

Práctica 2: Limpieza y análisis de los datos

Enrique Vilanova Vidal* Brais Suarez Souto

3 de enero de 2021

*Todos los que han colaborado en el proyecto

Notas previas al ejercicio:

Hemos contestado a las preguntas de acuerdo a lo indicado al ejercicio, sin embargo debido a la extensión del trabajo y del código, únicamente algunas partes del código y su desarrollo se han incluido en este texto. El objetivo es facilitar la comprensión del desarrollo de las diversas partes del trabajo. El código en su totalidad se puede encontrar en el enlace que se facilita a GitHub.

El código se ha hecho sobre *Google Colab* para facilitar la reproducción y ejecución del trabajo. La primera parte del código está implementado en *Python* y la segunda parte en *R*. Las dos partes se han implementado sobre el mismo notebook.

En *Python* se ha implementado la limpieza general de datos, el desarrollo de métricas, transformaciones exponenciales de los datos, gráficas de correlación y todo lo relacionado con *Natural Language Processing*.

En *R* se ha implementado modelos de regresión, análisis de componente principal y test estadísticos.

Con el fin de aumentar la reproducibilidad del trabajo, en el repositorio se incluyen objetos pickle y archivos csv con los datasets intermedios.

1. Descripción del dataset. ¿Por qué es importante y qué pregunta/problema pretende responder?

1.1. Descripción del dataset.

El conjunto de datos objeto de análisis se ha obtenido a partir de este enlace en Kaggle y esta constituido por 10 datasets (uno por cada mercado regional: Canada, Alemania, Francia, Gran Bretaña, India, Japon, Korea, Méjico, Rusia, Estados Unidos) de 16 características (columnas) que representan los videos que han llegado a ser Trending en sus respectivas regiones.

Las características representadas son las siguientes:

- **video_id**: ID unico para cada vídeo.
- **trending_date**: fecha en la que el vídeo entró en Trend.
- **title**: Título del vídeo.
- **channel_title**: Título del canal al que se ha subido el vídeo.
- **category_id**: ID de la categoría del vídeo. (El nombre de cada categoría viene dado en un fichero json externo)
- **publish_time**: Día y hora a la que se ha subido el vídeo.
- **tags**: SEO tags introducidas en el vídeo.
- **view**: Número de visualizaciones que ha tenido el vídeo.
- **likes**: Número de likes.
- **dislikes**: Número de dislikes.
- **comment_count**: Número de comentarios.
- **thumbnail_link**: Link dónde se almacena la imagen de la thumbnail.
- **comments_disabled**: Booleano que define si se han desactivado los comentarios.
- **ratings_disabled**: Booleano que define si se han desactivado los ratings.

- **video_error_or_removed**: Booleano que define si se han el video se ha borrado por algún error o manualmente.
- **description**: Descripción del video.

1.2. ¿Por qué es importante y qué pregunta/problema pretende responder?

El dataset se ha elegido como continuación o extensión de los datos recopilados en la práctica 1. A modo de recordatorio, en la práctica anterior nuestro objetivo era reconocer los temas que resultaban de mayor interés a la mayoría de la población en un momento en particular. Para cumplir este objetivo nos concentramos en la extracción de datos de las plataformas: *Meneame*, *Reddit* y *Twitter*. Sin embargo otra plataforma muy popular es *Youtube* de la que podemos extraer similar información, además los atributos que obtenemos de la plataforma *Youtube* son comparables a los que obtenemos de las tres plataformas que estudiamos en la primera práctica y un procesamiento de datos muy similar se puede establecer para los cuatro plataformas.

A partir del conjunto de datos para *Youtube* se plantean diferentes preguntas y análisis que pueden que pueden extrapolarse fácilmente a los datos presentados en la primera práctica. En nuestro caso hemos intentado responder a varias preguntas:

- ¿qué categorías son mas populares en cada mercado?
- ¿cuánto tarda un video en hacerse Trending?
- ¿Cuales son las palabras más utilizadas?
- ¿Cuales son las tags más relevantes?
- ¿Cuales son las temas más importantes?
- ¿Existen diferencias en el vocabulario que se utilizan los países que hablan la misma lengua?
- ¿Cual es el sentimiento y la polaridad expresada en los títulos?
- ¿Podemos modelar en número de veces que un vídeo se ve y establecer un modelo predictivo?

Estos análisis son de gran relevancia actualmente ya que como todos sabemos, la industria de creación de contenido online genera una gran cantidad

de dinero y público. Saber navegar éste mercado puede ser muy interesante para cualquier marca o individuo que quiera generar beneficios o atraer público a través de YouTube.

2. Integración y selección de los datos de interés a analizar

Tal y como se menciona en la sección anterior este conjunto de datos extraídos de *Kaggle* cuenta con 10 datasets correspondientes a datos para la plataforma *YouTube* para diez países diferentes. Los datasets de los diferentes países cuentan con los mismos atributos. En general, los datasets también presentan un fallo en la codificación de los caracteres que hace que las columnas con texto cuyos caracteres no están dentro del alfabeto inglés, no se cargen correctamente. Debido a esto se han seleccionado 4 datasets atendiendo a una serie de criterios:

- Podemos leer entender el idioma del país donde provienen los datos. De este modo, podemos corregir problemas de codificación de caracteres en un tiempo razonable.
- Representan zonas geográficas de varios continentes, en caso de que queramos hacer comparaciones regionales.
- Al menos dos datasets deben provenir de una regiones diferentes con la misma lengua. Esto permite hacer comparaciones en el uso del lenguaje.

Atendiendo a estos criterios hemos seleccionado 4 datasets sobre los que aplicaremos los mismo algoritmos. Todos los datos serán tratados de la misma forma. En caso de que algún dataset requiera un tratamiento especial en algún punto se especificará el motivo.

3. Limpieza de datos

Por otra parte, la extensión del código es demasiado grande para éste informe, por lo tanto mostraremos sólo las partes claves y omitiremos las celdas de análisis y descubrimiento de datos. El código completo se puede encontrar en el siguiente LINK.

3.1. Descarga de datos

Debido a que en nuestro caso, los datos vienen en múltiples datasets, vamos a realizar una conexión con Kaggle a través de su API utilizando nuestro fichero kaggle.json dónde guardaremos nuestras claves personales. Además como los ficheros de texto contienen caracteres en diferentes alfabetos, queremos evitar cualquier error de formato derivado de una incorrecta conversión local.

```
[ ] uploaded = files.upload()

Choose Files No file chosen Upload widget is only available when the cell has been executed in

[ ] # Since the data is a line of characters, it is possible to use:
user_key=uploaded['kaggle.json'].decode("utf-8", errors="ignore")

user = user_key[13:28] #Select the righth position for your name
key= user_key[37:-2] #Select the righth positions for the key

#Other method
user_key_2 = pd.read_json('kaggle.json', lines=True)

[ ] url="https://www.kaggle.com/datasnaek/youtube-new?select=USvideos.csv"

[ ] os.environ['KAGGLE_USERNAME'] = user # username from the json file
os.environ['KAGGLE_KEY'] = key # key from the json file
!kaggle datasets download -d datasnaek/youtube-new # api copied from kaggle
```

3.2. Carga de datos

Una vez nos hemos descargado los datasets y los tenemos en nuestro cloud, vamos a descomprimir el fichero y cargar los distintos datasets en un array de datasets que contendrá los diferentes datasets de cada mercado.

3.3. Selección datos

Primero de todo vamos a eliminar las filas con valores True en los atributos lógicos ya que son minoría y no son estadísticas que nos interesen para nuestro análisis.

Como podemos ver, eliminando estas columnas mantenemos el 96 por ciento de los datos.

Debido a que ya solo nos quedan valores verdaderos para esas 3 columnas (comments_disabled, ratings_disabled y video_error_or_removed), vamos a eliminarlas.

```
[ ] # Unzip the data

zf = zipfile.ZipFile(
    '/content/drive/MyDrive/Data_sets/YOUTUBE/youtube-new.zip',
    'r'
)

[ ] # Germany

videos_DE = zf.open(zipfile.ZipFile.namelist(zf)[3])
data_DE = pd.read_csv(videos_DE, encoding='maccentraleurope')

# Great Britain

videos_GB = zf.open(zipfile.ZipFile.namelist(zf)[7])
data_GB = pd.read_csv(videos_GB, encoding=' iso-8859-1')

# Mexico

videos_MX = zf.open(zipfile.ZipFile.namelist(zf)[15])
data_MX = pd.read_csv(videos_MX, encoding='latin1')

# United States of America

videos_US = zf.open(zipfile.ZipFile.namelist(zf)[19])
data_US = pd.read_csv(videos_US, encoding=' iso-8859-1')

[ ] data=[data_DE,data_GB,data_MX,data_US]
```

3.4. Eliminamos categoría sin información (category = 29)

No existe información sobre que categoría representa el id 29, por lo tanto eliminamos ésta del dataset. Otra razón para eliminarla es que al no saber que representa dicha categoría, aun en el caso que pudiésemos extraer alguna información para la categoría 29, no podríamos darle un sentido concreto

Aún eliminandola, preservamos el 95 porciento de la información.

```
[ ] data_treat=[]

for dt in data:
    data_treat.append(dt[(dt['comments_disabled']== False) &
                        (dt['ratings_disabled']== False) &
                        (dt['video_error_or_removed']== False)])

[ ] for i in range(4):
    print(round((len(data_treat[i])/len(data[i]))*100,2))

96.33
97.94
97.62
98.24
```

```
[ ] #Remove columns with constant values

for df in data_treat:

    drop_cols=['comments_disabled','ratings_disabled','video_error_or_removed']

    df.drop(drop_cols, axis=1, inplace=True)
    df.reset_index(drop=True, inplace=True)#Reset indexes
```

```
[ ] for i in range(4):

    data_treat[i]=data_treat[i][(data_treat[i]['category_id']!= 29)]
    data_treat[i].reset_index(drop=True, inplace=True)#Reset indexes

[ ] for i in range(4):

    print(round((len(data_treat[i])/len(data[i]))*100,2))

95.77
97.71
97.25
98.11
```

3.5. Corrección trending date

Los formatos en los que las fechas nos vienen dadas, no son consistentes entre ellos. Por lo tanto, vamos a transformar todas las columnas que contengan un valor de fecha a tipo datetime, empezando por la columna trending date.

3.6. Corrección publish time

Vamos a realizar la misma operación con la columna publish time.


```
#Format yy.dd.mm
data_treat_temp=data_treat

for i in range(4):
    data_treat_temp[i]['trending_date'] = pd.to_datetime(
        data_treat_temp[i]['trending_date'],format='%y.%d.%m'
    )
```

```
# Format yy-mm-dd

for i in range(len(data_treat_temp)):

    for j in range(len(data_treat_temp[i]['publish_time'])):

        data_treat_temp[i].loc[j, 'publish_time'] = pd.to_datetime(
            data_treat_temp[i].loc[j, 'publish_time'][0:10], format='%Y-%m-%d'
        )
```

3.7. Corrección textos

La parte de language cleaning es una de las más importantes para nuestro proyecto ya que tener textos limpios y analizables sirve como una muy buena fundación para proyectos de análisis futuros.

En nuestro caso hemos definido diferentes funciones para corregir caracteres mal codificados que hemos aplicado a las columnas: `title`, `channel_title`, `tags` y `description`.

```
#Latin characters|
def replace_latin(text):
    # https://stackoverflow.com/questions/6116978/how-to-replace-multiple-substrings-of-a-string

    rep = {"Ã¢": "z", "Ã¢": "i", "Ã¢\x81": "A", "Ã¢": "a", "Ã¢\x89": "E", "Ã¢": "e", "Ã¢\x91": "N", "Ã¢": "n",
           "Ã¢\x93": "O", "Ã¢": "o", "Ã¢\x9a": "U", "Ã¢": "u", "Ã¢": "u", "Ã¢\x8d": "f", "Ã¢\xad": "f", "Ã¢\x9c": "s",
           "Ã¢\x98\x85": ":", "Ã¢\x80c": "&", "oI\x81": "o", "o\x9f\x94": ":", "Ã¢": "o", "Ã¢": "f"}

    rep = dict((re.escape(k), v) for k, v in rep.items())
    pattern = re.compile("|".join(rep.keys()))

    text = pattern.sub(lambda m: rep[re.escape(m.group(0))], text) #correct latin

    return(text)

def remove_emotes(text):

    text_c= ".join(filter(lambda x:x[0]!='o', text.split()))

    return(text_c)
```

Éstas funciones se aplican a cada uno de los datasets para poder limpiar las columnas correctamente.

```
[ ] def special_delete(text):
    # https://stackoverflow.com/questions/6116978/how-to-replace-multiple-substrings-of-a-string

    rep = {"\\n": " ", "\x9c": "", "Äd": "", "ÄP": "", "üé": ""}
    rep = dict((re.escape(k), v) for k, v in rep.items())
    pattern = re.compile("|".join(rep.keys()))

    text = pattern.sub(lambda m: rep[re.escape(m.group(0))], text)

    return(text)

def replace_german(text):
    # https://stackoverflow.com/questions/6116978/how-to-replace-multiple-substrings-of-a-string

    rep = {"√s": "ä", "√x": "ü", "√ü": "ß", "√ö": "ö", "√ö": "é"}
    rep = dict((re.escape(k), v) for k, v in rep.items())
    pattern = re.compile("|".join(rep.keys()))

    text = pattern.sub(lambda m: rep[re.escape(m.group(0))], text)

    return(text)
```

```
def replace_miss_spelling(text):
    # https://stackoverflow.com/questions/6116978/how-to-replace-multiple-substrings-of-a-string

    rep = {"exatlon": "Exatlón", "Exatlon": "Exatlón", "capitulo": "capítulo", "Capitulo": "capítulo"}

    rep = dict((re.escape(k), v) for k, v in rep.items())
    pattern = re.compile("|".join(rep.keys()))

    text = pattern.sub(lambda m: rep[re.escape(m.group(0))], text)

    return(text)
```

4. Generación de métricas y análisis de datos

Es complicado analizar nuestros datos ya que no tenemos forma de saber si un video puede ser un outlier, por ejemplo tenemos videos con un gran número de visitas pero pocos comentarios (lo que llamaríamos un click bait), o un video con pocas visitas pero un gran número de comentarios.

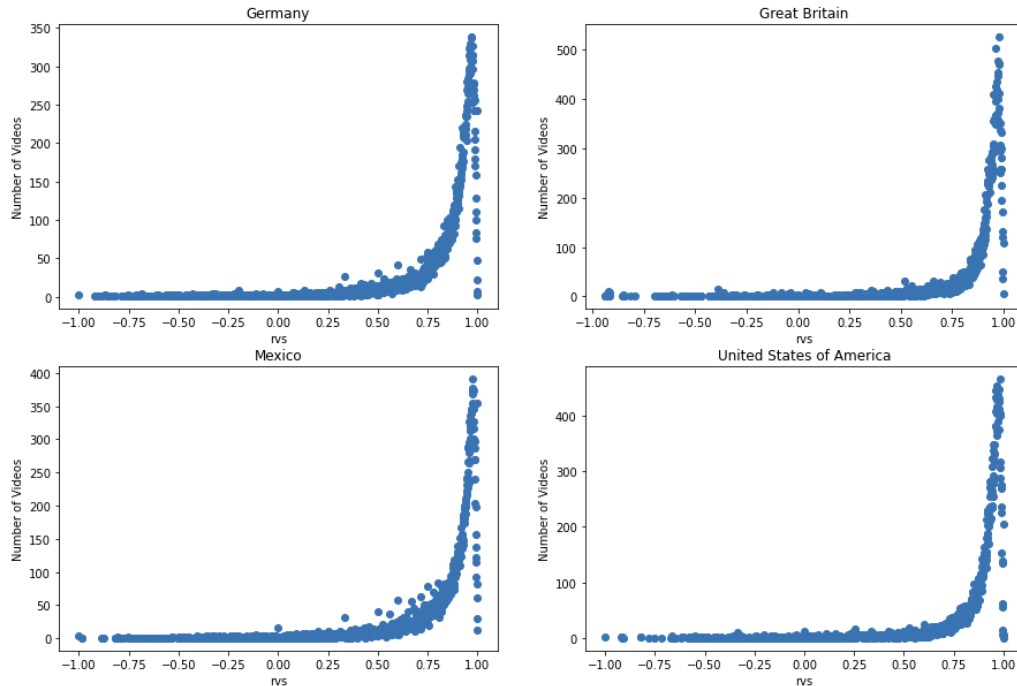
Para poder analizar estos sucesos vamos a crear una serie de métricas que nos ayudarán a realizar análisis más concretos sobre los vídeos.

4.1. Generación métrica video sentiment

La metrika relative video sentiment (o rvs) se calcula utilizando la fórmula $(likes-dislikes)/sum(likes+dislikes)$ y nos ayudará a hacernos una idea de el sentimiento que genera un video sobre 1.

```
[ ] for dt in data_treat:
    dt['rvs'] = round((dt['likes']-dt['dislikes'])/(dt['likes']+dt['dislikes']),3)
```

Haciendo un plot de las frecuencias obtenidas sobre esta métrica, nos damos cuenta de que los videos que llegan a ser Trend, tienden a tener un ratio de likes/dislikes de un 75 por ciento.



4.2. Generación métrica relevancia

Sabemos que para los usuarios comentar un video conlleva más esfuerzo que darle a like o a dislike. También entendemos que un video tiene mayor cantidad de comentarios cuando los viewers tienen algo que decir sobre el tema o algún comentario a provocado controversia.

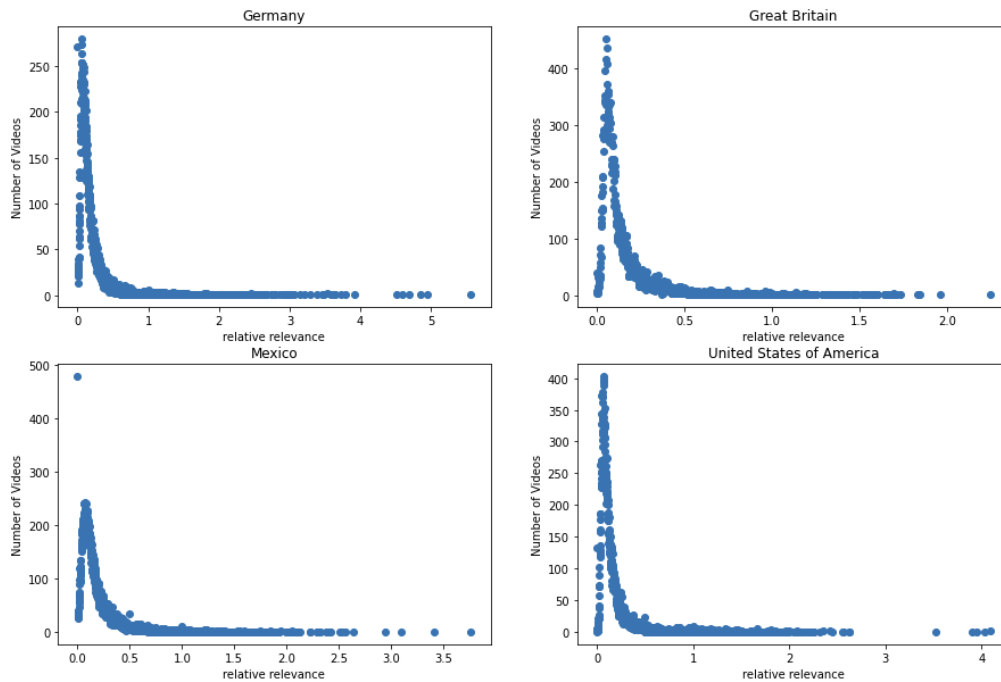
Para medir esto, hemos creado un atributo `rel_elevance`, el cual se calcula con la formula $comment_count / (likes + dislikes)$. Cuando su valor tiende a 1 asumimos que el video es relevante para los viewers y cuando es mayor que 1 quiere decir que es muy relevante.

```
[ ] for dt in data_treat:
    dt['rel_relevance']=round((dt['comment_count'])/(dt['likes']+dt['dislikes']),3)
```

Haciendo un plot de las frecuencias de este valor, podemos observar que la mayor parte de los videos tiene un valor menor que 1. De hecho, observamos

que los videos que alcanzan Trending tienen a tener un ratio menor que 0.5, alcanzando un pico claro en 0.25.

Esto nos da a entender que los videos que alcanzan Trending no suelen ser muy polémicos en general, lo cual tiene sentido ya que suelen ser videos que atraen a un público general.



4.3. Generación métrica sentiment engagement

Por último vamos a generar 3 atributos relacionados con el sentiment engagement de los videos, éstos serán el positive engagement, negative engagement y overall engagement.

El positive engagement se calculará utilizando la fórmula: $likes/views$

```
[ ] for dt in data_treat:
    dt['positive sentiment_engagement']=round((dt['likes'])/(dt['views']),3)
```

El negative engagement se calculará utilizando la fórmula: $dislikes/views$

El overall engagement se calculará utilizando la fórmula: $(likes-dislikes)/views$

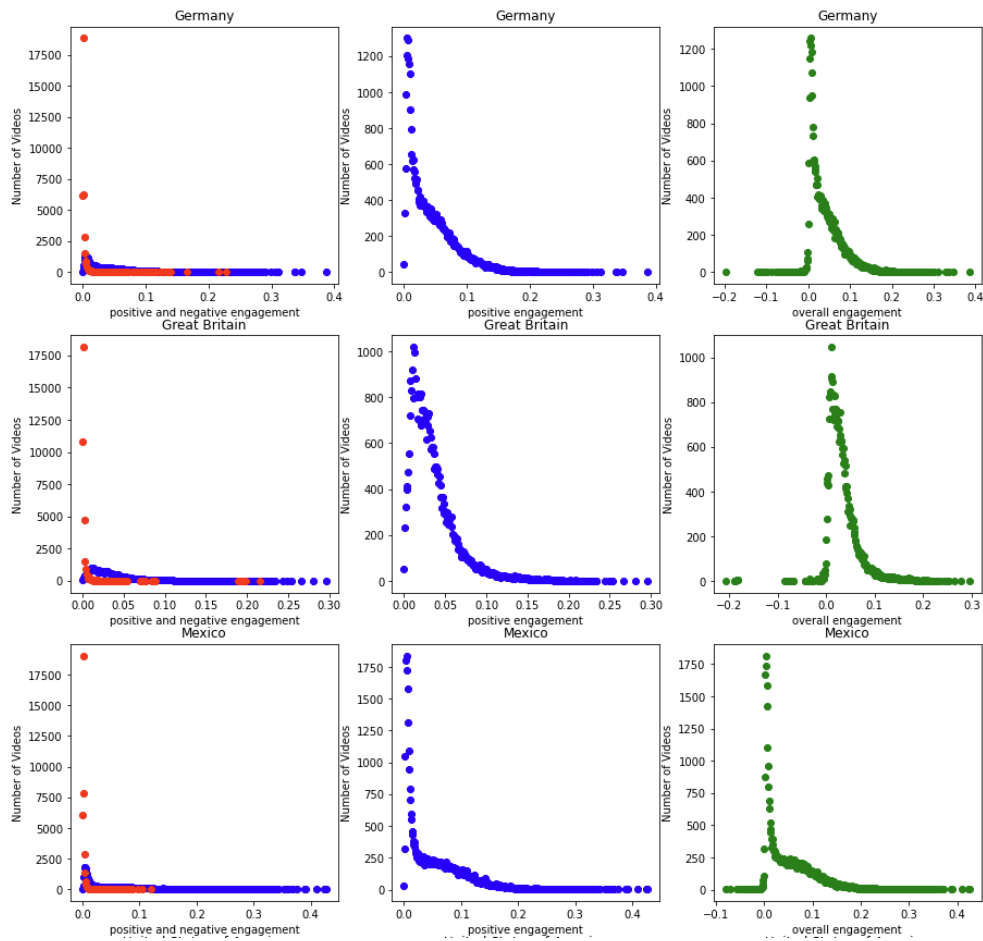
Realizando un plot de las frecuencias, es fácil darse cuenta de que la tendencia en todos los países es que más de el 75 % de los videos estén en un rango de [0-10 %] rvs (likes-dislikes) por visualización del video.

```
[ ] for dt in data_treat:

    dt['negative_sentiment_engagement']=round((dt['dislikes'])/(dt['views']),3)
```

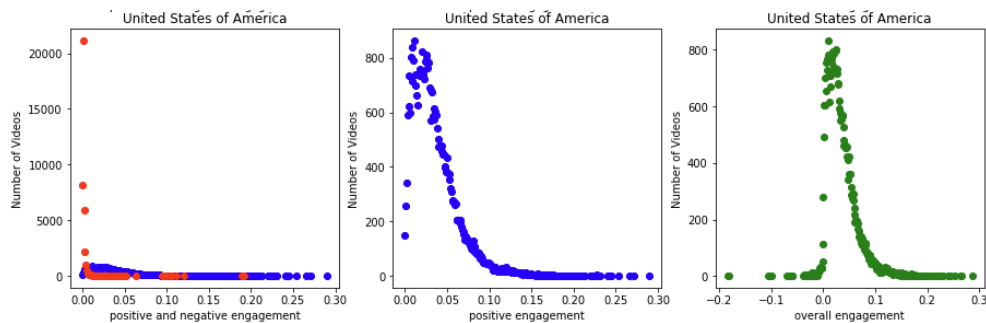
```
[ ] for dt in data_treat:

    dt['overall_sentiment_engagement']=round((dt['likes']-dt['dislikes'])/(dt['views']),3)
```



4.4. Comprobación de normalidad

Nuestro daset presenta dos dificultades a la hora de comprobar si los datos y las métricas siguen una distribución normal. Por una parte el gran número de datos hace inviable la aplicación del test de normalidad de *Shapiro Wilk*, por otra parte, al existir muchos valores repetidos para un atributo el test de *Anderson Darling* no es fiable ya que siempre devolverá como resultado el rechazo de la hipótesis nula. Por este motivo nos centramos en los valores



de *Kurtosis* y *Skewness*. Para estos test, típicamente se considera que una distribución es normal cuando el valor de *Kurtosis* está entre -2 y $+2$ y para *skewness* los valores deben caer dentro del intervalo entre $-0,8$ y $+0,8$.

Podemos comprobar que tanto los valores de los datos en crudo, como las métricas que hemos implementado no siguen una distribución normal. Con el fin de solucionar este problema y conseguir una distribución homogénea de la varianza, hemos optado por utilizar en primera instancia una transformación de *Yeo Johnson*, puesto que nos permite aplicar el mismo código sobre todos los atributos (una transformación de *Yeo Johnson* a diferencia de *Boxcox* nos permite trabajar con valores negativos y ceros).

```
# transform and save data & lambda value

fitted_data = []
fitted_lambda = {}

names=['views','likes','dislikes','comment_count','rvs','rel_relevance','overall
names_fitted=['fviews','flikes','fdislikes','fcomment_count','frvs','frel_rel','
country=['Germany','Great Britain','Mexico','United States of America']

for j,odata in enumerate(original_data):

    lambda_list= []

    for i,col in enumerate(names):

        fdata, lambda_val = stats.yeojohnson(odata.loc[:,col])

        odata[names_fitted[i]]=fdata
        lambda_list.append(lambda_val)

    fitted_lambda[country[j]]=lambda_list
```

	views	likes	dislikes	comment_count	rvs	rel_relevance	overall_sentiment_engagement
Germany	0	0	0	0	0	0	0
Great Britain	0	0	0	0	0	0	0
Mexico	0	0	0	0	0	0	0
United States of America	0	0	0	0	0	0	0

fviews	flikes	fdislikes	fcomment_count	frvs	frel_rel	foverall_sen_eng
1	1	1	1	0	0	0
1	1	1	1	0	0	0
1	1	1	1	0	1	0
1	1	1	1	0	0	0

Hemos creado una función *normal check* que nos devuelve 1 cuando los datos de un atributo siguen una distribución normal (en las condiciones especificadas arriba) y 0 cuando no la sigue. La hemos aplicado sobre todas las columnas de todos los datasets y los datos los hemos visualizado como un dataframe que se copia abajo:

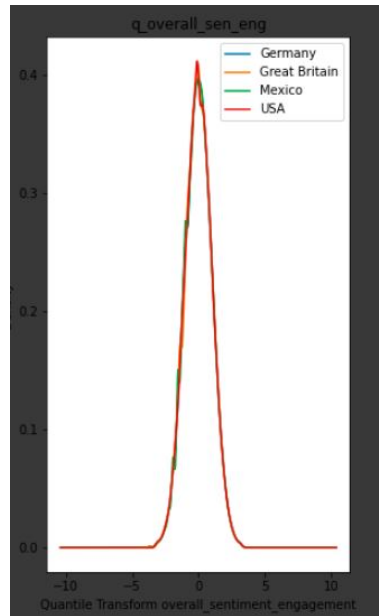
Como se puede ver los datos en crudo se han podido transformar de forma que ahora siguen una distribución normal, pero las métricas no siguen todavía una distribución normal.

Con el fin de corregir este hecho hemos realizado una transformación por cuantiles sobre las métricas y obtenido los siguientes resultados.

	q_rvs	q_rel_rel	q_overall_sen_eng
Germany	0	0	1
Great Britain	0	0	1
Mexico	0	0	0
United States of America	0	0	1

Con el fin de comprobar la discrepancia presente en los datos de *México* hemos hecho una representación de densidades para los datos *q_overall_sen_eng*,

es decir los datos que cuya variable ha sufrido una transformación por quantiles.



En esta curva no podemos apreciar ninguna diferencia significativa entre las curvas para cada país.

En conclusión ahora contamos con los datos transformados en crudo vía *YeoJohnson* que siguen una distribución normal y la métrica *overall_sent_eng* que transformada por cuantiles también sigue una distribución normal por lo que podremos aplicar test de correlación, modelos de regresión y otros test estadísticos para métricos.

5. Pruebas estadísticas

5.1. Cuáles son las 50 palabras más populares por país y atributo

Una de las formas que tenemos de seguir o analizar tendencias en videos de YouTube, es contar las palabras en los diferentes atributos de los videos.

En nuestro caso, vamos a contar las palabras en los títulos de los videos, comentarios, descripciones y canales.

Para empezar vamos a cargar los datos con los textos ya transformados y limpios.


```
[ ] # The final list of Stop Words is included
    cv = CountVectorizer(stop_words=stop_words)

[ ] list_of_dtms=[]
    column=['title','channel_title','tags','description']

    for i,col in enumerate(column):

        data_cv =cv.fit_transform(List_of_txt_frames[i][col])

        data_dtm = pd.DataFrame(data_cv.toarray(), columns=cv.get_feature_names())

        data_dtm.index = List_of_txt_frames[i].index

        list_of_dtms.append(data_dtm)

/usr/local/lib/python3.6/dist-packages/sklearn/feature_extraction/text.py:385: UserWarning:
'stop_words.' % sorted(inconsistent))

[ ] data_dtm_trans=[]

    for data in list_of_dtms:

        data = data.transpose()

        data_dtm_trans.append(data)
```

Una vez tenemos los datos cargados, vamos a encontrar las 50 palabras mas populares para cada atributo del video.

Por último, vamos a realizar un wordcloud de los datos obtenidos.

Empezamos con los wordcloud de las tags.

Como podemos ver, en alemania hay varias palabras en turco, esto se debe a la gran inmigración Turca que existe en el pais. Esto provoca que los datos estén divididos en turco y alemán.

El resto de wordclouds son bastante claros, a destacar en el Mejicano, que podemos ver la TV Azteca y exatlon, que es un programa de gran éxito en el pais.

```
[ ] # Find the top 50 words that appers the most on the titles for each country
top_title_dict = {}
for c in data_dtm_trans[0].columns:
    top = data_dtm_trans[0][c].sort_values(ascending=False).head(50)
    top_title_dict[c]= list(zip(top.index, top.values))

top_title_dict

[ ] # Find the top 50 words that appers the most on the channel titles for each country
top_chn_title_dict = {}
for c in data_dtm_trans[1].columns:
    top = data_dtm_trans[1][c].sort_values(ascending=False).head(50)
    top_chn_title_dict[c]= list(zip(top.index, top.values))

top_chn_title_dict

[ ] # Find the top 50 words that appers the most on the tags for each country
top_tags_dict = {}
for c in data_dtm_trans[2].columns:
    top = data_dtm_trans[2][c].sort_values(ascending=False).head(50)
    top_tags_dict[c]= list(zip(top.index, top.values))

top_tags_dict

▶ # Find the top 50 words that appers the most on the decription for each country
top_desc_dict = {}
for c in data_dtm_trans[3].columns:
    top = data_dtm_trans[3][c].sort_values(ascending=False).head(50)
    top_desc_dict[c]= list(zip(top.index, top.values))

top_desc_dict
```

Tags



A continuación mostraremos el wordcloud de los títulos de los videos.

A destacar que la palabra trump aparece más en Alemania que en EEUU.

Video Title

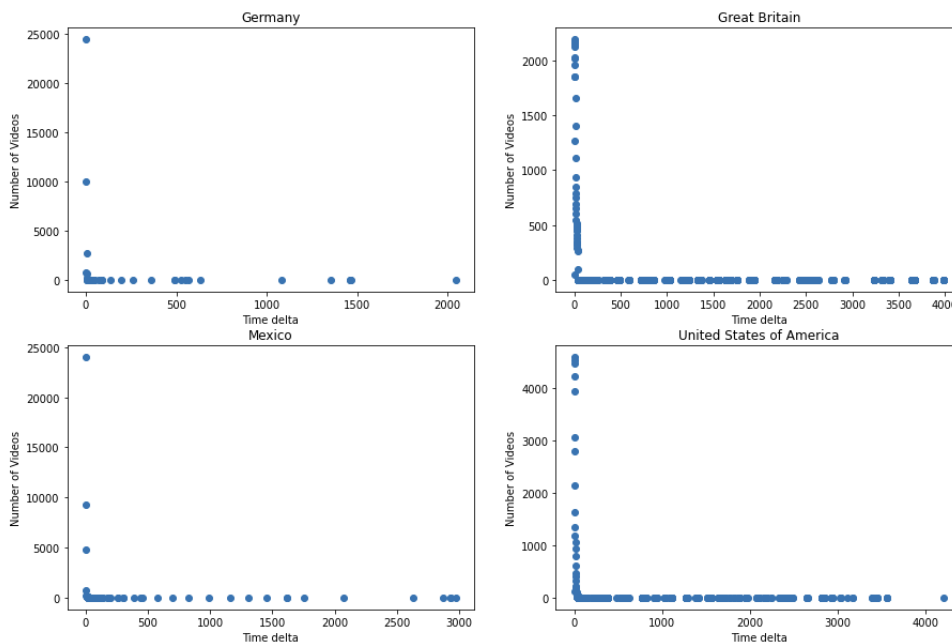


5.2. Cuánto tarda un video en llegar a Trending

Para analizar cuánto tarda un video en llegar a Trending, vamos a generar una nueva columna con este valor, que calcularemos restando la fecha de Trending con la fecha de publicación.

```
[ ] for i,data in enumerate(data_treat):  
    data_treat[i]['time_delta']= pd.to_timedelta(  
        data.loc[:,'trending_date']-data.loc[:,'publish_time']  
    ).dt.days
```

Una vez tenemos estos datos, vamos a realizar un plot de frecuencias para el número de videos por días que tarda en llegar a trending.



Con estas gráficas podemos ver que la mayoría de videos llegan a trending en los primeros días tras su lanzamiento. Para hacernos una mejor idea vamos a realizar un plot de que porcentajes de videos llegan a trending en diferentes rangos de tiempo.

```

▶ #checking for the percent of videos that become trending in at least 30 days

times=[0,1,2,5,7,10,15,30]

times_list=[]

for data in data_treat:

    time_list_temp=[]

    for i, time in enumerate(times):
        try:
            x=(data['time_delta'][(data['time_delta']>=time) & (
                data['time_delta'] < (times[i+1])) ].count()/(len(data['time_delta']))
                )

            x=round(x*100,2)

            time_list_temp.append(x)
        except:

            pass

    times_list.append(time_list_temp)

[ ] title_label=['Germany','Great Britain','Mexico','United States of America']
    times=[0,1,2,5,7,10,15,30]

    plt.figure(figsize=(15,10))

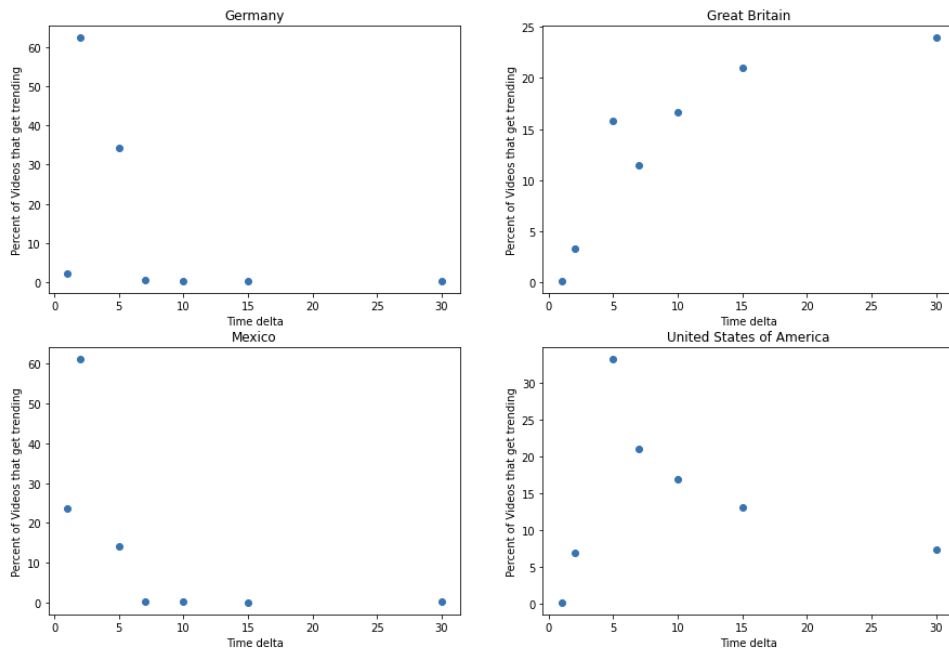
    index_=1

    for tm in times_list:

        plt.subplot(2,2,index_)
        plt.scatter(times[1:],tm)
        plt.title(title_label[index_-1], loc='center')
        plt.xlabel("Time delta")
        plt.ylabel("Percent of Videos that get trending")
        index_+=1

    plt.show()

```



Los datos mostrados representan un 98 % de los videos. Estos resultados son muy interesantes ya que nos dan una idea de cuando tarda algo en hacerse popular en un los diferentes paises, cada cuanto utilizan la plataforma los usuarios en los diferentes paises, o cuanto tarda un usuario en ver algo fuera de sus intereses habituales.

5.3. Qué categorías tienen más éxito en cada país

Para realizar visualizaciones correctas sobre las categorías de los videos, vamos crear una columna con el nombre de la categoría del video (ahora mismo solo teniamos sus ids).

```

import json

# Read JSON ID
categories_DE = zf.open(zipfile.ZipFile.namelist(zf)[2])
json_dict = json.load(categories_DE)

# Create dict id => name
categories_dict = {}
items = json_dict["items"]
for item in items:
    categories_dict[int(item["id"])] = item["snippet"]["title"]

cat_keys = categories_dict.values()

# Map and create categories name column
for df in data_treat:
    df["category_name"] = df["category_id"].map(categories_dict)

```

Una vez tenemos la columna creada, vamos a obtener el numero total de videos en trending por categoria en cada mercado.

```

[ ] # We get the list of categories
title_label=['Germany','Great Britain','Mexico','United States of America']
counter = 0
index_brais = 1
num_dict = {}
fig = plt.figure(figsize=(20,30))

# Iterate through each dataset
for df in data_treat:
    categories_list = df.category_name.unique()
    categories_df_list = []
    rvs_dict = {}
    num_dict = {}

    # For each category, we get the number of videos
    for category in categories_list:
        category_df = df.loc[df['category_name'] == category]
        rvs_dict[str(category)] = category_df.rvs.mean()
        num_dict[str(category)] = category_df.shape[0]

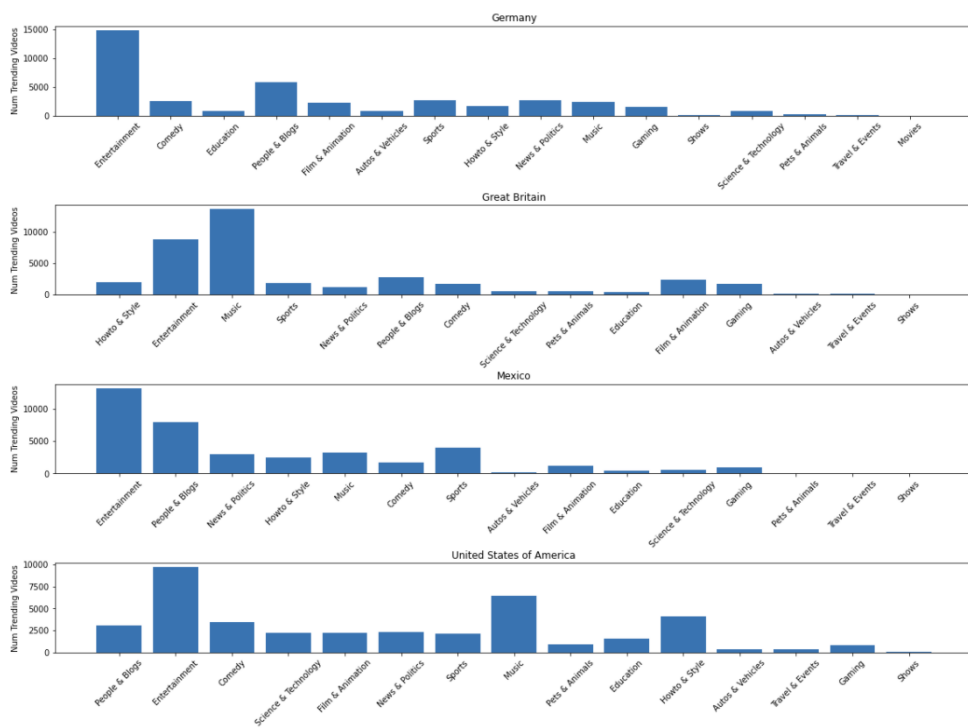
    # Show the count of videos per category
    plt.subplot(8,1,index_brais)
    plt.subplots_adjust(hspace=1)
    keys = num_dict.keys()
    values = num_dict.values()
    plt.title(title_label[counter])
    plt.ylabel("Num Trending Videos")
    plt.xticks(rotation=45)
    plt.bar(keys, values)
    index_brais +=1
    counter +=1

fig.show()

```

Con este plot vamos a ser capaces de ver la distribución de las vistas por cada categoría. Como datos más relevantes que se pueden observar, los videos de categoría entretenimiento son los que suelen alcanzar trending con más asiduidad.

Otro dato que me ha parecido muy interesante es como los videos musicales son muy populares en EEUU y UK, mientras que en Mexico o Alemania no son casi relevantes. Ésta información podría ser de gran utilidad para artistas que intentan crecer en diferentes regiones o mercados.



5.4. Modelo de Regresión

Ésta sección se ha realizado en R a diferencia de el resto de la práctica que está desarrollada en Python.

6. Conclusiones

Hemos obtenido múltiples conclusiones muy interesantes, hemos visto cómo los diferentes países interactúan con los videos, ya sea desde el punto de

vista de como reaccionan a ellos o cuáles son los más populares.

Todas estas conclusiones serán de gran utilidad para cualquier creador de contenido que trate de hacerse un hueco en YouTube, o incluso grandes empresas que quieran realizar propuestas de marketing a través de la plataforma podrían utilizar ésta información para especializar los anuncios por comunidad o mercado.

Referencias

- [1] Leonard Berzkowitz, *Advances in Experimental Social Psychology*, 1967, Academic press
- [2] Richard McElreath, Robert Boyd, *Mathematical Models of Social Evolution: A Guide for the Perplexed*, 2007, The University of Chicago Press
- [3] Thomas L. Saaty and Joyce M. Alexander, *Thinking with models: Mathematical Models in the Physical, Biological, and Social Sciences*, 2015, RWS Publications
- [4] Jason Radford and Kenneth Joseph, *Theory In, Theory Out: The Uses of Social Theory in Machine Learning for Social Science*, 2020, doi.org/10.3389/fdata.2020.00018