                  : chr  "MasterChef Junior 2017|\"TV Azteca\"|\"recetas\"|\"cocina\"|\"Anette Michel\"|
\"Betty VÃ¡zquez\"|\"Benito Molin"| __truncated__ "La Voz Mexico 7" "Chumel Torres|\"El Pulso de la Republica\"
|\"noticias\"|\"news\"|\"mexico\"|\"pri\"|\"pan\"|\"prd\"|\"morena\"|"| __truncated__ "temblor|\"costa rica\"|\
"sismo en costa rica\"" ...
 $ views                 : int  310130 104972 136064 96153 499965 129645 75197 142522 371110 1111314 ...
 $ likes                 : int  4182 271 10105 378 57781 506 2277 7745 20155 83582 ...
 $ dislikes              : int  361 174 266 171 681 67 69 659 912 2194 ...
 $ comment_count         : int  1836 369 607 208 7428 239 164 0 766 9106 ...
 $ thumbnail_link        : chr  "https://i.ytimg.com/vi/SbOwzAl9ZfQ/default.jpg" "https://i.ytimg.com/vi/klOV6X
h-DnI/default.jpg" "https://i.ytimg.com/vi/6L2ZF7Qzsbk/default.jpg" "https://i.ytimg.com/vi/hcY52MFWMDM/default
.jpg" ...
 $ comments_disabled     : chr  "FALSO" "FALSO" "FALSO" "FALSO" ...
 $ ratings_disabled      : chr  "FALSO" "FALSO" "FALSO" "FALSO" ...
 $ video_error_or_removed: chr  "FALSO" "FALSO" "FALSO" "FALSO" ...
 $ description           : chr  "Disfruta la presencia del Chef Torreblanca en MasterChef, el duelo de pasteles
, las mesas regionales de los che"| __truncated__ "ALEXA EX-INTEGRANTE DEL GRUPO TIMBIRICHE RENUNCIA A Â¨La Voz
 Mexico 7Â¨TELEVISA 11/11/2017" "La cancIÃ³n del principio se llama â\200œEste espÃrituâ\200\235 de nuestros ca
rnales de nuestros carnales de C"| __truncated__ "El video es de un Walmart en el pais centroamericano en el mo
mento del temblor\\n\\nEl movimiento telÃºrico se "| __truncated__ ...
 $ state                 : chr  "Tlaxcala" "Michoacan" "Guerrero" "Hidalgo" ...
 $ lat                   : num  19.3 19.7 17.5 20.2 19.8 ...
 $ lon                   : num  -98.2 -100.6 -99.5 -98.7 -104.4 ...
 $ geometry              : chr  "POINT (-98.2300096 19.31999514)" "POINT (-100.569996 19.67997316)" "POINT (-99
.5000096 17.54997398)" "POINT (-98.73003076000001 20.17043418)" ...
 $ category              : chr  "Entertainment" "People & Blogs" "News & Politics" "News & Politics" ...
```

```
> summary(DFVideos_MX)
   video_id         trending_date         title           channel_title       category_id
 Length:44043       Length:44043       Length:44043       Length:44043       Min.   : 0.00
 Class :character   Class :character   Class :character   Class :character   1st Qu.:17.00
 Mode  :character   Mode  :character   Mode  :character   Mode  :character   Median :23.00
                                                                            Mean   :19.29
                                                                            3rd Qu.:24.00
                                                                            Max.   :43.00

 publish_time           tags               views               likes             dislikes
 Length:44043       Length:44043       Min.   :      157   Min.   :      0   Min.   :     0.0
 Class :character   Class :character   1st Qu.:    16813   1st Qu.:    299   1st Qu.:    17.0
 Mode  :character   Mode  :character   Median :    56973   Median :   1246   Median :    63.0
                                       Mean   :   342382   Mean   :  15862   Mean   :   747.2
                                       3rd Qu.:   206894   3rd Qu.:   7226   3rd Qu.:   267.0
                                       Max.   :100912384   Max.   :4470923   Max.   :1353667.0
                                       NA's   :3592        NA's   :3592      NA's   :3592
 comment_count     thumbnail_link     comments_disabled  ratings_disabled   video_error_or_removed
 Min.   :     0   Length:44043       Length:44043       Length:44043       Length:44043
 1st Qu.:    42   Class :character   Class :character   Class :character   Class :character
 Median :   196   Mode  :character   Mode  :character   Mode  :character   Mode  :character
 Mean   :  2040
 3rd Qu.:   885
 Max.   :905925
 NA's   :3592
 description           state                lat              lon            geometry           category
 Length:44043       Length:44043       Min.   :-16.43   Min.   :-116.01   Length:44043       Length:44043
 Class :character   Class :character   1st Qu.: 18.92   1st Qu.:-103.50   Class :character   Class :character
 Mode  :character   Mode  :character   Median : 20.38   Median : -99.67   Mode  :character   Mode  :character
                                       Mean   : 21.48   Mean   :-100.24
                                       3rd Qu.: 22.73   3rd Qu.: -98.23
                                       Max.   : 30.76   Max.   : -86.71
```

Pre-procesamiento de datos

Para poder manipular los datos, se debe hacer un pre procesamiento para limpiar los datos.

Datos faltantes (NA)

Primero se deben identificar los valores nulos o NA y reemplazarlos. Para identificar los valores NA en el dataset, se usó la función is.na(). Luego, se creó otro dataframe llamado “ValoresVacios” para poder ver con la función summary() cuántas columnas tenían valores NA.

```
ValoresVacios <- is.na(DFVideos_MX)
View(ValoresVacios)
summary(ValoresVacios)
```

Luego de hacer summary(ValoresVacios), se puede observar que las columnas views, likes, dislikes y comment_count tienen valores NA.

```
i..video_id     trending_date     title         channel_title   category_id     publish_time       tags
Mode :logical   Mode :logical   Mode :logical   Mode :logical   Mode :logical   Mode :logical   Mode :logical
FALSE:44043     FALSE:44043     FALSE:44043     FALSE:44043     FALSE:44043     FALSE:44043     FALSE:44043

  views           likes           dislikes        comment_count   thumbnail_link  comments_disabled ratings_disabled
Mode :logical   Mode :logical   Mode :logical   Mode :logical   Mode :logical   Mode :logical     Mode :logical
FALSE:40451     FALSE:40451     FALSE:40451     FALSE:40451     FALSE:44043     FALSE:44043       FALSE:44043
TRUE :3592      TRUE :3592      TRUE :3592      TRUE :3592
video_error_or_removed description      state           lat             lon             geometry        category
Mode :logical          Mode :logical   Mode :logical   Mode :logical   Mode :logical   Mode :logical   Mode :logical
FALSE:44043            FALSE:44043     FALSE:44043     FALSE:44043     FALSE:44043     FALSE:44043     FALSE:44043
```

Para reemplazar estos valores se utilizará un valor aleatorio por lo que se crearán dos funciones: una función que recibe la variable a reemplazar los valores NA con uno aleatorio

y otra función que le asigne valores aleatorios a las columnas enviadas del dataframe. Los datos limpios serán almacenados en un DataFrame llamado DFMexico_Limpio.

Para la función de valores aleatorios, se creó una función llamada rand.valor(x). Los parámetros que recibe son la variable que desea reemplazar con un valor aleatorio. La variable faltantes es un vector booleano que contiene los valores que son NA de la columna enviada, tot.faltantes es la suma de todos los datos que son NA del vector faltantes. El vector x.obs contiene todas los valores de la columna que no son NA, este valor se le asigna a la variable valorado. Finalmente, se reemplazan los valores que se encuentren en valorado[faltantes] con valores aleatorios utilizando sample()[1].

```
rand.valor <- function(x){
  faltantes <- is.na(x)
  tot.faltantes <- sum(faltantes)
  x.obs <- x[!faltantes]
  valorado <- x
  valorado[faltantes] <- sample(x.obs, tot.faltantes, replace = TRUE)
  return (valorado)
}
```

La función random.df(df, cols) utilizará la función rand.valor(x) para reemplazar los valores NA con valores aleatorios a las columnas enviadas del dataframe. Esta función recibe el dataframe y las columnas que se van a reemplazar. El vector nombres contiene los nombres del dataframe enviado. En el for lo que se hace es un bucle en todas las columnas del dataframe, ahí la variable nombre contendrá el nuevo nombre de la columna del dataframe. Finalmente, se busca las columnas en el df[nombre] y se les asignará un nuevo valor llamando a la función rand.valor(df[,col]) donde se le enviará la columna.

```
random.df <- function(df, cols){
  nombres <- names(df)
  for (col in cols) {
    nombre <- paste(nombres[col], sep = ".")
    df[nombre] <- rand.valor(df[,col])
  }
  df
}
```

Finalmente, para comprobar que la función ha reemplazado todos los valores NA se crea un dataframe llamado Comprobacion_Aleatorio que contendrá los valores NA del dataframe DFMexico_Limpio utilizando la función is.na(). Luego se hace summary(Comprobacion_Aleatorio) para verificar que se cambiaron de forma efectiva con el valor aleatorio las columnas 8, 9, 10 y 11.

```
DFMexico_Limpio <- random.df(DFVideos_MX, c(8,9,10,11))
Comprobacion_Aleatorio <- is.na(DFMexico_Limpio)
summary(Comprobacion_Aleatorio)
```

Se puede observar que el DataFrame ya no tiene ningún valor NA

[1] sample() es una función que recibe la cantidad de valores a reemplazar (tot.faltantes) y de dónde sacará los valores aleatorios (x.obs).

```
i..video_id     trending_date     title         channel_title   category_id     publish_time       tags
Mode :logical   Mode :logical   Mode :logical   Mode :logical   Mode :logical   Mode :logical   Mode :logical
FALSE:44043     FALSE:44043     FALSE:44043     FALSE:44043     FALSE:44043     FALSE:44043     FALSE:44043
  views           likes          dislikes       comment_count   thumbnail_link  comments_disabled ratings_disabled
Mode :logical   Mode :logical   Mode :logical   Mode :logical   Mode :logical   Mode :logical      Mode :logical
FALSE:44043     FALSE:44043     FALSE:44043     FALSE:44043     FALSE:44043     FALSE:44043        FALSE:44043
video_error_or_removed description       state            lat             lon           geometry        category
Mode :logical          Mode :logical   Mode :logical   Mode :logical   Mode :logical   Mode :logical   Mode :logical
FALSE:44043            FALSE:44043     FALSE:44043     FALSE:44043     FALSE:44043     FALSE:44043     FALSE:44043
```

En un principio se pensó en utilizar el promedio, pero la diferencia de valores crearía un resultado no realista. Un ejemplo es con los likes, el valor máximo es de 4470923 likes y el mínimo es de 0, por lo que su promedio sería 15862. Esto no hace sentido ya que un video con 500 vistas no puede tener 15862, es por ello que se tomó la decisión de reemplazar los valores con valores aleatorios.

Limpieza de datos vacíos

En el dataset habían ciertos valores que no tenían información pero no estaban catalogados como NA. Para esos valores se tuvo que buscar de forma más precisa cuáles eran los valores que no aportan información para rellenarlos o removerlos en el caso del video_id. En el caso del video_id, hay dos valores que se consideran vacíos. Estos son “” y “\\n”, por lo que todas las filas que existan con uno de esos dos video_id es removida del dataset.

```r
#Eliminando filas con video_id vacíos
View(DFMexico_Limpio)
Vacios <- DFMexico_Limpio$video_id == ""
summary(Vacios)
DFMexico_Limpio <- subset(DFMexico_Limpio, video_id != "")
Vacios <- DFMexico_Limpio$video_id == ""
summary(Vacios)

N <- DFMexico_Limpio$video_id == "\\n"
summary(N)
DFMexico_Limpio <- subset(DFMexico_Limpio, video_id != "\\n")
N <- DFMexico_Limpio$video_id == "\\n"
summary(N)
```

En esta función se usó “subset”, que se encarga de seleccionar todas las filas que cumplan su requerimiento. Con “summary” se comprueba que ya no hay ningún valor “” o “\\n” en video_id. Solo para mantener el orden en el número de filas, se usa row.names = NULL.

```r
row.names(DFMexico_Limpio) <- NULL
```

Ahora se llenarán los datos vacíos. Primero se inicia con la descripción y los links al thumbnail. Para no distorsionar mucho los datos, se reemplazan los datos vacíos con valores neutrales como “Link inexistente” o “Sin descripción”. Después se usa “table” en otras columnas para ver si tienen valores vacíos.

```r
#Llenando valores vacíos
Vacios <- DFMexico_Limpio$description == ""
summary(Vacios)
DFMexico_Limpio$description <- replace(DFMexico_Limpio$description, Vacios, "Sin descripción")

Vacios <- DFMexico_Limpio$thumbnail_link == ""
summary(Vacios)
DFMexico_Limpio$thumbnail_link <- replace(DFMexico_Limpio$thumbnail_link, Vacios, "Link Inexistente")

table(DFMexico_Limpio$ratings_disabled)
table(DFMexico_Limpio$comments_disabled)
table(DFMexico_Limpio$video_error_or_removed)
```

Las tres tablas incluyen valores vacíos, verdaderos y falsos. En este caso se decidió reemplazar los vacíos con el valor “Desconocido” usando la función “replace”.

```
Vacios <- DFMexico_Limpio$ratings_disabled == ""
summary(Vacios)
DFMexico_Limpio$ratings_disabled <- replace(DFMexico_Limpio$ratings_disabled, Vacios, "DESCONOCIDO")

Vacios <- DFMexico_Limpio$comments_disabled == ""
summary(Vacios)
DFMexico_Limpio$comments_disabled <- replace(DFMexico_Limpio$comments_disabled, Vacios, "DESCONOCIDO")

Vacios <- DFMexico_Limpio$video_error_or_removed == ""
summary(Vacios)
DFMexico_Limpio$video_error_or_removed <- replace(DFMexico_Limpio$video_error_or_removed, Vacios, "DESCONOCIDO")
```

Por último se reemplazan los valores vacíos en otras columnas con más valores neutrales.

```
Vacios <- DFMexico_Limpio$tags == ""
summary(Vacios)
DFMexico_Limpio$tags <- replace(DFMexico_Limpio$tags, Vacios, "Sin tags")

Vacios <- DFMexico_Limpio$tags == "[none]"
summary(Vacios)
DFMexico_Limpio$tags <- replace(DFMexico_Limpio$tags, Vacios, "Sin tags")

Vacios <- DFMexico_Limpio$title == ""
summary(Vacios)
DFMexico_Limpio$title <- replace(DFMexico_Limpio$title, Vacios, "Sin titulo")

Vacios <- DFMexico_Limpio$channel_title == ""
summary(Vacios)
DFMexico_Limpio$channel_title <- replace(DFMexico_Limpio$channel_title, Vacios, "Sin canal")
```

Datos atípicos (outliers)

Los datos atípicos son un gran problema para el análisis de datos por eso en el pre procesamiento tenias que ver por los outliers, si es que se encuentran para poder hacer algo con ellos. Por eso se comienza a revisar en todos los encabezados que sean de tipo numérico si se encuentran muy alejados, para eso se usa el boxplot y boxplot.stats

```
boxplot(DFMexico_Limpio$likes, horizontal = TRUE)
boxplot.stats(DFMexico_Limpio$likes)
```

Ahora que se puede visualizar, lo que tienes que hacer es reemplazar la variable en función del valor de la variable. Si está por debajo del percentil 5, use el promedio en su lugar, si es mayor que el percentil 95, use la mediana en su lugar. En el caso de la segunda función, reemplace el valor con un percentil de 0.05 o 0.95 dependiendo de si el valor está por encima del bigote superior o por debajo del bigote inferior.

Visualizar datos

Por categoría de videos

Categorías de videos que son tendencia mayormente

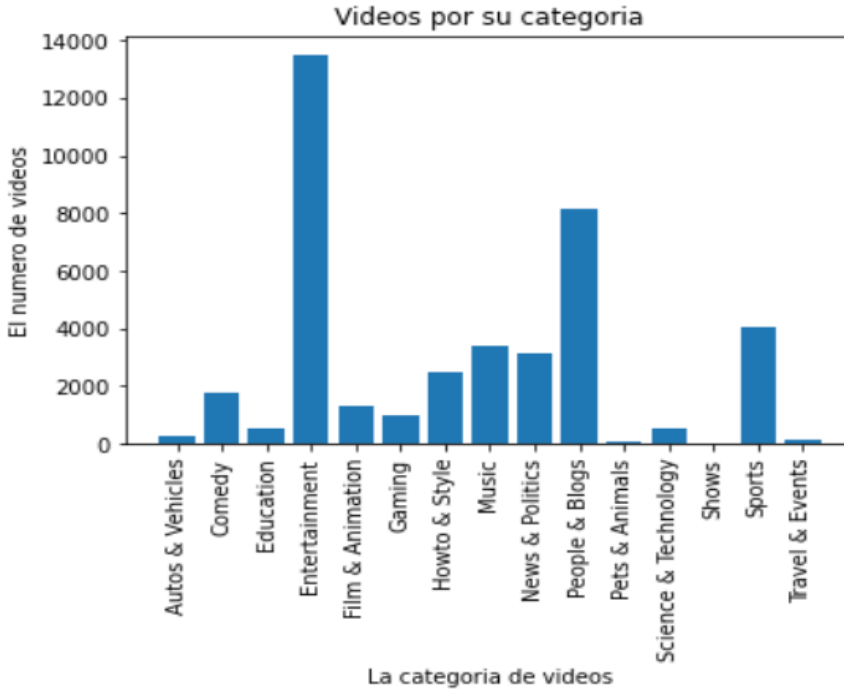
Se quiere saber las categorias de videos en que se tienen mayor tendencia para eso debemos visualizar los tipos de categorías que tienen los videos comparando con la cantidad de videos que se logran mostrar, primero se agrupa por su categoría

```
#Se guarda las categorias y se agrupan
categorias=DFMexico_Limpio.groupby("category").count()
```

Luego se hace la visualización de los datos con un bar

```
#se hace la visualizacion
plt.bar(categorias.index,categorias['category_id'])
plt.title("Videos por su categoria")
#Se pone en vertical para que se pueda ver la categoria del videc
plt.xticks(rotation='vertical')
plt.xlabel("La categoria de videos")
plt.ylabel("El numero de videos")
```

y finalmente se puede analizar



Como se puede visualizar se puede llegar a la conclusión que el “Entertainment” es lo que más tendencia en México por lo que se puede decir que en México les gusta el entretenimiento y el segundo más buscado son “people & blogs” es que le gusta ver la vida de las personas.

Categorías de videos con mayor cantidad de dislikes

Para saber qué categorías tienen la mayor cantidad de dislikes y la menor cantidad de dislikes, se hará un gráfico de barras. Primero se hará un lista que contenga todas las categorías con la cantidad de dislikes que tiene llamado categorias. Para ello se utiliza .unique() para tener todas las categorías sin que estén repetidas. Luego, se elimina una de ellas ya que aparece una que es Nan, y finalmente se le agrega otro elemento que contendrá la cantidad de dislikes.

```
categorias = [list(Mexico['category'].unique())]
# Removiendo el valor NaN
categorias[0].pop(9)
# Contadores
categorias.append([0 for _ in range(len(categorias[0]))])
```

Ahora se agregan la cantidad de dislikes utilizando un doble for para poder acceder al segundo elemento. En el primer for, se utiliza enumerate para obtener los valores cada vez que se itera, idx que será el índice/Valor para acceder a los nombres de la categoria y el elemento “categoria” que contiene el nombre de las categorías. En el segundo for, se compara si el nombre de la categoría evaluada es el mismo que el que aparece en Mexico[‘category’][‘dislikes] y si es así, se le suma el numerador que es “dislikes”.

```
for idx,categoria in enumerate(categorias[0]):
    for dislikes in Mexico[(Mexico['category'] == categoria)]['dislikes']:
        categorias[1][idx] += dislikes
    print(categoria,categorias[1][idx])
```

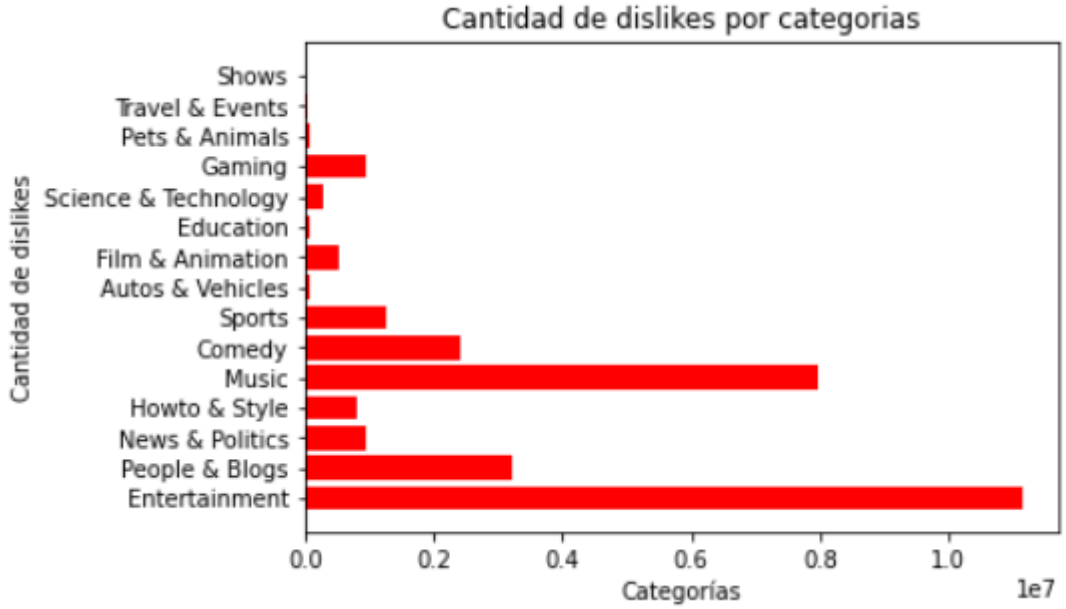
A partir de la información con el print(categorias[1][idx]) en el doble for anterios, se puede ver que la categoría con mayor cantidad de dislikes es Entertainment con 11139060 dislikes, y la de menor cantidad de dislikes es shows con 91 dislikes.

```
Entertainment 11139060
People & Blogs 3219558
News & Politics 957379
Howto & Style 792362
Music 7957335
Comedy 2400688
Sports 1279451
Autos & Vehicles 83752
Film & Animation 516653
Education 67408
Science & Technology 275862
Gaming 938239
Pets & Animals 63713
Travel & Events 26253
Shows 91
```

Con este nuevo dataframe se utilizará un gráfico de barras utilizando la función barh() para que salga el gráfico de barras horizontalmente. Los parámetros que recibe son el índice/nombre de la categoría, la cantidad de dislikes y el color de las barras.

```
eje_x = categorias[0]
eje_y = categorias[1]
plt.barh(eje_x, eje_y, color="red")
plt.xlabel('Categorías')
plt.ylabel('Cantidad de dislikes')
plt.title('Cantidad de dislikes por categorias')
```

Finalmente, esta es la visualización:



Categorías de videos con mayor cantidad de likes

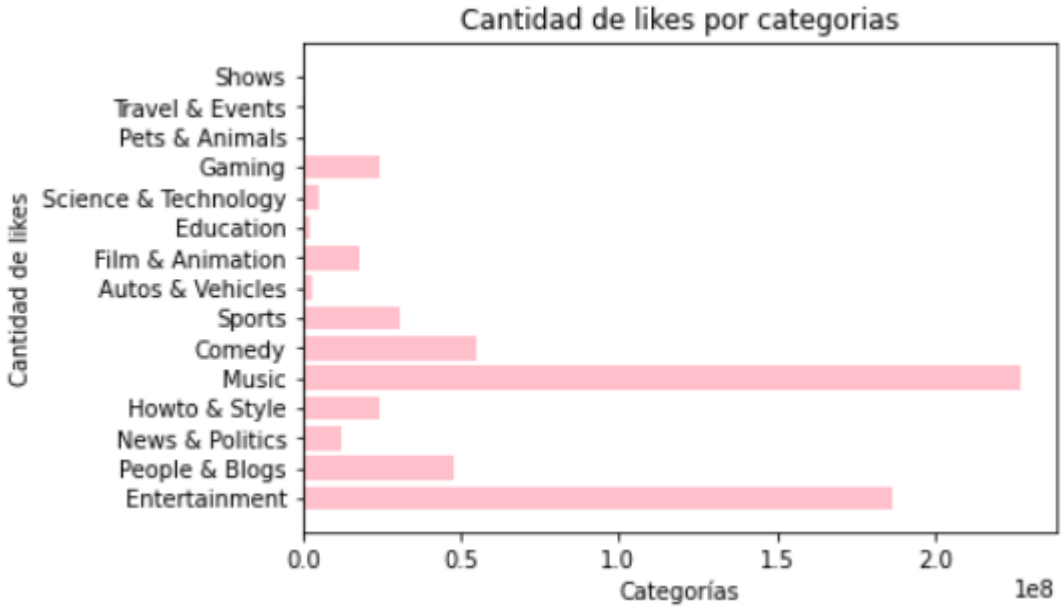
Para saber qué categorías tienen la mayor cantidad de likes y la menor cantidad de likes, se hará un gráfico de barras. Primero se hará un vector que contenga las categorias [2]y la cantidad de likes que tienen llamado categorias. Este se crea mediante el mismo proceso que en la visualización anterior.

A partir de la información al poner print(categorias[1][idx]) en el doble for, se puede ver que la categoría con mayor cantidad de likes es People & Blogs con 47859054 likes, y la de menor cantidad de likes es shows con 491 likes.

[2] La razón por la que se utiliza el link del .csv de los datos es porque daba error al poner el archivo

```
Entertainment 186692595
People & Blogs 47859054
News & Politics 11910414
Howto & Style 24651100
Music 227013300
Comedy 55273506
Sports 30866251
Autos & Vehicles 2980893
Film & Animation 17915248
Education 2550632
Science & Technology 5114569
Gaming 24002503
Pets & Animals 990508
Travel & Events 897709
Shows 491
```

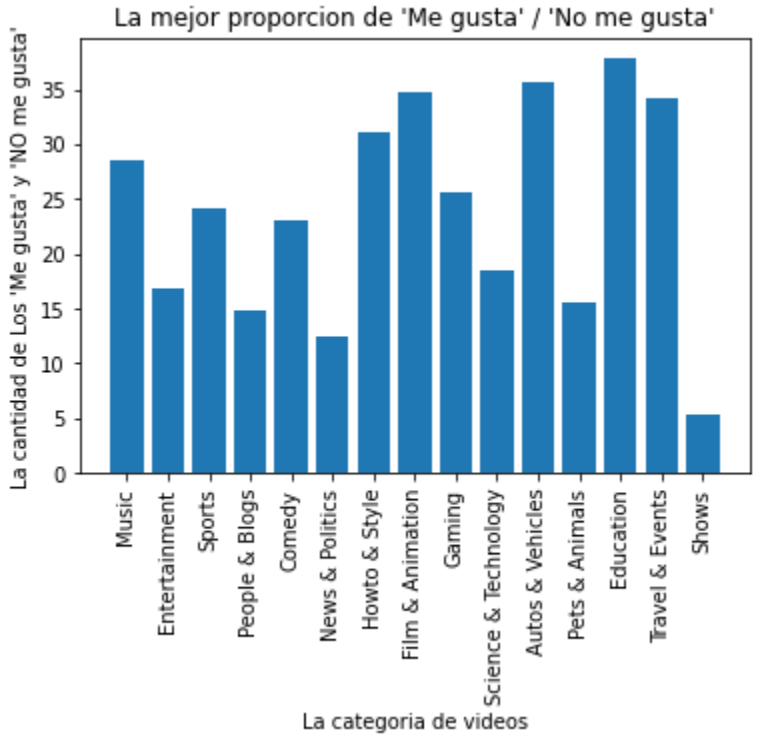
Finalmente la visualización es



Categorías con mejor proporción (ratio) de likes y dislikes

Para permitir la visualización de la proporción entre likes y dislikes, se va a crear un gráfico de barras. Primero se agruparon los datos por categoría y se sumaron la cantidad total de visitas. Después se obtiene el ratio para cada categoría y finalmente se grafica para permitir la visualización. De esta visualización se obtuvo que la categoría con mejor proporción de likes y dislikes es la de educación, mientras que la de menor proporción es la de shows.

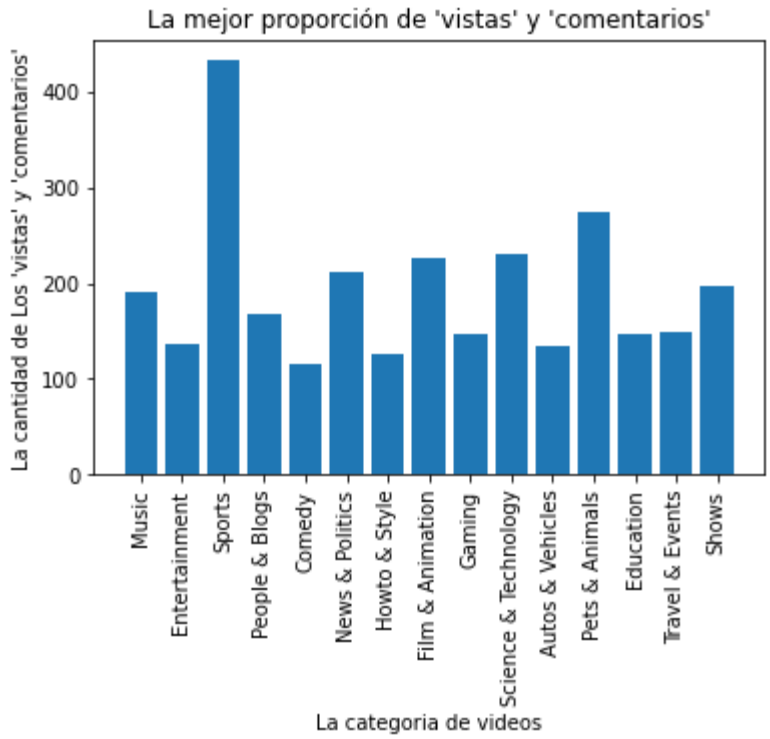
```
#Se hace una agrupacion por categoria y las vistas que llegan a tener
RatioLike=DFMexico_Limpio.groupby("category").sum().sort_values('views',ascending=False)
#se hace la visualizacion
plt.bar(RatioLike.index,(cat_pro_likes['likes']/RatioLike['dislikes']))
plt.title("La mejor proporcion de 'Me gusta' / 'No me gusta'")
#Se pone en vertical para que se pueda ver la categoria del video
plt.xticks(rotation='vertical')
plt.xlabel("La categoria de videos")
plt.ylabel("La cantidad de Los 'Me gusta' y 'NO me gusta'")
```



Categorías con mejor proporción (ratio) de vistas y comentarios

Para permitir la visualización de la proporción entre vistas y comentarios, se va a crear un gráfico de barras. Ya que al igual que el gráfico anterior se debe obtener un ratio y graficarlo, el procedimiento es muy similar, se divide los videos por categoría y de ahí se obtiene la proporción entre las visitas y los comentarios en cada categoría. Con los resultados adquiridos se ve que la categoría con mayor ratio es la de deportes, mientras que la de menor ratio sería la de comedia.

```
RatioCommentary=DFMexico_Limpio.groupby("category").sum().sort_values('views',ascending=False)
#se hace la visualizacion
plt.bar(RatioCommentary.index,(RatioCommentary['views']/RatioCommentary['comment_count']))
plt.title("La mejor proporción de 'vistas' y 'comentarios'")
plt.xticks(rotation='vertical')
plt.xlabel("La categoria de videos")
plt.ylabel("La cantidad de Los 'vistas' y 'comentarios'")
```



Por el tiempo transcurrido

Cambio del volumen de los videos en tendencia a lo largo del tiempo

Por canales de youtube

Estas visualizaciones se hacen con el propósito de saber qué canales son los que están en tendencia más frecuentemente y los que son tendencia menos frecuentemente. Para ello se crea una lista llamada canales que contiene los nombres de todos los canales y luego se agrega un segundo elemento a la lista para que contenga la cantidad de veces que fue tendencia. Luego en un doble for se cuenta la cantidad de veces que el canal estuvo en tendencia y se almacena.

```
url = "https://drive.google.com/file/d/1QB6Mjc8KC8RBM0GD2cs1ljMmuSSZiawN/view?usp=sharing"
url = "https://drive.google.com/uc?export=download&id="+url.split('/')[-2]

Mexico = pd.read_csv(url, encoding='latin1')
canales = [list(Mexico['channel_title'].unique())]
canales.append([0 for _ in range(len(canales[0]))])
```

```
for idx,canal in enumerate(canales[0]):
  for j in Mexico['channel_title'] == canal:
    canales[1][idx] += j
```

Una vez se tiene la cantidad de veces que cada canal ha sido tendencia, se busca el valor máximo y mínimo. Estos valores indican la máxima cantidad de veces que un canal(es) estuvo en tendencia y la mínima cantidad que un canal(es) estuvo en tendencia. A partir de estos valores, se crean 2 vectores llamados id_max e id_min que contienen los canales que cumplan con los valores de máximo y mínimo. A partir de estos vectores es que se hacen las visualizaciones a continuación.

```
print("La cantidad de veces que un canal es menos frecuente en tendencias es: ", min(canales[1]))
print("La cantidad de veces que un canal es más frecuente en tendencias es: ",max(canales[1]))
id_max = [] #Vector de los canales que son más tendencia
id_min = [] #Vector de los canales que no son tendencia
```

Canales en youtube en tendencia frecuentemente

Para esta visualización se utiliza el vector pid_max. Al poner len(id_max) se sabe que hay x canales que son los más frecuentes por lo que se hará un gráfico de barras y una tabla mostrando dichos canales.

Canales que son tendencia menos frecuentemente

Para esta visualización se utiliza el vector id_min. Al poner len(id_min) se sabe que hay x canales que son los menos frecuentemente tendencias por lo que se hará un gráfico de barras y una tabla mostrando dichos canales.

Por la geografía del país

El objetivo de estas visualizaciones es conocer las estadísticas de likes, dislikes y vistas de manera geográfica y cómo es que esta afecta el impacto de los videos.

Estados en que se presenta la mayor cantidad de likes, dislikes y vistas

Para saber qué estados presentan la mayor cantidad de vistas, se hizo una lista llamada estados que contiene el nombre de cada estado de México, luego se le agregó otro elemento que contiene la cantidad de vistas que tiene. Para ello se utiliza .unique() para tener todas los estados sin que estén repetidos.

```
estados = [list(Mexico['state'].unique())]
estados.append([0 for _ in range(len(estados[0]))])
```

Para almacenar los datos se crea un doble for. En el primer for, se utiliza enumerate para obtener los valores cada vez que se itera, idx que será el índice/Valor para acceder a los nombres de los estados y el elemento “region” que contiene el nombre de los estados. En el segundo for, se compara si el nombre del estado evaluado es el mismo que el que aparece en Mexico[‘state’][‘views] y si es así, se le suma el numerador que es “vistas”.

```
for idx,region in enumerate(estados[0]):
    for vistas in Mexico[(Mexico['state'] == region)]['views']:
        estados[1][idx] += vistas
    print(region,estados[1][idx])
```

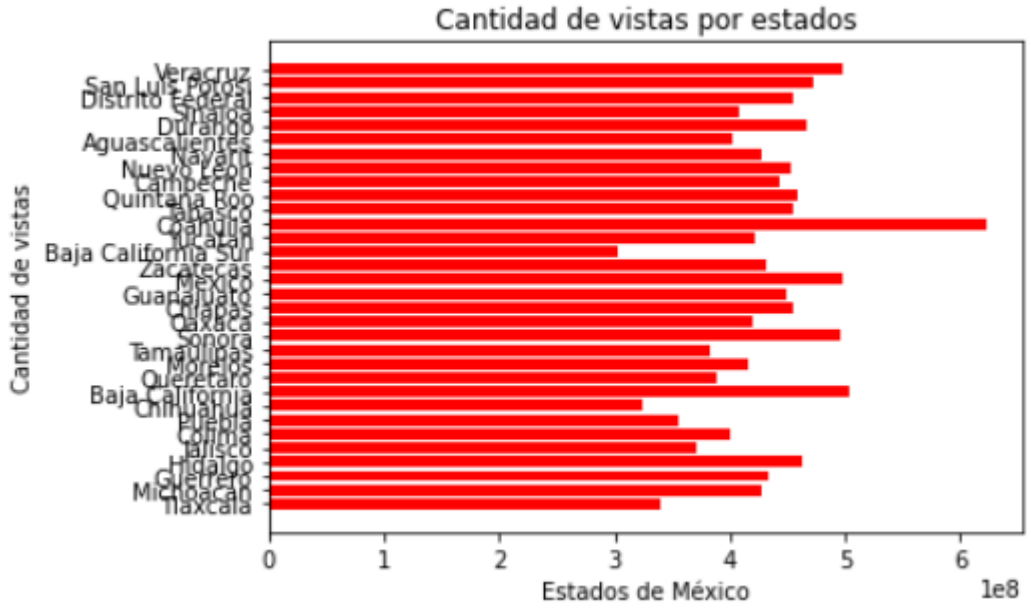
Con el print del for anterior, se obtienen los siguientes datos. Como se puede observar, el estado que tiene la mayor cantidad de vistas es Coahuila con 622718748 vistas. El estado que tiene la menor cantidad de vistas es Baja California Sur con 303374161 vistas.

```
Tlaxcala 340297637
Michoacan 427236181
Guerrero 434288782
Hidalgo 463274484
Jalisco 371324602
Colima 400271305
Puebla 356241284
Chihuahua 323788133
Baja California 503384409
Queretaro 388573340
Morelos 415970687
Tamaulipas 383459478
Sonora 495643062
Oaxaca 420783504
Chiapas 455488228
Guanajuato 448894670
Mexico 498357831
Zacatecas 431359321
Baja California Sur 303374161
Yucatan 422600895
Coahuila 622718748
Tabasco 455094066
Quintana Roo 459486058
Campeche 442934874
Nuevo Leon 452486387
Nayarit 427018835
Aguascalientes 402648634
Durango 467619850
Sinaloa 408458248
Distrito Federal 455518655
San Luis Potosi 472286493
Veracruz 498810152
```

Con esta lista se utilizará un gráfico de barras utilizando la función barh() para que salga el gráfico de barras horizontalmente. Los parámetros que recibe son el índice/nombre del estado, la cantidad de likes y el color de las barras.

```
eje_x = estados[0]
eje_y = estados[1]
plt.barh(eje_x, eje_y, color="red")
plt.xlabel('Cantidad de vistas')
plt.ylabel('Estados de Mexico')
plt.title('Cantidad de vistas por estados')
plt.show()
```

Finalmente la visualización es:



Para saber la cantidad de likes por estado se hizo el mismo proceso que para vistas, solo en el doble for se cambia Mexico[‘state’][‘views] por Mexico[‘state’][‘likes’]. Todo esto se almacena en un vector llamado estados que contiene el nombre de cada estado y su cantidad de likes.

```
for idx,region in enumerate(estados[0]):
    for likes in Mexico[(Mexico['state'] == region)]['likes']:
        estados[1][idx] += likes
    print(region,estados[1][idx])
```

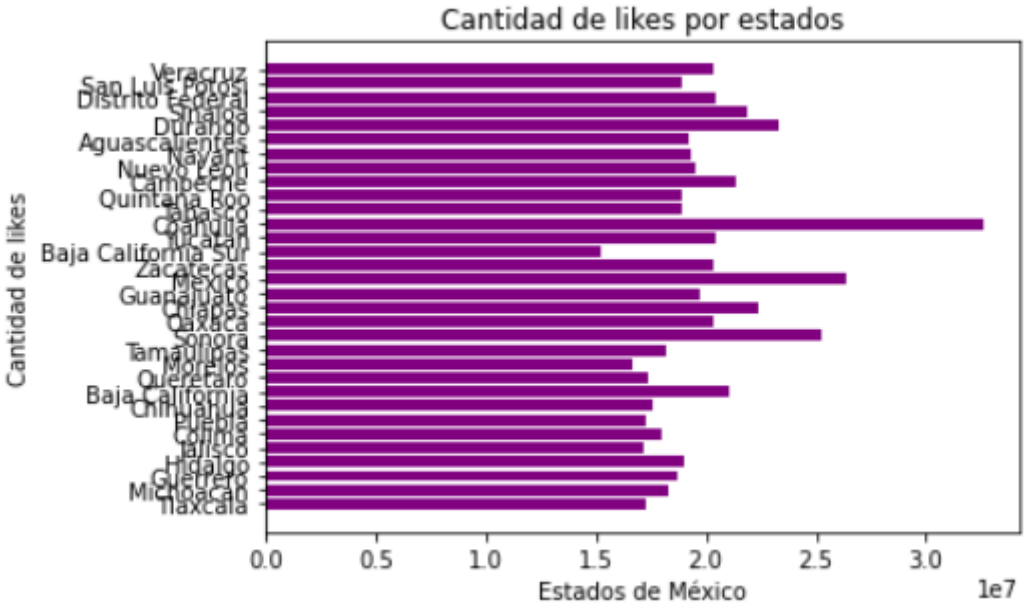
Con el print del doble for se obtienen los siguientes datos. El estado con la mayor cantidad de likes es Coahuila con 32590392 likes y el estado con menor cantidad de likes es Baja California Sur con 15276286 likes.

```
Tlaxcala 17245092
Michoacan 18318820
Guerrero 18736635
Hidalgo 18967555
Jalisco 17213627
Colima 18007111
Puebla 17268004
Chihuahua 17557558
Baja California 21090305
Queretaro 17351376
Morelos 16653171
Tamaulipas 18202903
Sonora 25286059
Oaxaca 20339380
Chiapas 22421954
Guanajuato 19684816
Mexico 26361211
Zacatecas 20384159
Baja California Sur 15276286
Yucatan 20486045
Coahuila 32590392
Tabasco 18941609
Quintana Roo 18879857
Campeche 21329902
Nuevo Leon 19559840
Nayarit 19363910
Aguascalientes 19198404
Durango 23314904
Sinaloa 21854911
Distrito Federal 20468000
San Luis Potosi 18926855
Veracruz 20346535
```

Con esta lista se utilizará un gráfico de barras utilizando la función barh() para que salga el gráfico de barras horizontalmente. Los parámetros que recibe son el índice/nombre del estado, la cantidad de likes y el color de las barras.

```
eje_x = estados[0]
eje_y = estados[1]
plt.barh(eje_x, eje_y, color="red")
plt.xlabel('Estados de México')
plt.ylabel('Cantidad de vistas')
plt.title('Cantidad de vistas por estados')
```

Finalmente, la visualización es:



Para la cantidad de dislikes se hizo el mismo procedimiento. Un vector estados que almacena los nombres de los estados y la cantidad de dislikes que tiene. En el doble for en vez de x usa x.

```
estados = [list(Mexico['state'].unique())]
estados.append([0 for _ in range(len(estados[0]))])


for idx,region in enumerate(estados[0]):
    for dislikes in Mexico[(Mexico['state'] == region)]['dislikes']:
        estados[1][idx] += dislikes
    print(region,estados[1][idx])
```

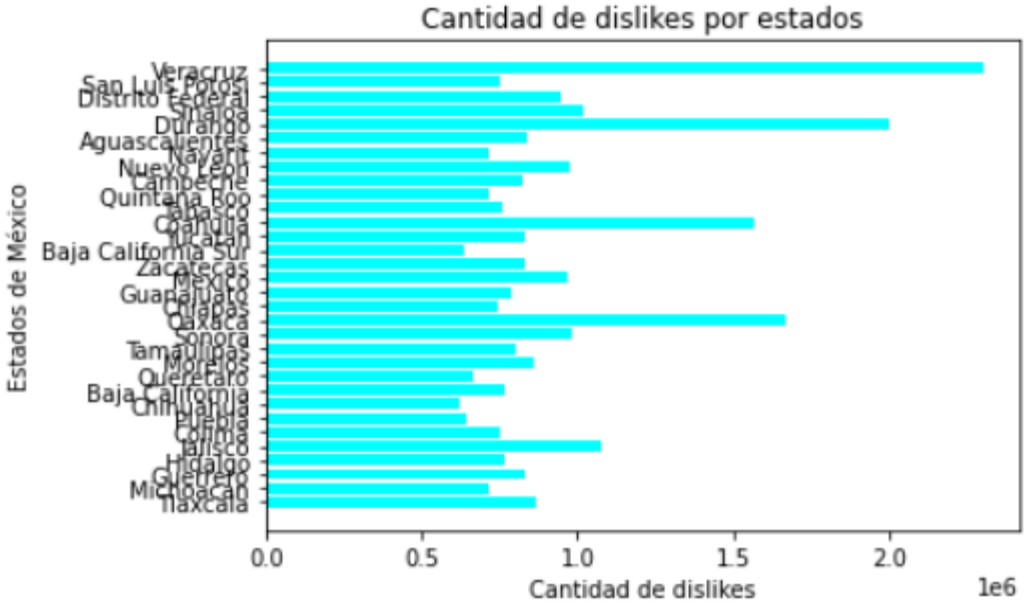
Con el print del doble for se obtienen los siguientes datos. El estado con la mayor cantidad de dislikes es Veracruz con 2302398 dislikes y el estado con menor cantidad de dislikes es Chihuahua con 617762 dislikes.

```
Tlaxcala 866418
Michoacan 715553
Guerrero 829653
Hidalgo 766856
Jalisco 1076955
Colima 750616
Puebla 641181
Chihuahua 617762
Baja California 762277
Queretaro 666235
Morelos 862391
Tamaulipas 803181
Sonora 981735
Oaxaca 1667885
Chiapas 743954
Guanajuato 785112
Mexico 966618
Zacatecas 830241
Baja California Sur 637922
Yucatan 828142
Coahuila 1570262
Tabasco 757687
Quintana Roo 715923
Campeche 826010
Nuevo Leon 973691
Nayarit 713405
Aguascalientes 840109
Durango 2000129
Sinaloa 1022085
Distrito Federal 949771
San Luis Potosi 751228
Veracruz 2302398
```

Las visualizaciones fueron hechas con barh().

```
eje_x = estados[0]
eje_y = estados[1]
plt.barh(eje_x, eje_y, color="cyan")
plt.xlabel('Cantidad de dislikes')
plt.ylabel('Estados de México')
plt.title('Cantidad de dislikes por estados')
plt.show()
```

Finalmente las visualizaciones son:



Modelizar y evaluar los datos

La consultora internacional desea saber si es factible predecir el tipo de comentarios en los videos en tendencia, y el número de vistas, “me gusta” y “no me gusta” en un video. Para ello se crearán dos modelos para responder si es factible o no hacer dichas predicciones de manera precisa.

Para poder predecir si los videos en tendencia son los que reciben más comentarios positivos, se debe hacer un modelo de regresión lineal. En este modelo, se compararán las variables de los comentarios positivos, negativos y neutrales con un análisis emocional, la cantidad de likes y dislikes. Una vez entrenadas dichas variables, se creará el modelo para poder predecir el tipo de comentario (positivo, negativo, neutral) que reciben los modelos en tendencia.

Para poder predecir el número de vistas, “me gusta” y “no me gusta” se debe hacer un modelo de regresión logística. Para ello, se utilizarán las categorías a la que pertenecen los videos, fecha de publicación, título y las etiquetas del video. Una vez entrenadas dichas variables, se creará el modelo para poder predecir la cantidad de vistas, “me gusta” y “no me gusta” del video.

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

Primero debemos analizar los datos si es una buena idea hacer una predicción de los “Vistas” o “Me gusta” o “No me gusta” de los videos, para eso vemos los datos que tenemos

una vez ya vistos entonces debemos ver si es factible el análisis de los datos para eso debemos mirar sus coeficientes

e interpretarlos en una tabla de regresión lineal para poder ver los datos y que posibles predicciones se pueden hacer

cómo se pueden visualizar en la tabla de regresión lineal se puede ver los datos que al principio puede ser una predicción.

Conclusiones

A partir de las visualizaciones realizadas se pueden obtener conclusiones que respondan a preguntas importantes de la consultora internacional. En cuanto a las categorías de videos, se pueden obtener las siguientes conclusiones. Basándose en los gráficos de likes y dislikes por categoría, se puede concluir que las categorías que más les gusta a las personas son music y comedy ya que tienen un gran volumen de likes que supera al de dislikes. En cuanto a qué categorías son las que les gustan menos a las personas, se puede concluir que shows, pets & animals y travel & events entran en esta categoría. Si bien shows tienen 491 likes y 90 dislikes, no tiene mucha interacción comparado con music o comedy, por lo que se puede concluir que le gusta a un grupo de personas pequeño pero no se puede considerar como las favoritas debido al poco volumen. Lo mismo aplica para pets & animals y travel & events, ya que todas tienen más likes que dislikes pero tienen muy poca gente que reacciona a los videos por lo que se puede concluir que son las categorías que menos le gustan a las personas. Entertainment tiene la mayor cantidad de dislikes, pero esta equivale al 5% de la cantidad de likes+dislikes que recibe por lo que se puede concluir que a la gente sí le gusta esta categoría por más que sea la que más dislikes recibe.

En cuanto a las proporciones de likes y dislikes y vistas y comentarios se obtiene que no necesariamente la categoría con mayor proporción en un área tendrá la mayor proporción en otra. Mientras que los fans de la categoría deportes suelen dejar más comentarios y vistas en relación, no hay una ratio muy grande en cuanto a likes y dislikes. Lo inverso sucede con la categoría educación, donde hay más variedad entre likes y dislikes, sin embargo, no se hallan muchos comentarios en esa categoría de videos.

En cuanto a la geografía por estado del país, se puede concluir que los estados con mayor cantidad de vistas son también los que tiene mayor cantidad de likes y dislikes, y viceversa para los que tienen la menor cantidad de vistas. Un ejemplo es Coahuila que tiene la mayor cantidad de vistas y la mayor cantidad de likes. Esto no es anormal, debido a que mientras más grande sean las vistas, tendrá mayores reacciones al video lo que permite que se de este fenómeno.

En el modelado se usó la regresión lineal que es un manera muy fácil de utilizar por lo que se pueden observar en el gráfico nos facilitó la obtención de datos. Se pudieron analizar los coeficientes de “Vistas” o “Me gusta” o “No me gusta” de los videos para un mejor entendimiento. Al final de cuentas resulta muy difícil predecir por sus valores tan cambiantes que tienen los datos de Mexico de youtube.

Archivar y publicar

Todo este análisis realizado, se encuentra en un repositorio creado en la página web de Github. En este repositorio hay 2 carpetas, una que se llama data y otra que se llama code. La carpeta data contiene el dataset original y el dataset final resultante. La carpeta code contiene los scripts en R y los notebooks en python utilizados para el proceso de carga, inspección, pre-procesado, visualización del dataset, y modelar y evaluar los datos.

Aparte de las carpetas, el repositorio deberá tener un archivo .Readme. Este archivo contiene el objetivo, nombre de los integrantes, breve descripción del dataset y el link del pdf, conclusiones y la licencia utilizada para el proyecto.

Link del repositorio: https://github.com/OscarFloresP/EB-2021-1-CC51
Nombre del repositorio: EB-2021-1-CC51

Bibliografía

Kaggle (S/F). *Kaggle Progression System*. Recuperado de https://www.kaggle.com/progression [Consultado el 29 de junio del 2021]

Mitchell. J (2017). *Mitchell J*. Recuperado de https://www.kaggle.com/datasnaek [Consultado el 29 de junio del 2021]

Mitchell. J (2019). *Trending YouTube Video Statistics: Daily statistics for trending YouTube videos*. Recuperado de https://www.kaggle.com/datasnaek/youtube-new [Consultado el 29 de junio del 2021]

Mitchell. J (2017). *Trending YouTube Video Statistics and Comments: Daily statistics (views, likes, category, comments+) for trending YouTube videos*. Recuperado de https://www.kaggle.com/datasnaek/youtube [Consultado el 29 de junio del 2021].

Mitchell.J (S/F). *Mitchell Jolly*. Recuperado de https://uk.linkedin.com/in/mitchell-jolly-2bab06169?trk=public_profile_samename-profile [Consultado el 29 de junio del 2021]

Wikipedia (2021). *Glasgow*. Recuperado de https://es.wikipedia.org/wiki/Glasgow [Consultado el 29 de junio del 2021]