Sakura Dat

DOCUMENTACIÓN TÉCNICA

Prueba - Fundamentos Data Science

Martes 19 de mayo de 2020, Santiago de Chile

Sakura SPA

Miembros de la Célula:

Susana Arce

Fabiola Aravena

Rodrigo Pereira

Administrador de Contrato: Gonzalo Seguel

Sponsor: Andrea Villaroel

Objetivo:

Nuestro sponsor posee un cátalogo de contenido audiovisual de 12.294 títulos de Anime, requiere poder efecturar recomendaciones de qué ver a los usuarios en base a otros Anime han sido de su gusto.

Propuesta:

Aplicación web en donde el usuario seleccione de una lista de Anime propuestos los que han sido de su gusto, y en base a ello se le entregue una lista de otras alternativas que sean a fin con sus preferencias.

Los archivos pueden ser descargados desde: ***https://drive.google.com/open?id=1TlhsxxaTENAO6xIw-EvvZhRAuJaJtPFB (https://drive.google.com/open?id=1TlhsxxaTENAO6xIw-EvvZhRAuJaJtPFB)***

Procesamiento de los datos

Se procesan los data sets recibidos para su posterior modelamiento y predicción

In [229]:
```
# Importación de librerías para procesamiento de datos.
import pandas as pd
import numpy as np

import matplotlib.pyplot as plt
import seaborn as sns
plt.style.use('seaborn')

import factor_analyzer as factor
import missingno as msngo

import warnings
warnings.filterwarnings(action="ignore")
```

Recogida de datos

In [230]:
```
#Importamos los data sets para su procesamiento
df_interacciones = pd.read_csv("rating.csv")
df_interacciones.head()
```

Out[230]:

	user_id	anime_id	rating
0	1	20	-1
1	1	24	-1
2	1	79	-1
3	1	226	-1
4	1	241	-1

In [231]:
```
df_titulos = pd.read_csv("anime.csv", sep=";")
df_titulos.head()
```

Out[231]:

	anime_id	name	genre	type	episodes	rating	members
0	32281	Kimi no Na wa.	Drama, Romance, School, Supernatural	Movie	1	9,37	200630
1	5114	Fullmetal Alchemist: Brotherhood	Action, Adventure, Drama, Fantasy, Magic, Mili...	TV	64	9,26	793665
2	28977	Gintama°	Action, Comedy, Historical, Parody, Samurai, S...	TV	51	9,25	114262
3	9253	Steins;Gate	Sci-Fi, Thriller	TV	24	9,17	673572
4	9969	Gintama'	Action, Comedy, Historical, Parody, Samurai, S...	TV	51	9,16	151266

Procesamiento de datos

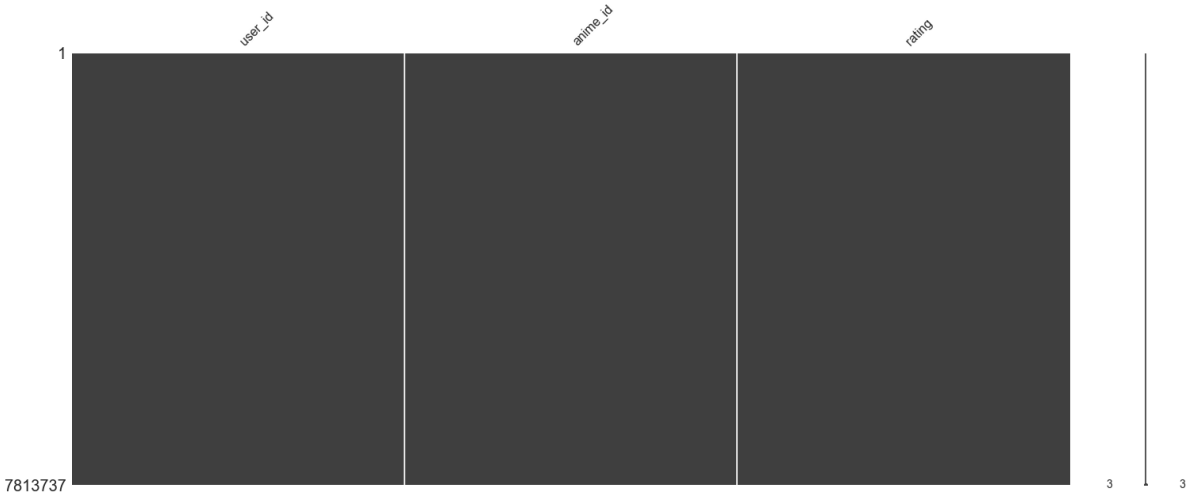
Data Set de Interacciones

In [232]:
```
len(df_interacciones) #Exploramos la cantidad de registros del data set
```

Out[232]: 7813737

In [233]:
```
msngo.matrix(df_interacciones) #exploramos la completitud de los da
tos
```

Out[233]: <matplotlib.axes._subplots.AxesSubplot at 0x1a175f62b0>



In [234]:
```
for i in df_interacciones.columns:
    a= df_interacciones[i].unique()
    print("--------------------------------")
    print("Columna: "+i)
    print(len(a)) #Exploramos la cantidad de categorías de cada atr
ibuto
    print(df_interacciones[i].isnull().value_counts()) #Exploramos
la cantidad de valores nulo en los registros
    print(a)
```

```
--------------------------------
Columna: user_id
73515
False    7813737
Name: user_id, dtype: int64
[    1     2     3 ... 73514 73515 73516]
--------------------------------
Columna: anime_id
11200
False    7813737
Name: anime_id, dtype: int64
[   20    24    79 ... 29481 34412 30738]
--------------------------------
Columna: rating
11
False    7813737
Name: rating, dtype: int64
[-1 10  8  6  9  7  3  5  4  1  2]
```

Se observa que no existen registros nulos

Existen 73.515 user_id y 11.200 títulos de anime diferentes

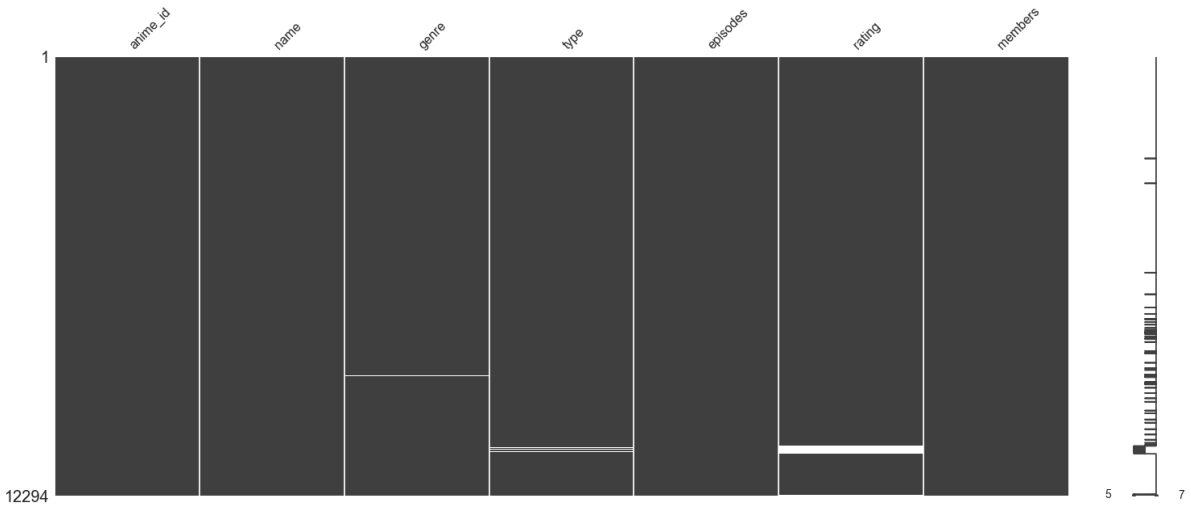
Data Set de Títulos

In [235]:
```
len(df_titulos) #Exploramos la cantidad de registros del data set
```

Out[235]: 12294

In [236]:
```
msngo.matrix(df_titulos) #exploramos la completitud de los datos
```

Out[236]: <matplotlib.axes._subplots.AxesSubplot at 0x1a3f242320>



In [237]:
```
for i in df_titulos.columns:
    a= df_titulos[i].unique()
    print("-------------------------------")
    print("Columna: "+i)
    print(len(a)) #Exploramos la cantidad de categorías de cada atr
ibuto
    print(df_titulos[i].isnull().value_counts()) #Exploramos la can
tidad de valores nulo en los registros
    print(a)
```

```
-------------------------------
Columna: anime_id
12294
False    12294
Name: anime_id, dtype: int64
[32281  5114 28977 ...  5621  6133 26081]
-------------------------------
Columna: name
12292
False    12294
Name: name, dtype: int64
```

```
['Kimi no Na wa.' 'Fullmetal Alchemist: Brotherhood' 'Gintama°' ..
.
 'Violence Gekiga David no Hoshi'
 'Violence Gekiga Shin David no Hoshi: Inma Densetsu'
 'Yasuji no Pornorama: Yacchimae!!']
--------------------------------
Columna: genre
3272
False    12247
True        47
Name: genre, dtype: int64
['Drama, Romance, School, Supernatural'
 'Action, Adventure, Drama, Fantasy, Magic, Military, Shounen'
 'Action, Comedy, Historical, Parody, Samurai, Sci-Fi, Shounen' ..
.
 'Hentai, Sports' 'Drama, Romance, School, Yuri' 'Hentai, Slice of
Life']
--------------------------------
Columna: type
7
False    12269
True        25
Name: type, dtype: int64
['Movie' 'TV' 'OVA' 'Special' 'Music' 'ONA' nan]
--------------------------------
Columna: episodes
187
False    12294
Name: episodes, dtype: int64
['1' '64' '51' '24' '10' '148' '110' '13' '201' '25' '22' '75' '4'
'26'
 '12' '27' '43' '74' '37' '2' '11' '99' 'Unknown' '39' '101' '47'
'50'
 '62' '33' '112' '23' '3' '94' '6' '8' '14' '7' '40' '15' '203' '7
7' '291'
 '120' '102' '96' '38' '79' '175' '103' '70' '153' '45' '5' '21' '
63' '52'
 '28' '145' '36' '69' '60' '178' '114' '35' '61' '34' '109' '20' '
9' '49'
 '366' '97' '48' '78' '358' '155' '104' '113' '54' '167' '161' '42
' '142'
 '31' '373' '220' '46' '195' '17' '1787' '73' '147' '127' '16' '19
' '98'
 '150' '76' '53' '124' '29' '115' '224' '44' '58' '93' '154' '92'
'67'
 '172' '86' '30' '276' '59' '72' '330' '41' '105' '128' '137' '56'
'55'
 '65' '243' '193' '18' '191' '180' '91' '192' '66' '182' '32' '164
' '100'
 '296' '694' '95' '68' '117' '151' '130' '87' '170' '119' '84' '10
8' '156'
 '140' '331' '305' '300' '510' '200' '88' '1471' '526' '143' '726'
'136'
 '1818' '237' '1428' '365' '163' '283' '71' '260' '199' '225' '312
```

```
' '240'
 '1306' '1565' '773' '1274' '90' '475' '263' '83' '85' '1006' '80'
'162'
 '132' '141' '125']
------------------------------
Columna: rating
599
False    12064
True       230
Name: rating, dtype: int64
[' 9,37' ' 9,26' ' 9,25' ' 9,17' ' 9,16' ' 9,15' ' 9,13' ' 9,11' '
9,10'
 ' 9,06' ' 9,05' ' 9,04' ' 8,98' ' 8,93' ' 8,92' ' 8,88' ' 8,84' '
8,83'
 ' 8,82' ' 8,81' ' 8,80' ' 8,78' ' 8,77' ' 8,76' ' 8,75' ' 8,74' '
8,73'
 ' 8,72' ' 8,71' ' 8,69' ' 8,68' ' 8,67' ' 8,66' ' 8,65' ' 8,64' '
8,62'
 ' 8,61' ' 8,60' ' 8,59' ' 8,58' ' 8,57' ' 8,56' ' 8,55' ' 8,54' '
8,53'
 ' 8,52' ' 8,51' ' 8,50' ' 8,49' ' 8,48' ' 8,47' ' 8,46' ' 8,45' '
8,44'
 ' 8,43' ' 8,42' ' 8,41' ' 8,40' ' 8,39' ' 8,38' ' 8,37' ' 8,36' '
8,35'
 ' 8,34' ' 8,33' ' 8,32' ' 8,31' ' 8,30' ' 8,29' ' 8,28' ' 8,27' '
8,26'
 ' 8,25' ' 8,24' ' 8,23' ' 8,22' ' 8,21' ' 8,20' ' 8,19' ' 8,18' '
8,17'
 ' 8,16' ' 8,15' ' 8,14' ' 8,13' ' 8,12' ' 8,11' ' 8,10' ' 8,09' '
8,08'
 ' 8,07' ' 8,06' ' 8,05' ' 8,04' ' 8,03' ' 8,02' ' 8,01' ' 8,00' '
7,99'
 ' 7,98' ' 7,97' ' 7,96' ' 7,95' ' 7,94' ' 7,93' ' 7,92' ' 7,91' '
7,90'
 ' 7,89' ' 7,88' ' 7,87' ' 7,86' ' 7,85' ' 7,84' ' 7,83' ' 7,82' '
7,81'
 ' 7,80' ' 7,79' ' 7,78' ' 7,77' ' 7,76' ' 7,75' ' 7,74' ' 7,73' '
7,72'
 ' 7,71' ' 7,70' ' 7,69' ' 7,68' ' 7,67' ' 7,66' ' 7,65' ' 7,64' '
7,63'
 ' 7,62' ' 7,61' ' 7,60' ' 7,59' ' 7,58' ' 7,57' ' 7,56' ' 7,55' '
7,54'
 ' 7,53' ' 7,52' ' 7,51' ' 7,50' ' 7,49' ' 7,48' ' 7,47' ' 7,46' '
7,45'
 ' 7,44' ' 7,43' ' 7,42' ' 7,41' ' 7,40' ' 7,39' ' 7,38' ' 7,37' '
7,36'
 ' 7,35' ' 7,34' ' 7,33' ' 7,32' ' 7,31' ' 7,30' ' 7,29' ' 7,28' '
7,27'
 ' 7,25' ' 7,26' ' 7,24' ' 7,23' ' 7,22' ' 7,21' ' 7,20' ' 7,19' '
7,18'
 ' 7,17' ' 7,16' ' 7,14' ' 7,15' ' 7,13' ' 7,12' ' 7,11' ' 7,10' '
7,09'
 ' 7,08' ' 7,07' ' 7,06' ' 7,05' ' 7,04' ' 7,03' ' 7,02' ' 7,01' '
7,00'
```

```
 ' 6,99' ' 6,98' ' 6,97' ' 6,96' ' 6,95' ' 6,94' ' 6,93' ' 6,92' '
6,91'
 ' 6,90' ' 6,89' ' 6,88' ' 6,87' ' 6,86' ' 6,85' ' 6,84' ' 6,83' '
6,82'
 ' 6,81' ' 6,80' ' 6,79' ' 6,78' ' 6,75' ' 6,77' ' 6,76' ' 6,74' '
6,73'
 ' 6,72' ' 6,71' ' 6,70' ' 6,69' ' 6,68' ' 6,67' ' 6,66' ' 6,65' '
6,64'
 ' 6,63' ' 6,62' ' 6,61' ' 6,60' ' 6,59' ' 6,58' ' 6,57' ' 6,56' '
6,55'
 ' 6,54' ' 6,53' ' 6,52' ' 6,51' ' 6,47' ' 6,50' ' 6,49' ' 6,48' '
6,46'
 ' 6,45' ' 6,42' ' 6,44' ' 6,43' ' 6,39' ' 6,41' ' 6,40' ' 6,38' '
6,37'
 ' 6,35' ' 6,36' ' 6,34' ' 6,33' ' 6,32' ' 6,31' ' 6,30' ' 6,29' '
6,28'
 ' 6,27' ' 6,26' ' 6,25' ' 6,22' ' 6,24' ' 6,23' ' 6,21' ' 6,20' '
6,19'
 ' 6,18' ' 6,17' ' 6,16' ' 6,15' ' 6,14' ' 6,13' ' 6,12' ' 6,10' '
6,11'
 ' 6,09' ' 6,08' ' 6,06' ' 6,07' ' 6,05' ' 6,04' ' 6,03' ' 6,01' '
6,02'
 ' 6,00' ' 5,99' ' 5,98' ' 5,97' ' 5,96' ' 5,95' ' 5,94' ' 5,93' '
5,92'
 ' 5,91' ' 5,89' ' 5,90' ' 5,88' ' 5,87' ' 5,86' ' 5,85' ' 5,84' '
5,83'
 ' 5,82' ' 5,81' ' 5,80' ' 5,79' ' 5,78' ' 5,77' ' 5,76' ' 5,75' '
5,74'
 ' 5,73' ' 5,72' ' 5,70' ' 5,71' ' 5,69' ' 5,68' ' 5,67' ' 5,66' '
5,65'
 ' 5,64' ' 5,63' ' 5,62' ' 5,61' ' 5,60' ' 5,59' ' 5,58' ' 5,57' '
5,56'
 ' 5,55' ' 5,53' ' 5,54' ' 5,52' ' 5,51' ' 5,50' ' 5,49' ' 5,48' '
5,46'
 ' 5,47' ' 5,45' ' 5,44' ' 5,43' ' 5,42' ' 5,41' ' 5,40' ' 5,39' '
5,38'
 ' 5,37' ' 5,36' ' 5,35' ' 5,34' ' 5,33' ' 5,32' ' 5,31' ' 5,30' '
5,29'
 ' 5,28' ' 5,27' ' 5,26' ' 5,24' ' 5,25' ' 5,23' ' 5,22' ' 5,21' '
5,20'
 ' 5,19' ' 5,14' ' 5,18' ' 5,17' ' 5,16' ' 5,15' ' 5,13' ' 5,11' '
5,12'
 ' 5,10' ' 5,09' ' 5,07' ' 5,08' ' 5,06' ' 5,05' ' 5,04' ' 5,03' '
5,02'
 ' 5,01' ' 5,00' ' 4,99' ' 4,98' ' 4,97' ' 4,96' ' 4,95' ' 4,94' '
4,93'
 ' 4,92' ' 4,91' ' 4,90' ' 4,89' ' 4,88' ' 4,84' ' 4,87' ' 4,86' '
4,85'
 ' 4,83' ' 4,82' ' 4,81' ' 4,80' ' 4,79' ' 4,78' ' 4,77' ' 4,76' '
4,75'
 ' 4,74' ' 4,73' ' 4,72' ' 4,71' ' 4,70' ' 4,69' ' 4,68' ' 4,66' '
4,67'
 ' 4,65' ' 4,64' ' 4,63' ' 4,62' ' 4,60' ' 4,59' ' 4,58' ' 4,57' '
4,56'
```

```
 ' 4,55' ' 4,54' ' 4,53' ' 4,52' ' 4,49' ' 4,50' ' 4,48' ' 4,46' '
4,45'
 ' 4,44' ' 4,43' ' 4,42' ' 4,40' ' 4,39' ' 4,38' ' 4,36' ' 4,35' '
4,34'
 ' 4,32' ' 4,31' ' 4,30' ' 4,28' ' 4,27' ' 4,26' ' 4,25' ' 4,24' '
4,23'
 ' 4,22' ' 4,21' ' 4,19' ' 4,17' ' 4,16' ' 4,15' ' 4,11' ' 4,08' '
4,04'
 ' 4,03' ' 4,02' ' 4,00' ' 3,99' ' 3,98' ' 3,96' ' 3,91' ' 3,90' '
3,88'
 ' 3,87' ' 3,86' ' 3,84' ' 3,83' ' 3,82' ' 3,80' ' 3,78' ' 3,76' '
3,75'
 ' 3,74' ' 3,73' ' 3,71' ' 3,70' ' 3,68' ' 3,65' ' 3,63' ' 3,62' '
3,60'
 ' 3,58' ' 3,59' ' 3,56' ' 3,47' ' 3,46' ' 3,41' ' 3,36' ' 3,33' '
3,32'
 ' 3,27' ' 2,95' ' 2,93' ' 2,78' ' 2,67' ' 2,37' ' 2,14' ' 2,00' '
4,06'
 ' 4,18' ' 4,09' ' 3,67' ' 3,00' ' 4,33' ' 3,89' ' 4,20' ' 3,61' '
4,13'
 ' 3,11' ' 2,58' ' 4,29' ' 3,43' ' 3,57' ' 4,05' ' 4,51' ' 3,40' '
3,79'
 ' 3,81' ' 3,92' ' 3,48' ' 3,38' ' 3,95' ' 4,61' ' 3,34' ' 3,02' '
2,69'
 ' 3,50' ' 3,97' ' 3,85' nan ' 3,77' ' 3,53' ' 9,33' ' 4,14' ' 4,1
2'
 ' 3,20' ' 3,93' ' 3,52' ' 2,55' ' 9,00' ' 2,97' ' 4,10' ' 3,39' '
3,17'
 ' 9,50' ' 3,25' ' 2,80' ' 2,91' ' 2,75' ' 4,41' ' 3,72' ' 3,94' '
1,67'
 ' 4,37' ' 3,69' ' 3,26' ' 3,49' ' 2,49' ' 2,84' ' 4,07' ' 3,35' '
3,54'
 ' 3,21' ' 3,42' ' 9,60' ' 3,28' '10,00' ' 3,51' ' 3,29' ' 2,72' '
3,64'
 ' 2,98' ' 3,44' ' 1,92' ' 2,86' ' 3,14']
------------------------------
Columna: members
6706
False    12294
Name: members, dtype: int64
[200630 793665 114262 ...  27411  57355    652]
```

Se observa que existen registros nulos

Existen 12.294 títulos de anime diferentes

In [238]:
```
# Limpiamos y pasamos a float el atributo ranking
df_titulos["rating"] = df_titulos["rating"].str.replace(",",".")
df_titulos["rating"] = df_titulos["rating"].str.replace(" ","")
df_titulos["rating"] = df_titulos["rating"].astype(float)
df_titulos["rating"].isnull().value_counts()
```

Out[238]:
```
False    12064
True       230
Name: rating, dtype: int64
```

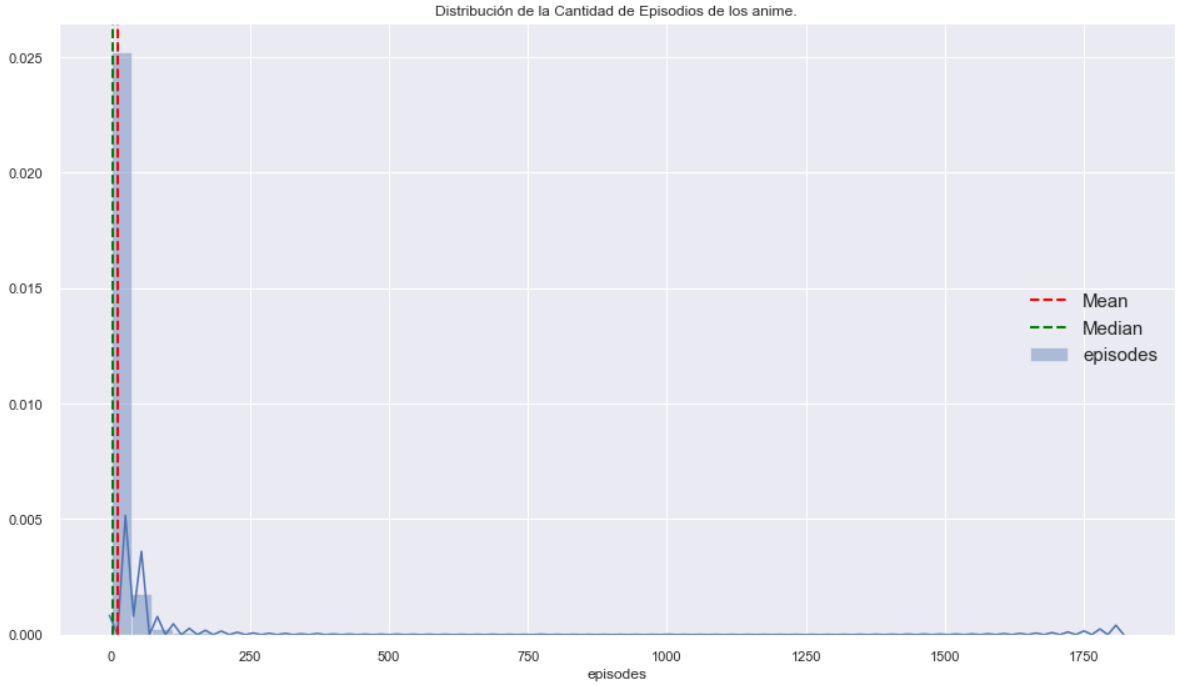
In [239]:
```
# Limpiamos y pasamos a float el atributo episodes
df_titulos["episodes"].value_counts()
```

Out[239]:
```
1          5677
2          1076
12          816
13          572
26          514
3           505
Unknown     340
4           327
6           268
24          181
52          177
25          165
5           121
10          114
51          103
39           86
50           83
11           72
7            72
8            60
22           42
9            40
20           36
48           35
23           33
14           32
49           31
16           30
47           25
38           23
            ...
87            1
1306          1
224           1
366           1
71            1
67            1
175           1
83            1
125           1
```

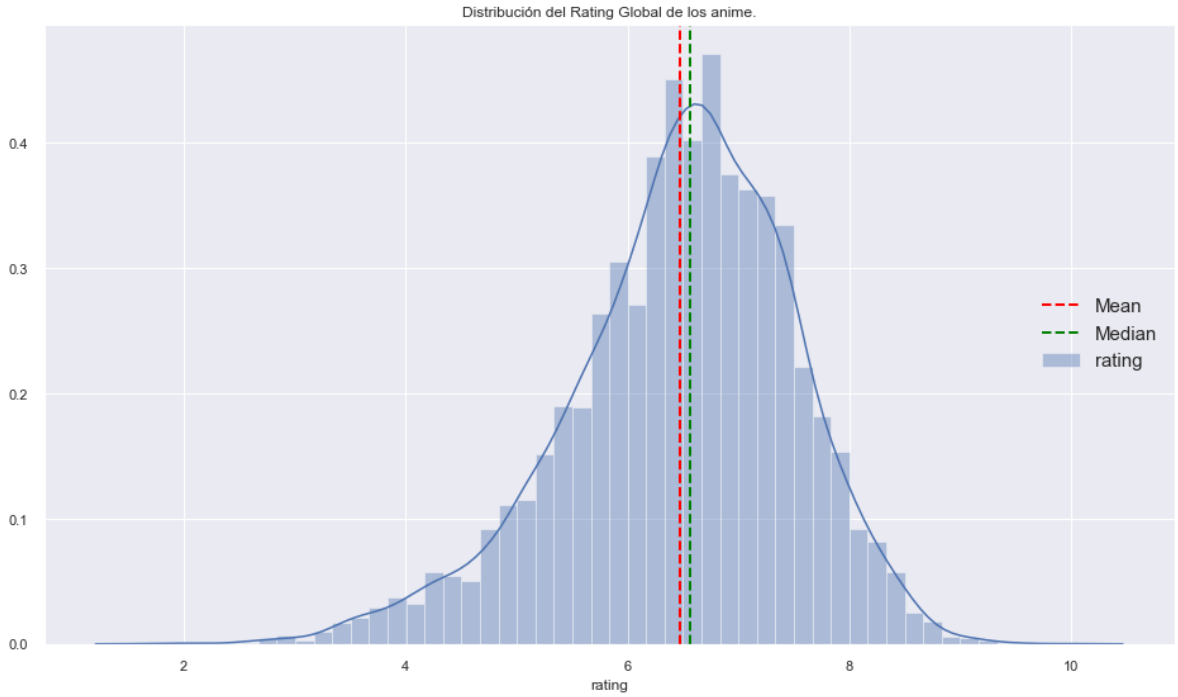
```
201           1
191           1
331           1
163           1
237           1
1006          1
694           1
124           1
1428          1
726           1
150           1
283           1
1818          1
1274          1
90            1
263           1
373           1
172           1
120           1
330           1
167           1
Name: episodes, Length: 187, dtype: int64
```

In [240]:
```
# Limpiamos y pasamos a float el atributo episodes
df_titulos["episodes"] = df_titulos["episodes"].replace("Unknown",None)
df_titulos["episodes"] = df_titulos["episodes"].astype(float)
```

In [241]:
```
var = "episodes"
sns.set(rc={'figure.figsize':(16,9)})
sns.distplot(df_titulos[var].dropna())
plt.axvline(df_titulos[var].mean(),lw=2, color='red', linestyle='--
')
plt.axvline(df_titulos[var].median(),lw=2, color='green', linestyle
='--')
plt.legend(loc=(5),fontsize=15,labels=["Mean","Median",var])
plt.title("Distribución de la Cantidad de Episodios de los anime.")
;
```



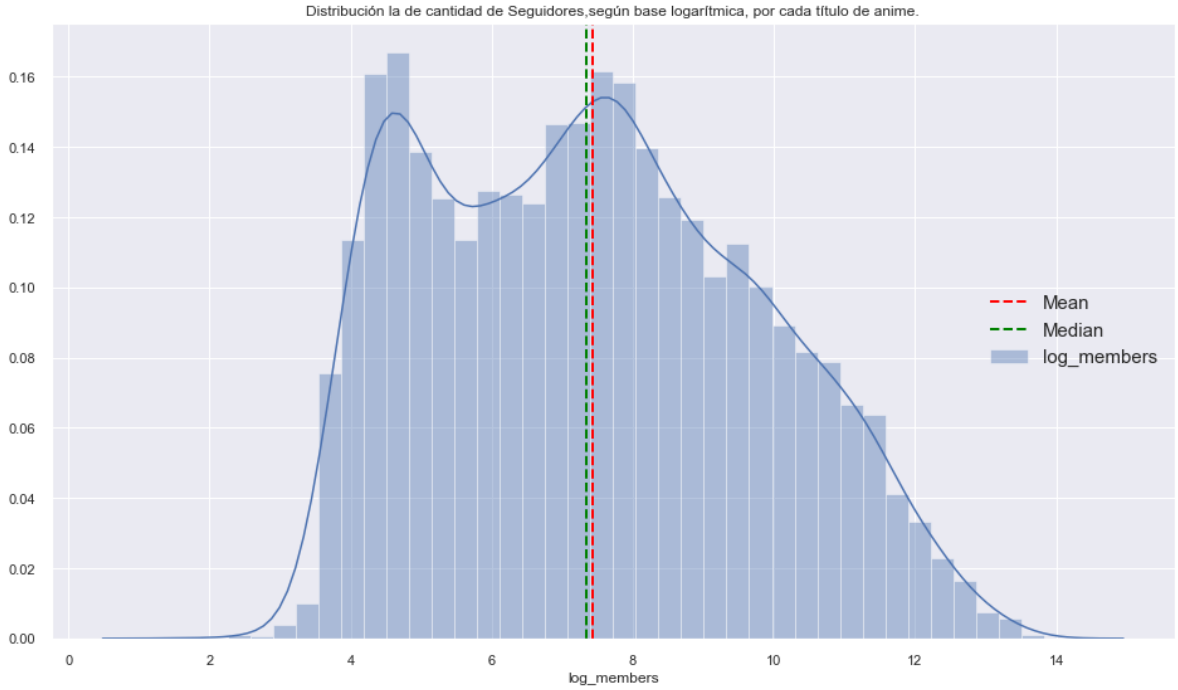
In [242]:
```
var = "rating"
sns.set(rc={'figure.figsize':(16,9)})
sns.distplot(df_titulos[var].dropna())
plt.axvline(df_titulos[var].mean(),lw=2, color='red', linestyle='--
')
plt.axvline(df_titulos[var].median(),lw=2, color='green', linestyle
='--')
plt.legend(loc=(5),fontsize=15,labels=["Mean","Median",var])
plt.title("Distribución del Rating Global de los anime.");
```



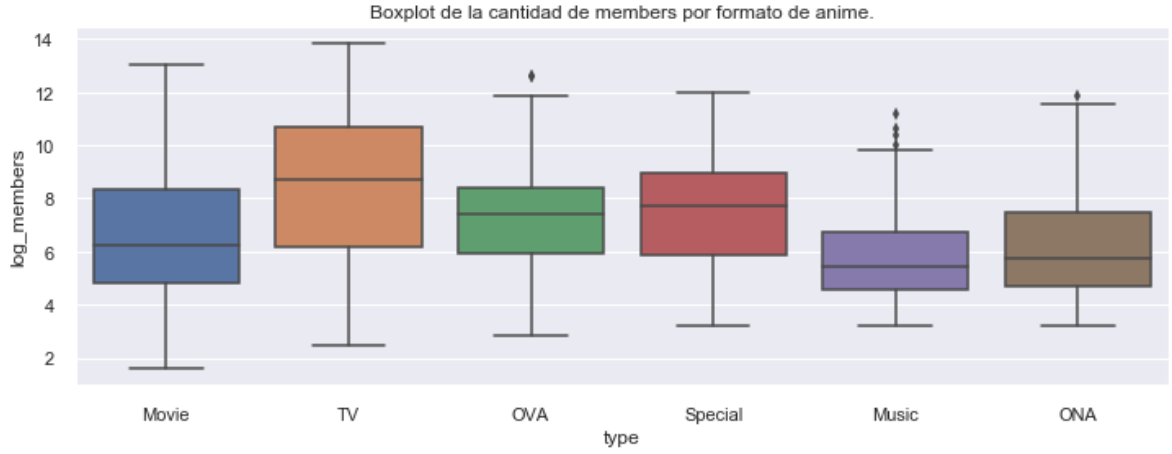
In [243]:
```
var = "members"
sns.set(rc={'figure.figsize':(12,4)})
sns.distplot(df_titulos[var].dropna())
plt.axvline(df_titulos[var].mean(),lw=2, color='red', linestyle='--
')
plt.axvline(df_titulos[var].median(),lw=2, color='green', linestyle
='--')
plt.legend(loc=(5),fontsize=15,labels=["Mean","Median",var])
plt.title("Distribución la de cantidad de Seguidores por cada títul
o de anime.");
```

In [244]:
```
df_titulos["log_members"] = np.log(df_titulos["members"])
```

In [245]:
```
var = "log_members"
sns.set(rc={'figure.figsize':(16,9)})
sns.distplot(df_titulos[var].dropna())
plt.axvline(df_titulos[var].mean(),lw=2, color='red', linestyle='--
')
plt.axvline(df_titulos[var].median(),lw=2, color='green', linestyle
='--')
plt.legend(loc=(5),fontsize=15,labels=["Mean","Median",var])
plt.title("Distribución la de cantidad de Seguidores,según base log
arítmica, por cada título de anime.");
```

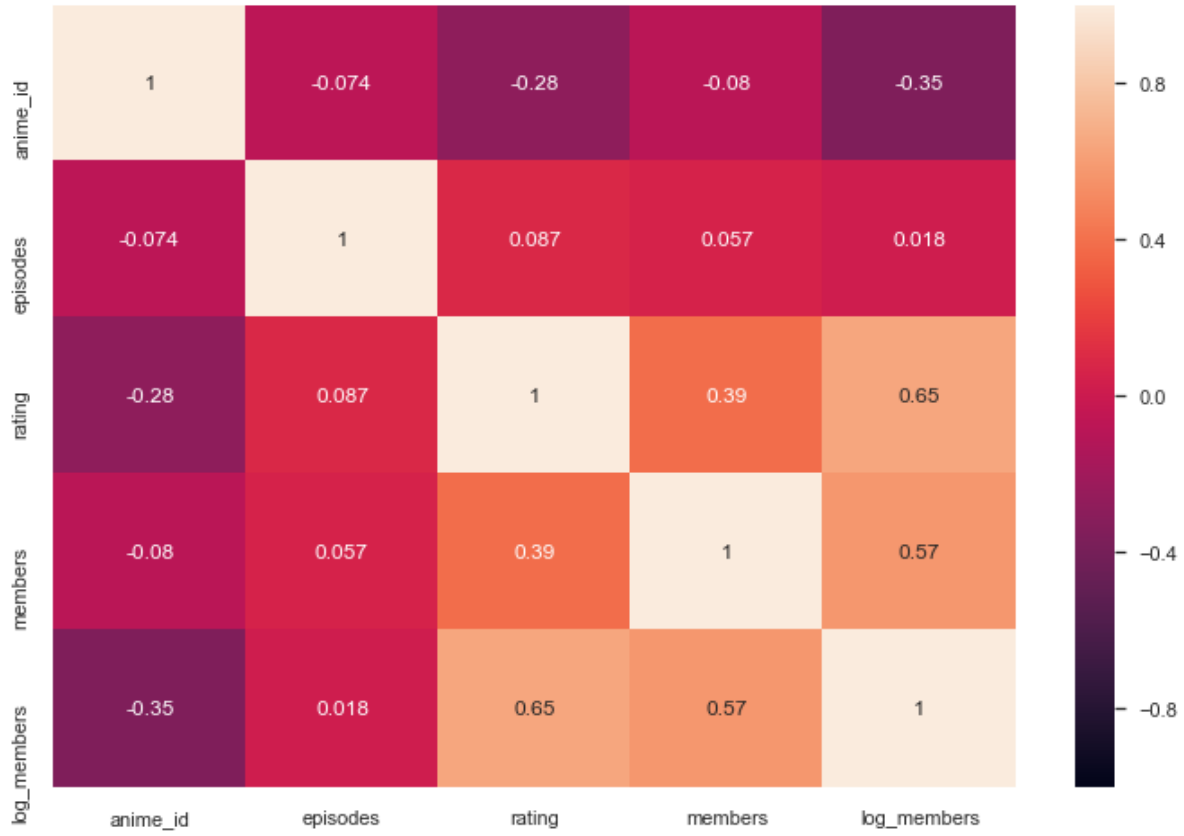


In [246]:
```
plt.figure(figsize=(12,4))
plt.title("Boxplot de la cantidad de members por formato de anime."
)
sns.boxplot(y=df_titulos["log_members"], x=df_titulos["type"]);
```



In [247]:
```
plt.subplots(figsize=(12, 8))
data = df_titulos
sns.heatmap(data.corr(), annot=True, vmin=-1, vmax=1)
```

Out[247]: <matplotlib.axes._subplots.AxesSubplot at 0x1a2a9ced30>



In [248]:
```
users = df_interacciones["user_id"].unique()
titulos_evaluados = df_interacciones["anime_id"].unique()
titulos_all = df_titulos["anime_id"].unique()
```

In [249]:
```
len(titulos_evaluados)
```

Out[249]: 11200

Relación entre data sets

In [250]:
```
# Se revisan titulos que se encuentren en df_interacciones y no en
df_titulos para sacarlos
registro = 0
titles_cruce = []
faltantes = []
for a in titulos_evaluados:
    faltantes.append(a)
    for b in titulos_all:
        if a == b:
            registro = registro +1
            titles_cruce.append(a)
            faltantes.remove(a)
            break
```

In [251]:
```
faltantes
```

Out[251]:
```
[30913, 30924, 20261]
```

In [252]:
```
#Se obtienen 3 registros faltantes en df_titulos que se proceden a
eliminar de df_interacciones
eliminar_1 = []
for i in faltantes:
    eliminar_1.append(df_interacciones[df_interacciones["anime_id"]
== i].index)
    df_interacciones.drop(df_interacciones[df_interacciones["anime_
id"] == i].index, inplace=True)
```

In [253]:
```
#total de registros eliminados
total = 0
for i in eliminar_1:
    largo=len(i)
    total = total + largo
total
```

Out[253]:
```
10
```

In [254]:
```
# Se revisan titulos que se encuentren en en df_titulos y no df_int
eracciones para sacarlos
registro2 = 0
titles_cruce2 = []
faltantes2 = []
for a in titulos_all:
    faltantes2.append(a)
    for b in titulos_evaluados:
        if a == b:
            registro2 = registro +1
            titles_cruce2.append(a)
            faltantes2.remove(a)
            break
```

In [255]:
```
len(faltantes2)
```

Out[255]: 1097

In [256]:
```
#Se obtienen 1.097 registros faltantes en df_interacciones que se p
roceden a eliminar de df_titulos
eliminar_2 = []
for i in faltantes2:
    eliminar_2.append(df_titulos[df_titulos["anime_id"] == i].index
)
    df_titulos.drop(df_titulos[df_titulos["anime_id"] == i].index,
inplace=True)
```

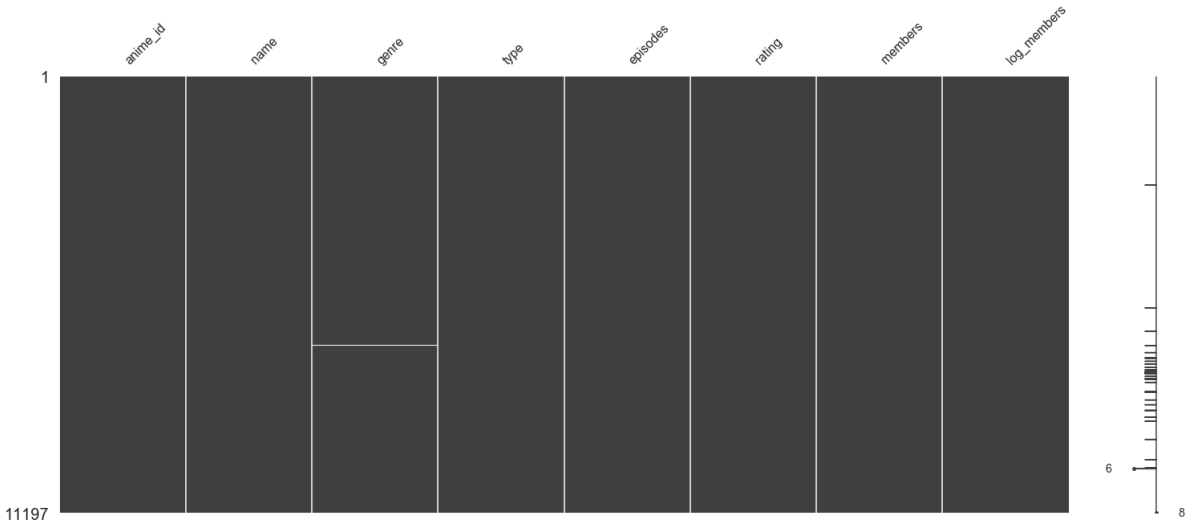
In [257]:
```
#total de registros eliminados
total2 = 0
for i in eliminar_2:
    largo=len(i)
    total2 = total2 + largo
total2
```

Out[257]: 1097

Completitud de datos de df_titulos

In [258]:
```
msngo.matrix(df_titulos) #re-exploramos la completitud de los datos
```

Out[258]: <matplotlib.axes._subplots.AxesSubplot at 0x1a3325a7b8>



In [259]:
```
len(df_titulos)
```

Out[259]: 11197

Se observa un aumento considerable en la completitud de los datos

In [260]:
```
# Re-exploración en detalle de los registros nulos
for i in df_titulos.columns:
    a= df_titulos[i].unique()
    print("-------------------------------")
    print("Columna: "+i)
    print(len(a)) #Exploramos la cantidad de categorías de cada atr
ibuto
    print(df_titulos[i].isnull().value_counts()) #Exploramos la can
tidad de valores nulo en los registros
    print("Detalle de registros nulos:")
    print(df_titulos[df_titulos[i].isnull()])
```

```
-------------------------------
Columna: anime_id
11197
False    11197
Name: anime_id, dtype: int64
Detalle de registros nulos:
Empty DataFrame
Columns: [anime_id, name, genre, type, episodes, rating, members,
log_members]
Index: []
-------------------------------
Columna: name
11196
False    11197
Name: name, dtype: int64
Detalle de registros nulos:
Empty DataFrame
Columns: [anime_id, name, genre, type, episodes, rating, members,
log_members]
Index: []
-------------------------------
Columna: genre
3155
False    11165
True        32
Name: genre, dtype: int64
Detalle de registros nulos:
       anime_id                                          name genre
type  \
2844      33242  IS: Infinite Stratos 2 - Infinite Wedding   NaN
Special
6040      29765                          Metropolis (2009)   NaN
Movie
6646      32695                               Match Shoujo   NaN
ONA
7018      33187                             Katsudou Shashin   NaN
Movie
7198      30862                         Yubi wo Nusunda Onna   NaN
Movie
7335      28987                                    Kamakura   NaN
Movie
```

7349	19219	Modern No.2	NaN	Movie
7426	29629	Coffee Break	NaN	Movie
7498	28653	Maze	NaN	Movie
7591	31834	Mormorando	NaN	Movie
7645	31507	Ari Ningen Monogatari	NaN	Movie
7685	31760	Tsuru Shitae Waka Kan	NaN	Movie
7708	28587	Modern	NaN	Movie
7716	31831	Fantasy	NaN	Movie
7738	31833	Metamorphose	NaN	Movie
7759	30399	Arigatou Gomennasai	NaN	Movie
7824	28655	PiKA PiKA	NaN	Movie
7876	31832	Zawazawa	NaN	Movie
7899	28647	Kappo	NaN	Movie
7982	29957	Count Down	NaN	Movie
8279	29921	Bunbuku Chagama (1958)	NaN	Movie
8304	29655	Chanda Gou	NaN	Movie
8565	29923	Fukusuke	NaN	Movie
8716	31510	Guitar	NaN	Movie
8897	32636	Hokori Inu no Hanashi	NaN	ONA
8900	31511	Holiday	NaN	Movie
9137	30435	Kankou Taisen Saitama: Sakuya no Tatakai	NaN	ONA
9142	31506	Kappa no Ude	NaN	Movie
9265	29920	Kobutori (1957)	NaN	Movie
9903	29922	Ou-sama Ninatta Kitsune	NaN	Movie
10575	30408	Tokyo SOS	NaN	Movie
10863	30309	Yuuyake Dandan	NaN	Movie

	episodes	rating	members	log_members

```
2844        1.0    7.15     6604     8.795431
6040        1.0    6.27      313     5.746203
6646        1.0    6.02      242     5.488938
7018        1.0    5.79      607     6.408529
7198        1.0    5.65      223     5.407172
7335        1.0    5.53      164     5.099866
7349        1.0    5.52      374     5.924256
7426        1.0    5.44      265     5.579730
7498        1.0    5.37      138     4.927254
7591        1.0    5.26      181     5.198497
7645        1.0    5.20      191     5.252273
7685        1.0    5.15      195     5.273000
7708        1.0    5.12      237     5.468060
7716        1.0    5.12      193     5.262690
7738        1.0    5.08      175     5.164786
7759        1.0    5.05      115     4.744932
7824        1.0    4.92      289     5.666427
7876        1.0    4.80      216     5.375278
7899        1.0    4.71      335     5.814131
7982        1.0    4.27      231     5.442418
8279        1.0    5.52       86     4.454347
8304        1.0    4.42       91     4.510860
8565        1.0    4.69      103     4.634729
8716        1.0    4.11       47     3.850148
8897        1.0    4.36       51     3.931826
8900        1.0    6.56       37     3.610918
9137        4.0    4.24      103     4.634729
9142        1.0    4.46       62     4.127134
9265        1.0    4.75       90     4.499810
9903        1.0    4.16       74     4.304065
10575       1.0    2.72       87     4.465908
10863       6.0    5.55      542     6.295266
------------------------------
Columna: type
7
False    11196
True         1
Name: type, dtype: int64
Detalle de registros nulos:
       anime_id           name             genre type  episodes  r
ating  \
10898     30484  Steins;Gate 0  Sci-Fi, Thriller  NaN      32.0
NaN

       members  log_members
10898    60999    11.018613
------------------------------
Columna: episodes
183
False    11197
Name: episodes, dtype: int64
Detalle de registros nulos:
Empty DataFrame
Columns: [anime_id, name, genre, type, episodes, rating, members,
```

```
log_members]
Index: []
------------------------------
Columna: rating
586
False    11194
True         3
Name: rating, dtype: int64
Detalle de registros nulos:
       anime_id                    name  \
10898     30484          Steins;Gate 0
10919     33674  No Game No Life Movie
10951      9488            Cencoroll 2

                                                  genre   type  e
pisodes  \
10898                                 Sci-Fi, Thriller    NaN
32.0
10919  Adventure, Comedy, Ecchi, Fantasy, Game, Super...  Movie
1.0
10951                                   Action, Sci-Fi  Movie
1.0

       rating  members  log_members
10898     NaN    60999    11.018613
10919     NaN    32041    10.374772
10951     NaN    15181     9.627800
------------------------------
Columna: members
6487
False    11197
Name: members, dtype: int64
Detalle de registros nulos:
Empty DataFrame
Columns: [anime_id, name, genre, type, episodes, rating, members,
log_members]
Index: []
------------------------------
Columna: log_members
6487
False    11197
Name: log_members, dtype: int64
Detalle de registros nulos:
Empty DataFrame
Columns: [anime_id, name, genre, type, episodes, rating, members,
log_members]
Index: []
```

In [261]:
```
df_titulos.loc[df_titulos["anime_id"]==30484,"type"] = "ONA"
# Se añade la información de la cantidad de capítulos source:
#https://en.wikipedia.org/wiki/Steins;Gate_0_(TV_series)
```

In [262]:
```
df_titulos.loc[df_titulos["anime_id"]==30484,"episodes"] = 23
# Se añade la información de la cantidad de capitulos del anime:
#https://www.imdb.com/title/tt5514358/
df_titulos.loc[df_titulos["anime_id"]==33674,"episodes"] = 1
# Se añade la información de la cantidad de capitulos del anime:
# Dado que es del formato Movie, episodes = 1
df_titulos.loc[df_titulos["anime_id"]==9488,"episodes"] = 1
# Se añade la información de la cantidad de capitulos del anime:
# Dado que es del formato Movie, episodes = 1
```

In [263]:
```
df_titulos.loc[df_titulos["anime_id"]==30484,"rating"] = 8.4
# Se añade la información del rating global del anime:
#https://www.imdb.com/title/tt5514358/
df_titulos.loc[df_titulos["anime_id"]==33674,"rating"] = 7.9
# Se añade la información del rating global del anime:
#https://www.imdb.com/title/tt3431758/
df_titulos.loc[df_titulos["anime_id"]==9488,"rating"] = 6.7
# Se añade la información del rating global del anime:
#https://www.imdb.com/title/tt3741646/
```

In [264]:
```
# Observamos la completitud de los datos, excepto en los géneros,
# que serán dummizados por lo que quedarán representada 0 en todos
los campos dummies
for i in df_titulos.columns:
    a= df_titulos[i].unique()
    print("--------------------------------")
    print("Columna: "+i)
    print(len(a)) #Exploramos la cantidad de categorías de cada atr
ibuto
    print(df_titulos[i].isnull().value_counts()) #Exploramos la can
tidad de valores nulo en los registros
    print("Detalle de registros nulos:")
    print(df_titulos[df_titulos[i].isnull()])
```

```
--------------------------------
Columna: anime_id
11197
False    11197
Name: anime_id, dtype: int64
Detalle de registros nulos:
Empty DataFrame
Columns: [anime_id, name, genre, type, episodes, rating, members,
log_members]
Index: []
--------------------------------
Columna: name
11196
False    11197
Name: name, dtype: int64
Detalle de registros nulos:
Empty DataFrame
Columns: [anime_id, name, genre, type, episodes, rating, members,
```

```
log_members]
Index: []
--------------------------------
Columna: genre
3155
False    11165
True        32
Name: genre, dtype: int64
Detalle de registros nulos:
       anime_id                                          name genre
type  \
2844      33242  IS: Infinite Stratos 2 - Infinite Wedding   NaN
Special
6040      29765                          Metropolis (2009)   NaN
Movie
6646      32695                               Match Shoujo   NaN
ONA
7018      33187                            Katsudou Shashin   NaN
Movie
7198      30862                       Yubi wo Nusunda Onna   NaN
Movie
7335      28987                                    Kamakura   NaN
Movie
7349      19219                                  Modern No.2   NaN
Movie
7426      29629                                  Coffee Break   NaN
Movie
7498      28653                                         Maze   NaN
Movie
7591      31834                                  Mormorando   NaN
Movie
7645      31507                       Ari Ningen Monogatari   NaN
Movie
7685      31760                       Tsuru Shitae Waka Kan   NaN
Movie
7708      28587                                       Modern   NaN
Movie
7716      31831                                      Fantasy   NaN
Movie
7738      31833                                  Metamorphose   NaN
Movie
7759      30399                        Arigatou Gomennasai   NaN
Movie
7824      28655                                    PiKA PiKA   NaN
Movie
7876      31832                                     Zawazawa   NaN
Movie
7899      28647                                        Kappo   NaN
Movie
7982      29957                                  Count Down   NaN
Movie
8279      29921                       Bunbuku Chagama (1958)   NaN
Movie
8304      29655                                  Chanda Gou   NaN
```

```
Movie
8565      29923                                  Fukusuke   NaN
Movie
8716      31510                                    Guitar   NaN
Movie
8897      32636                      Hokori Inu no Hanashi   NaN
ONA
8900      31511                                   Holiday   NaN
Movie
9137      30435  Kankou Taisen Saitama: Sakuya no Tatakai   NaN
ONA
9142      31506                               Kappa no Ude   NaN
Movie
9265      29920                           Kobutori (1957)   NaN
Movie
9903      29922                    Ou-sama Ninatta Kitsune   NaN
Movie
10575     30408                                 Tokyo SOS   NaN
Movie
10863     30309                              Yuuyake Dandan   NaN
Movie

       episodes  rating  members  log_members
2844        1.0    7.15     6604     8.795431
6040        1.0    6.27      313     5.746203
6646        1.0    6.02      242     5.488938
7018        1.0    5.79      607     6.408529
7198        1.0    5.65      223     5.407172
7335        1.0    5.53      164     5.099866
7349        1.0    5.52      374     5.924256
7426        1.0    5.44      265     5.579730
7498        1.0    5.37      138     4.927254
7591        1.0    5.26      181     5.198497
7645        1.0    5.20      191     5.252273
7685        1.0    5.15      195     5.273000
7708        1.0    5.12      237     5.468060
7716        1.0    5.12      193     5.262690
7738        1.0    5.08      175     5.164786
7759        1.0    5.05      115     4.744932
7824        1.0    4.92      289     5.666427
7876        1.0    4.80      216     5.375278
7899        1.0    4.71      335     5.814131
7982        1.0    4.27      231     5.442418
8279        1.0    5.52       86     4.454347
8304        1.0    4.42       91     4.510860
8565        1.0    4.69      103     4.634729
8716        1.0    4.11       47     3.850148
8897        1.0    4.36       51     3.931826
8900        1.0    6.56       37     3.610918
9137        4.0    4.24      103     4.634729
9142        1.0    4.46       62     4.127134
9265        1.0    4.75       90     4.499810
9903        1.0    4.16       74     4.304065
10575       1.0    2.72       87     4.465908
```

```
10863       6.0    5.55      542     6.295266
--------------------------------
Columna: type
6
False    11197
Name: type, dtype: int64
Detalle de registros nulos:
Empty DataFrame
Columns: [anime_id, name, genre, type, episodes, rating, members,
log_members]
Index: []
--------------------------------
Columna: episodes
183
False    11197
Name: episodes, dtype: int64
Detalle de registros nulos:
Empty DataFrame
Columns: [anime_id, name, genre, type, episodes, rating, members,
log_members]
Index: []
--------------------------------
Columna: rating
585
False    11197
Name: rating, dtype: int64
Detalle de registros nulos:
Empty DataFrame
Columns: [anime_id, name, genre, type, episodes, rating, members,
log_members]
Index: []
--------------------------------
Columna: members
6487
False    11197
Name: members, dtype: int64
Detalle de registros nulos:
Empty DataFrame
Columns: [anime_id, name, genre, type, episodes, rating, members,
log_members]
Index: []
--------------------------------
Columna: log_members
6487
False    11197
Name: log_members, dtype: int64
Detalle de registros nulos:
Empty DataFrame
Columns: [anime_id, name, genre, type, episodes, rating, members,
log_members]
Index: []
```

Dummizar atributo genre

In [323]:
```
subgenre = ["Action","Adventure","Cars","Comedy","Dementia","Demons
","Drama","Ecchi","Fantasy","Game","Harem","Hentai","Historical","H
orror","Josei","Kids","Magic","MartialArts","Mecha","Military","Mus
ic","Mystery","Parody","Police","Psychological","Romance","Samurai"
,
"School","Sci-Fi","Seinen","Shoujo","ShoujoAi","Shounen","ShounenAi
","SliceofLife","Space","Sports","Supernatural","SuperPower","Thril
ler","Vampire","Yaoi","Yuri"]
```

In [266]:
```
for subg in subgenre:
    df_titulos[subg] = None
df_titulos.head()
```

Out[266]:

	anime_id	name	genre	type	episodes	rating	members	log_members
0	32281	Kimi no Na wa.	Drama, Romance, School, Supernatural	Movie	1.0	9.37	200630	12.209218
1	5114	Fullmetal Alchemist: Brotherhood	Action, Adventure, Drama, Fantasy, Magic, Mili...	TV	64.0	9.26	793665	13.584417
2	28977	Gintama°	Action, Comedy, Historical, Parody, Samurai, S...	TV	51.0	9.25	114262	11.646249
3	9253	Steins;Gate	Sci-Fi, Thriller	TV	24.0	9.17	673572	13.420350
4	9969	Gintama'	Action, Comedy, Historical, Parody, Samurai, S...	TV	51.0	9.16	151266	11.926795

5 rows × 51 columns

In [267]:
```
df_titulos.reset_index(inplace=True)
df_titulos.head()
```

Out[267]:

	index	anime_id	name	genre	type	episodes	rating	members	log_me
0	0	32281	Kimi no Na wa.	Drama, Romance, School, Supernatural	Movie	1.0	9.37	200630	12.
1	1	5114	Fullmetal Alchemist: Brotherhood	Action, Adventure, Drama, Fantasy, Magic, Mili...	TV	64.0	9.26	793665	13.
2	2	28977	Gintama°	Action, Comedy, Historical, Parody, Samurai, S...	TV	51.0	9.25	114262	11.
3	3	9253	Steins;Gate	Sci-Fi, Thriller	TV	24.0	9.17	673572	13.
4	4	9969	Gintama'	Action, Comedy, Historical, Parody, Samurai, S...	TV	51.0	9.16	151266	11.

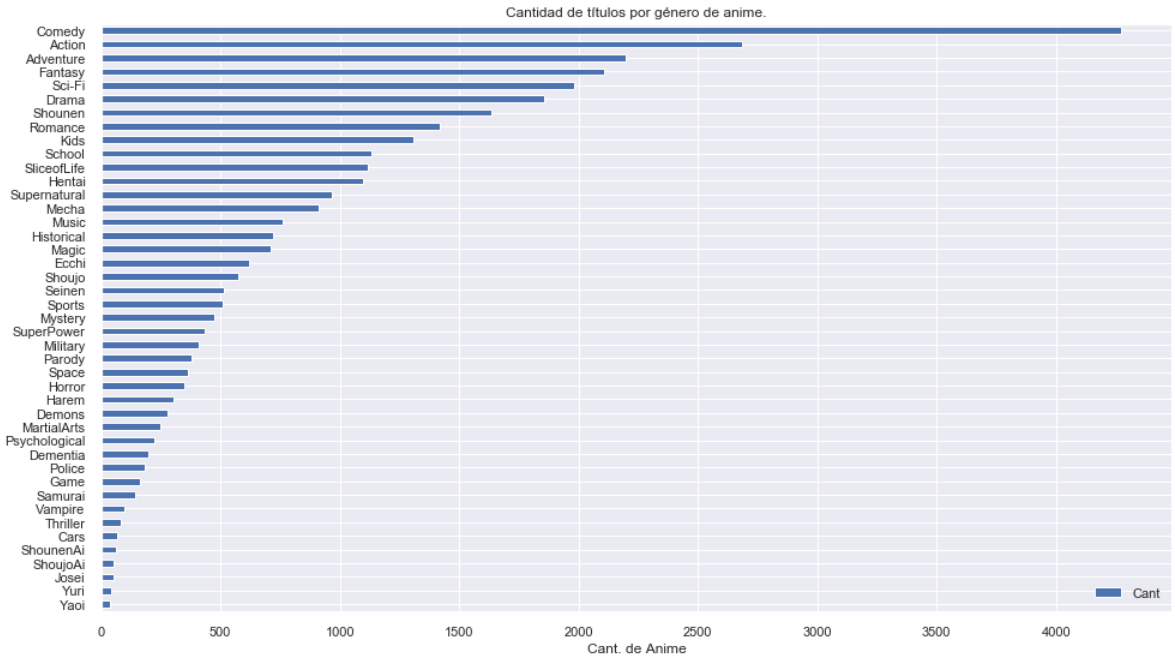
5 rows × 52 columns

In [268]:
```
columnas = list(df_titulos.columns)
for n, i in enumerate(df_titulos["genre"]):
    if isinstance(i,str):
        tmp = i.split(",")
        for j in tmp:
            jj = j.strip().replace(" ", "")
            df_titulos.iloc[n,columnas.index(jj)] = 1
```

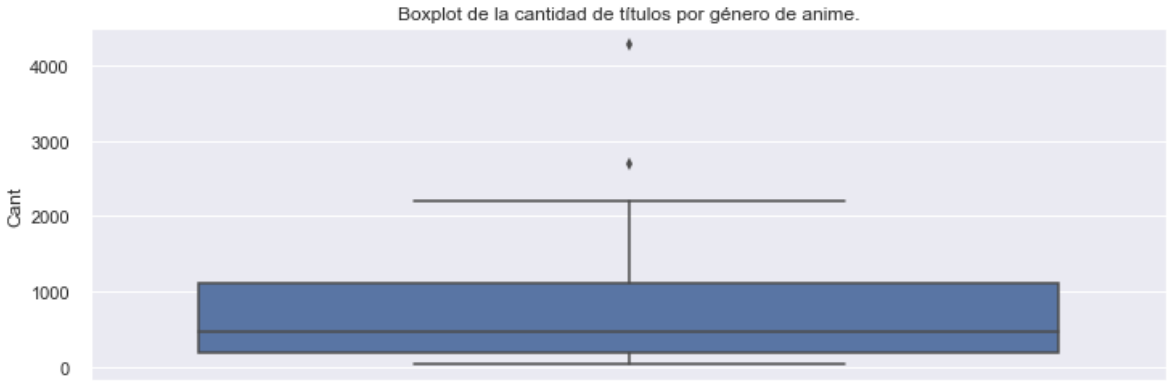
In [269]:
```
# Creamos una tabla resumen de la cantidad de titulos por genero de anime
frec_genres = []
for i in subgenre:
    frec_genres.append(df_titulos[i].sum())
df_genres = pd.DataFrame({"subgenre":subgenre,"Cant":frec_genres})
df_genres.sort_values("Cant",inplace=True, ascending=True)
```

In [270]:
```
plt.figure(figsize=(12,24));
df_genres.plot(kind= 'barh');
plt.yticks(np.arange(len(df_genres["subgenre"])), df_genres["subgenre"]);
plt.xlabel("Cant. de Anime")
plt.title("Cantidad de títulos por género de anime.");
```

```
<Figure size 864x1728 with 0 Axes>
```



In [271]:
```
plt.figure(figsize=(12,4))
plt.title("Boxplot de la cantidad de títulos por género de anime.")
sns.boxplot(y=df_genres["Cant"]);
```



In [272]:
```
df_interacciones.head()
```

Out[272]:

	user_id	anime_id	rating
0	1	20	-1
1	1	24	-1
2	1	79	-1
3	1	226	-1
4	1	241	-1

In [273]:
```
df_titulos.head()
```

Out[273]:

	index	anime_id	name	genre	type	episodes	rating	members	log_me
0	0	32281	Kimi no Na wa.	Drama, Romance, School, Supernatural	Movie	1.0	9.37	200630	12.
1	1	5114	Fullmetal Alchemist: Brotherhood	Action, Adventure, Drama, Fantasy, Magic, Mili...	TV	64.0	9.26	793665	13.
2	2	28977	Gintama°	Action, Comedy, Historical, Parody, Samurai, S...	TV	51.0	9.25	114262	11.
3	3	9253	Steins;Gate	Sci-Fi, Thriller	TV	24.0	9.17	673572	13.
4	4	9969	Gintama'	Action, Comedy, Historical, Parody, Samurai, S...	TV	51.0	9.16	151266	11.

5 rows × 52 columns

In [274]:
```
df_interacciones.shape
```

Out[274]: (7813727, 3)

In [275]:
```
df_titulos.shape
```

Out[275]: (11197, 52)

In [276]:
```
merged_interacciones = pd.merge(left=df_interacciones, left_on='anime_id',
                                right=df_titulos, right_on='anime_id')

merged_interacciones.shape
```

Out[276]: (7813727, 54)

In [277]:
```
merged_interacciones.head()
```

Out[277]:

	user_id	anime_id	rating_x	index	name	genre	type	episodes	rating_y	members
0	1	20	-1	841	Naruto	Action, Comedy, Martial Arts, Shounen, Super P...	TV	220.0	7.81	683297
1	3	20	8	841	Naruto	Action, Comedy, Martial Arts, Shounen, Super P...	TV	220.0	7.81	683297
2	5	20	6	841	Naruto	Action, Comedy, Martial Arts, Shounen, Super P...	TV	220.0	7.81	683297
3	6	20	-1	841	Naruto	Action, Comedy, Martial Arts, Shounen, Super P...	TV	220.0	7.81	683297
4	10	20	-1	841	Naruto	Action, Comedy, Martial Arts, Shounen, Super P...	TV	220.0	7.81	683297

5 rows × 54 columns

In [278]:
```
merged_interacciones.rename(columns={"rating_x": "rating", "rating_
y": "rating_global"},inplace=True)
merged_interacciones.drop(["index","genre"],axis=1,inplace=True)
merged_interacciones.columns
```

Out[278]:
```
Index(['user_id', 'anime_id', 'rating', 'name', 'type', 'episodes'
,
       'rating_global', 'members', 'log_members', 'Action', 'Adven
ture',
       'Cars', 'Comedy', 'Dementia', 'Demons', 'Drama', 'Ecchi', '
Fantasy',
       'Game', 'Harem', 'Hentai', 'Historical', 'Horror', 'Josei',
'Kids',
       'Magic', 'MartialArts', 'Mecha', 'Military', 'Music', 'Myst
ery',
       'Parody', 'Police', 'Psychological', 'Romance', 'Samurai',
'School',
       'Sci-Fi', 'Seinen', 'Shoujo', 'ShoujoAi', 'Shounen', 'Shoun
enAi',
       'SliceofLife', 'Space', 'Sports', 'Supernatural', 'SuperPow
er',
       'Thriller', 'Vampire', 'Yaoi', 'Yuri'],
      dtype='object')
```

In [279]:
```
merged_interacciones.head()
```

Out[279]:

	user_id	anime_id	rating	name	type	episodes	rating_global	members	log_members
0	1	20	-1	Naruto	TV	220.0	7.81	683297	13.434685
1	3	20	8	Naruto	TV	220.0	7.81	683297	13.434685
2	5	20	6	Naruto	TV	220.0	7.81	683297	13.434685
3	6	20	-1	Naruto	TV	220.0	7.81	683297	13.434685
4	10	20	-1	Naruto	TV	220.0	7.81	683297	13.434685

5 rows × 52 columns

Creación de los indicadores para medir las interacciones usuarios - anime

Creación Indicador de Afinidad Se crea un indicador de afinidad entre un usuario dado y un anime dado, de acuerdo a los siguientes criterios, que se aplican en orden:

Si el rating otorgado por el usuario es superior al rating global del anime = **5 puntos**

Si el rating otorgado por el usuario es superior a 5 = **4 puntos**

Si el usuario a visualizado la película = **3 puntos**

Si el rating otorgado por el usuario es inferior a 5 = **0 puntos**

*Posteriormente se le asignarán 2 puntos a todas las interacciones no existentes como puntaje basal**

In [280]:
```
merged_interacciones["afinidad"]=None
```

In [281]:
```
# Se implementa el indicador definido

condiciones = [(merged_interacciones.rating <= 5) & (merged_interac
ciones.rating > -1) ,
               (merged_interacciones.rating == -1),
               (merged_interacciones.rating > 5) & (merged_interacc
iones.rating < merged_interacciones.rating_global),
               (merged_interacciones.rating > 5) & (merged_interacc
iones.rating >= merged_interacciones.rating_global)
               ]
afinidades = np.array((0, 3, 4, 5), dtype="int8")
merged_interacciones["afinidad"] = np.select(condiciones, afinidade
s, None)
```

Creación Indicador Apriori Se crea un indicador de afinidad entre un usuario dado y un anime dado, para ser utilizado con el algoritmo apriori de caracter binario, de acuerdo a los siguientes criterios, que se aplican en orden:

Si el rating otorgado por el usuario es superior a 6 = **1**

Si el usuario a visualizado la película = **1**

En todos los otros casos se otorgará un **0**

*Posteriormente se construirán listas de relaciones de cada user_id, en donde sólo se incluirán los anime_id con indicador 1**

In [282]:
```
merged_interacciones["apriori"]=None
```

In [283]:
```
# Se implementa el indicador definido

condiciones2 = [(merged_interacciones.rating > 5),
                (merged_interacciones.rating == -1)
               ]
apriori = np.array((1, 1), dtype="int8")
merged_interacciones["apriori"] = np.select(condiciones2, apriori,
0)
```

In [284]:
```
merged_interacciones.head()
```

Out[284]:

	user_id	anime_id	rating	name	type	episodes	rating_global	members	log_members
0	1	20	-1	Naruto	TV	220.0	7.81	683297	13.434685
1	3	20	8	Naruto	TV	220.0	7.81	683297	13.434685
2	5	20	6	Naruto	TV	220.0	7.81	683297	13.434685
3	6	20	-1	Naruto	TV	220.0	7.81	683297	13.434685
4	10	20	-1	Naruto	TV	220.0	7.81	683297	13.434685

5 rows × 54 columns

Exportamos a .csv los nuevos data sets creados

In [ ]:
```
merged_interacciones.to_csv('interacciones_generos.csv',sep=';')
df_titulos.to_csv('titulos_generos.csv',sep=';')
```

Limpieza del Data Sets de Interacciones

Se procede a la limpieza de los data set, tomando sólo la data con un valor predictivo alto, eliminado los registro de usuarios anómalos, y/o animes con bajas interacciones.

In [285]:
```
def fx_users_animes(data,limit_up=5000,limit_down=10):
### Función devuelve un data frame con los usuarios
### y su frecuencia de aparición, de igual forma devuelvo otro
### con la misma información para los títulos de anime

### limit_up: default =5000 , la cantidad máxima de interacciones a
ceptadas para un usuario, si lo sobre pasa, elimina al usuario.
### limit_down: default =10 , la cantidad mínima de interacciones a
ceptadas para un usuario, si tienes menos, elimina al usuario.

    # Se realiza una tabla resumen de las cantidades de interaccion
es por user_id y por anime_id respectivamente

    df_users = pd.DataFrame({"users":data["user_id"].value_counts()
.index,"Frec":data["user_id"].value_counts()})
    df_users.sort_values("Frec",inplace=True, ascending=False)
    df_users = df_users.reset_index()

    df_animes = pd.DataFrame({"animes":data["anime_id"].value_count
s().index,"Frec":data["anime_id"].value_counts()})
    df_animes.sort_values("Frec",inplace=True, ascending=False)
    df_animes = df_animes.reset_index()

    # Se eliminan los valores sobre 5.000, dado que probablemente n
o corresponde a humanos
    #y los con una sola interacción dado que no aportan para predec
ir la recomendación
    df_users = df_users[df_users["Frec"] >= limit_down]
    df_users = df_users[df_users["Frec"] <= limit_up]

    # Se eliminan los anime con una sola interacción dado que no ap
ortan para predecir la recomendación
    df_animes = df_animes[df_animes["Frec"] != 1]
    return df_users , df_animes
```

In [324]:
```
def analisis_anomalos(df,var="Frec",title=""):

# Esta función realiza el analisis estadistico por
# cuartiles para determinan los umbrales de corte en
# la cantidad de interacciones

    q1,q3 = df[var].quantile([.25, .75])
    print("Q1")
    print(q1)
    print("Q3")
    print(q3)
    iqc = q3-q1
    mini_u = q1-1.5*iqc
    maxi_u = q3+1.5*iqc
    print("Corte Inferior:")
    print(mini_u)
    print("Corte Superior:")
    print(maxi_u)
    plt.figure(figsize=(16,16))
    plt.title("Boxplot de la cantidad de interacciones."+title)
    sns.boxplot(y=df[var]);
    print("----------------------")

    return mini_u , maxi_u

def recorte(df, maxi_corte, var = "Frec"):
    df_work = df[df[var] <= maxi_corte]
    print(len(df_work))
    return df_work
```

Selección de Datos a trabajar: recorte por users

In [287]:
```
users_works,animes_works=fx_users_animes(merged_interacciones)
```

In [288]:
```
len(users_works)
```

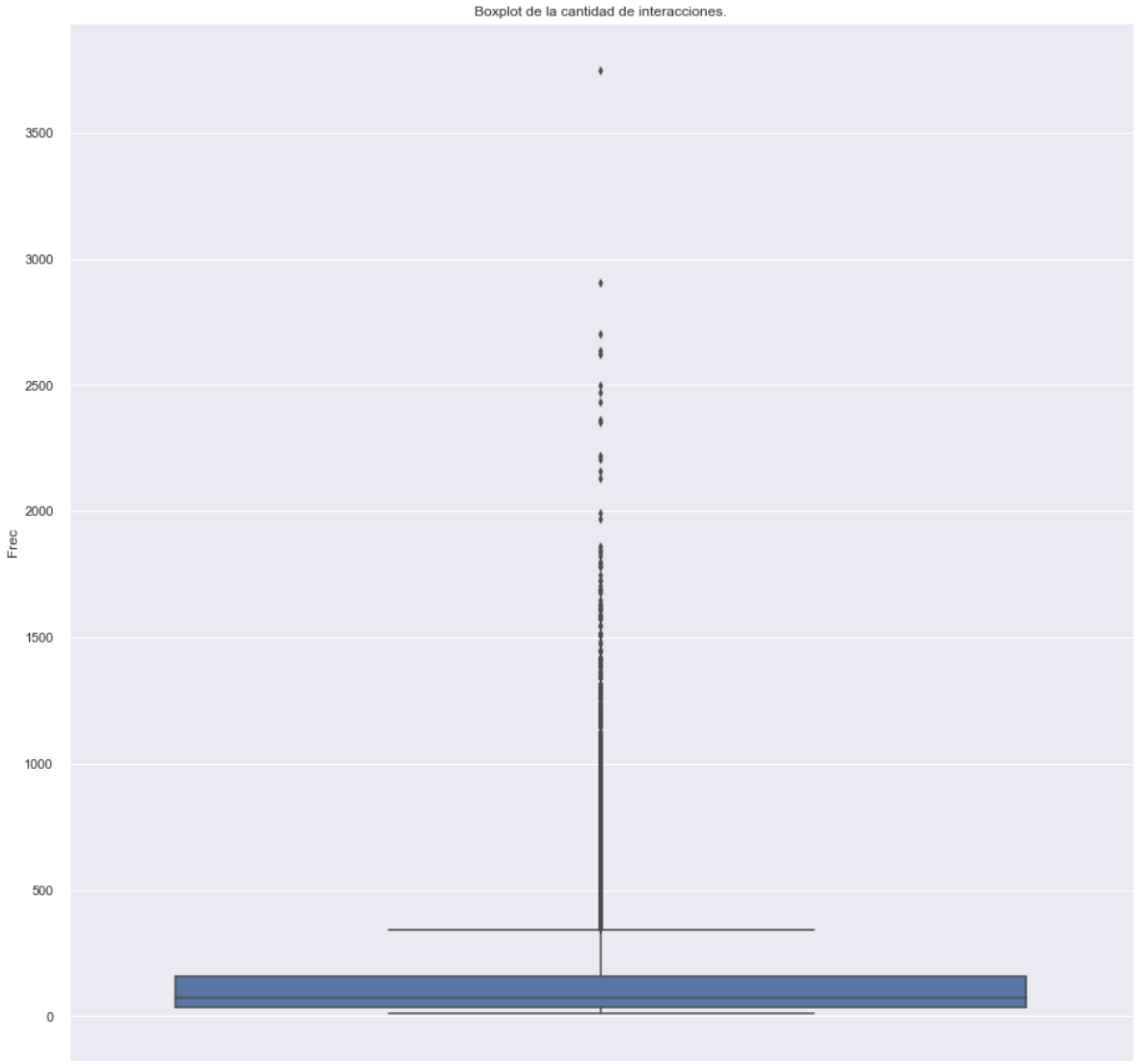
Out[288]: 61772

In [289]:
```
len(animes_works)
```

Out[289]: 9840

In [290]:
```
mi,ma=analisis_anomalos(users_works)
```

```
Q1
34.0
Q3
158.0
Corte Inferior:
-152.0
Corte Superior:
344.0
----------------------
```



Sólo se trabajará con los registros de usuarios bajo 344 interacciones y sobre 10 interacciones

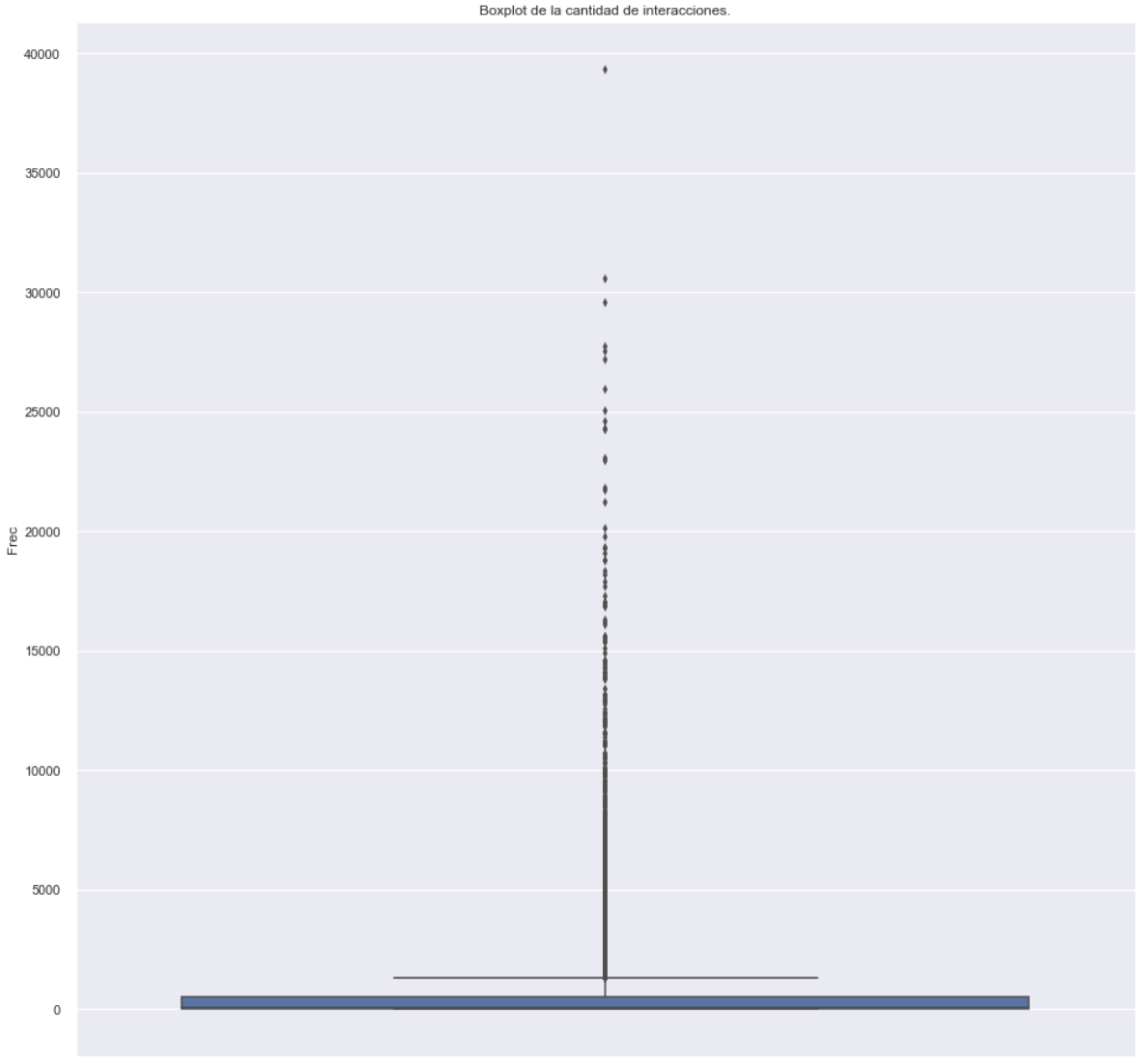
In [291]:
```
users_vf = recorte(users_works,ma)
```

```
57449
```

Selección de Datos a trabajar: recorte por títulos

In [292]:
```
mi_a,ma_a=analisis_anomalos(animes_works)
```

```
Q1
14.0
Q3
523.0
Corte Inferior:
-749.5
Corte Superior:
1286.5
----------------------
```



Sólo se trabajará con los registros de animes bajo 1.287 interacciones y sobre 10 interacciones

In [293]:
```
animes_vf = recorte(animes_works,1287)
```

```
8386
```

Realizamos el procesamiento final, dejando sólo los registros según las condiciones anteriores

In [295]:
```
merged_interacciones.shape
```

Out[295]:
```
(7813727, 54)
```

In [296]:
```
tmp = pd.merge(left=merged_interacciones, left_on='anime_id',
               right=animes_vf, right_on='animes')
```

In [297]:
```
df_interacciones = pd.merge(left=tmp, left_on='user_id',
                            right=users_vf, right_on='users')

df_interacciones.columns
```

Out[297]:
```
Index(['user_id', 'anime_id', 'rating', 'name', 'type', 'episodes'
,
       'rating_global', 'members', 'log_members', 'Action', 'Adven
ture',
       'Cars', 'Comedy', 'Dementia', 'Demons', 'Drama', 'Ecchi', '
Fantasy',
       'Game', 'Harem', 'Hentai', 'Historical', 'Horror', 'Josei',
'Kids',
       'Magic', 'MartialArts', 'Mecha', 'Military', 'Music', 'Myst
ery',
       'Parody', 'Police', 'Psychological', 'Romance', 'Samurai',
'School',
       'Sci-Fi', 'Seinen', 'Shoujo', 'ShoujoAi', 'Shounen', 'Shoun
enAi',
       'SliceofLife', 'Space', 'Sports', 'Supernatural', 'SuperPow
er',
       'Thriller', 'Vampire', 'Yaoi', 'Yuri', 'afinidad', 'apriori
', 'index_x',
       'animes', 'Frec_x', 'index_y', 'users', 'Frec_y'],
      dtype='object')
```

In [298]:
```
df_interacciones.rename(columns={"Frec_x": "Frec_Anime", "Frec_y":
"Frec_User"},inplace=True)
df_interacciones.drop(["index_x","index_y","animes","users"],axis=1
,inplace=True)
df_interacciones.head()
```

Out[298]:

	user_id	anime_id	rating	name	type	episodes	rating_global	members	log_n
0	1	1692	-1	_Summer	OVA	2.0	5.88	7051	8
1	1	6163	-1	Kuroshitsuji Recap	Special	1.0	7.32	20616	9
2	1	9581	-1	MM! Specials	Special	9.0	6.77	21462	9
3	1	11161	-1	Hoshizora e Kakaru Hashi: Kakaru ka? Gakuensai...	Special	1.0	7.02	17770	9
4	1	13561	-1	Guilty Crown: 4-koma Gekijou	Special	11.0	7.23	13053	9

5 rows × 56 columns

In [299]:
```
df_interacciones.shape
```

Out[299]: (775258, 56)

In [300]:
```
len(df_interacciones.user_id.unique())
```

Out[300]: 48303

In [301]:
```
len(df_interacciones.anime_id.unique())
```

Out[301]: 7492

In [ ]:
```
# Exportamos a .csv los df de interacciones final
df_interacciones.to_csv('interacciones_generos_VF.csv',sep=';')
```

In [302]:
```
anime_final = pd.DataFrame({"anime_id":df_interacciones["anime_id"]
.unique()})
```

In [303]:
```
df_titulos_VF = pd.merge(left=df_titulos, left_on='anime_id',
                         right=anime_final["anime_id"], right_on='anime_id')
df_titulos_VF.shape
```

Out[303]: (7492, 52)

In [ ]:
```
# Exportamos a .csv los df de titulos final
df_titulos_VF.to_csv('titulos_generos_VF.csv',sep=';')
```

Generamos la tabla final de usuarios versus animes

In [19]:
```
df_interacciones = pd.read_csv("interacciones_generos_VF.csv", sep=
";")
```

In [20]:
```
df_interacciones.columns
```

Out[20]:
```
Index(['Unnamed: 0', 'user_id', 'anime_id', 'rating', 'name', 'typ
e',
       'episodes', 'rating_global', 'members', 'log_members', 'Act
ion',
       'Adventure', 'Cars', 'Comedy', 'Dementia', 'Demons', 'Drama
', 'Ecchi',
       'Fantasy', 'Game', 'Harem', 'Hentai', 'Historical', 'Horror
', 'Josei',
       'Kids', 'Magic', 'MartialArts', 'Mecha', 'Military', 'Music
', 'Mystery',
       'Parody', 'Police', 'Psychological', 'Romance', 'Samurai',
'School',
       'Sci-Fi', 'Seinen', 'Shoujo', 'ShoujoAi', 'Shounen', 'Shoun
enAi',
       'SliceofLife', 'Space', 'Sports', 'Supernatural', 'SuperPow
er',
       'Thriller', 'Vampire', 'Yaoi', 'Yuri', 'afinidad', 'apriori
',
       'Frec_Anime', 'Frec_User'],
      dtype='object')
```

Indicador de Afinidad

In [304]:
```
df_inter_afin = df_interacciones
```

In [305]:
```
for j in subgenre:
    df_inter_afin[j] = df_inter_afin[j]*df_inter_afin["afinidad"]
```

In [306]:
```
tmp = pd.pivot_table(df_inter_afin,values=subgenre,index=["user_id"
],aggfunc=sum)
vs_afinidad_generos = tmp.fillna(0)
```

In [307]:
```
vs_afinidad_generos.head()
```

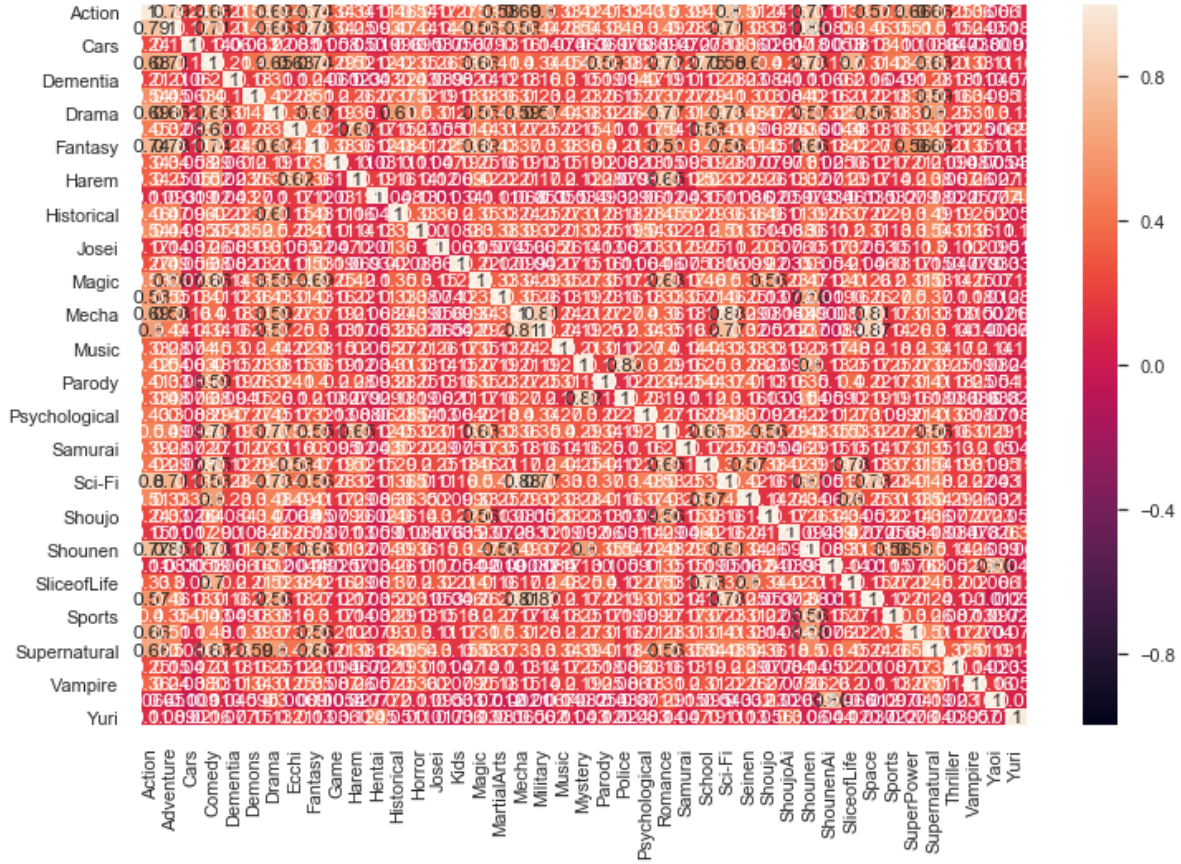
Out[307]:

user_id	Action	Adventure	Cars	Comedy	Dementia	Demons	Drama	Ecchi	Fantasy	G
1	9	0	0	18	0	3	3	12	6	
3	18	23	0	23	0	0	9	0	23	
4	3	0	0	6	0	0	0	0	6	
6	0	0	0	3	0	0	0	0	0	
7	103	69	0	204	0	17	38	67	81	

5 rows × 43 columns

In [ ]:
```
# Exportamos a .csv
vs_afinidad_generos.to_csv('vs_afinidad_generos_VF.csv',sep=';')
```

In [308]:
```
plt.subplots(figsize=(12, 8))
sns.heatmap(vs_afinidad_generos.corr(), annot=True, vmin=-1, vmax=1
)
plt.show()
```



Se observan correlaciones medias entre los 43 géneros a utilizar para los algoritmos de modelamiento (agrupación)

Indicador de Apriori

In [309]:
```
df_inter_apri = df_interacciones
```

In [310]:
```
df_inter_apri["apriori"].value_counts()
```

Out[310]:
```
1    705736
0     69522
Name: apriori, dtype: int64
```

In [311]:
```
for j in subgenre:
    df_inter_apri[j] = df_inter_apri[j]*df_inter_apri["apriori"]
```

In [312]:
```
tmp = pd.pivot_table(df_inter_apri,values=subgenre,index=["user_id"
],aggfunc=max)
vs_apriori_generos = tmp.fillna(0)
```

In [313]:
```
vs_apriori_generos.head()
```

Out[313]:

user_id	Action	Adventure	Cars	Comedy	Dementia	Demons	Drama	Ecchi	Fantasy	G
1	3.0	0.0	0.0	3.0	0.0	3.0	3.0	3.0	3.0	
3	5.0	5.0	0.0	5.0	0.0	0.0	5.0	0.0	5.0	
4	3.0	0.0	0.0	3.0	0.0	0.0	0.0	0.0	3.0	
6	0.0	0.0	0.0	3.0	0.0	0.0	0.0	0.0	0.0	
7	5.0	5.0	0.0	5.0	0.0	5.0	5.0	5.0	5.0	

5 rows × 43 columns

In [314]:
```
vs_apri_backup = vs_apriori_generos
```

In [315]:
```
for i in subgenre:
    vs_apriori_generos[i] = np.where(vs_apriori_generos[i]==0,None,
vs_apriori_generos[i])
```

In [316]:
```
vs_apriori_generos.head()
```

Out[316]:

user_id	Action	Adventure	Cars	Comedy	Dementia	Demons	Drama	Ecchi	Fantasy	G
1	3	None	None	3	None	3	3	3	3	I
3	5	5	None	5	None	None	5	None	5	I
4	3	None	None	3	None	None	None	None	3	I
6	None	None	None	3	None	None	None	None	None	I
7	5	5	None	5	None	5	5	5	5	

5 rows × 43 columns

In [317]:
```
for j in subgenre:
    vs_apriori_generos[j].replace(1,j,inplace=True)
vs_apriori_generos.head()
```

Out[317]:

user_id	Action	Adventure	Cars	Comedy	Dementia	Demons	Drama	Ecchi	Fantasy	G
1	3.0	NaN	NaN	3.0	NaN	3.0	3.0	3.0	3.0	
3	5.0	5.0	NaN	5.0	NaN	NaN	5.0	NaN	5.0	
4	3.0	NaN	NaN	3.0	NaN	NaN	NaN	NaN	3.0	
6	NaN	NaN	NaN	3.0	NaN	NaN	NaN	NaN	NaN	
7	5.0	5.0	NaN	5.0	NaN	5.0	5.0	5.0	5.0	

5 rows × 43 columns

In [326]:
```
def apriorizar(fila):
# Se construye esta función para entregar
# en un atributo, como una lista, todos
# los géneros relacionados a un usuario

    tmp = fila.values.tolist()
    res = list(filter(None, tmp))
    return res

vs_apriori_generos["aprioris"] = vs_apriori_generos[subgenre].apply
((apriorizar), axis=1)
```

In [319]:
```
vs_apriori_generos.head()
```

Out[319]:

user_id	Action	Adventure	Cars	Comedy	Dementia	Demons	Drama	Ecchi	Fantasy	G
1	3.0	NaN	NaN	3.0	NaN	3.0	3.0	3.0	3.0	
3	5.0	5.0	NaN	5.0	NaN	NaN	5.0	NaN	5.0	
4	3.0	NaN	NaN	3.0	NaN	NaN	NaN	NaN	3.0	
6	NaN	NaN	NaN	3.0	NaN	NaN	NaN	NaN	NaN	
7	5.0	5.0	NaN	5.0	NaN	5.0	5.0	5.0	5.0	

5 rows × 44 columns

In [48]:
```
# Exportamos a .csv
vs_apriori_generos.to_csv('vs_apriori_generos_VF_woNone.csv',sep=';
')
```

In [ ]:

In [49]:
```
df_titulos_VF = pd.read_csv("titulos_generos_VF20200518.csv", sep=";")
df_interacciones = pd.read_csv("interacciones_generos_VF.csv", sep=";")
vs_afinidad_generos = pd.read_csv('vs_afinidad_generos_VF.csv',sep=';')
```

In [ ]:

Se genera un data sets, basado en el indicador de afinidad anterior, pero en el cual se determinará más de un género afin por usuario para mejorar la predicción

In [320]:
```
vs_afinidad_5generos = vs_afinidad_generos
vs_afinidad_5generos.head()
```

Out[320]:

user_id	Action	Adventure	Cars	Comedy	Dementia	Demons	Drama	Ecchi	Fantasy	G
1	9	0	0	18	0	3	3	12	6	
3	18	23	0	23	0	0	9	0	23	
4	3	0	0	6	0	0	0	0	6	
6	0	0	0	3	0	0	0	0	0	
7	103	69	0	204	0	17	38	67	81	

5 rows × 43 columns

In [51]:
```
vs_afinidad_5generos ["Cant.Genre"] = vs_afinidad_5generos[subgenre].apply(np.count_nonzero, axis=1)
```

In [52]:
```
vs_afinidad_5generos ["Mean Afinidad"] = vs_afinidad_5generos[subgenre].apply(np.mean, axis=1)
```

In [53]:
```
def prom_ajustado(fila):
    prom_ajustado = np.sum(fila)/np.count_nonzero(fila)
    return prom_ajustado

vs_afinidad_5generos ["Mean2 Afinidad"] = vs_afinidad_5generos[subg
enre].apply((prom_ajustado), axis=1)
```

In [54]:
```
vs_afinidad_5generos.head()
```

Out[54]:

	user_id	Action	Adventure	Cars	Comedy	Dementia	Demons	Drama	Ecchi	Fantasy
0	1	9	0	0	18	0	3	3	12	6
1	3	18	23	0	23	0	0	9	0	23
2	4	3	0	0	6	0	0	0	0	6
3	6	0	0	0	3	0	0	0	0	0
4	7	103	69	0	204	0	17	38	67	81

5 rows × 47 columns

In [55]:
```
vs_afinidad_5generos ["Max1"] = vs_afinidad_5generos[subgenre].appl
y(np.amax, axis=1)
```

In [56]:
```
vs_afinidad_5generos ["Max1_G"] = vs_afinidad_5generos[subgenre].ap
ply(np.argmax, axis=1)
```

In [57]:
```
vs_afinidad_5generos.head()
```

Out[57]:

	user_id	Action	Adventure	Cars	Comedy	Dementia	Demons	Drama	Ecchi	Fantasy
0	1	9	0	0	18	0	3	3	12	6
1	3	18	23	0	23	0	0	9	0	23
2	4	3	0	0	6	0	0	0	0	6
3	6	0	0	0	3	0	0	0	0	0
4	7	103	69	0	204	0	17	38	67	81

5 rows × 49 columns

In [219]:
```
def maximos(fila):
    a = np.amax(fila)
    vec = np.where(fila.tolist() == a,1,0)
    tmp2 = fila.tolist()
    vec2 = np.zeros(len(tmp2))
    tmp2.remove(a)
    if len(tmp2) == len(subgenre)-1:
        b = np.amax(tmp2)
        vec2 = np.where(fila.tolist() == b,1,0)
    #vvf= [ j+k for (j, k) in zip(vec, vec2) ]
    maxs = []
    for n,i in enumerate(vec):
        if i==1 or vec2[n]==1:
            maxs.append(subgenre[n])
    return maxs

vs_afinidad_5generos ["Maximos"] = vs_afinidad_5generos[subgenre].a
pply((maximos), axis=1)
```

In [223]:
```
vs_afinidad_5generos.head()
```

Out[223]:

	user_id	Action	Adventure	Cars	Comedy	Dementia	Demons	Drama	Ecchi	Fantasy
0	1	9	0	0	18	0	3	3	12	6
1	3	18	23	0	23	0	0	9	0	23
2	4	3	0	0	6	0	0	0	0	6
3	6	0	0	0	3	0	0	0	0	0
4	7	103	69	0	204	0	17	38	67	81

5 rows × 50 columns

In [136]:
```
vs_afinidad_5generos["Max1_G"].value_counts()
```

Out[136]:
```
Comedy           18738
Action           10146
Adventure         3404
Drama             3374
Sci-Fi            2192
Romance           1982
Shounen           1636
Fantasy           1461
Hentai             874
Shoujo             584
School             478
Ecchi              434
Mecha              360
SliceofLife        329
Music              292
Supernatural       274
Mystery            211
Sports             193
Horror             183
Magic              174
Historical         126
Dementia           103
Military            98
Harem               97
Seinen              97
Game                88
Demons              72
Yaoi                66
Kids                48
ShoujoAi            44
Psychological       41
Cars                35
ShounenAi           34
MartialArts         13
SuperPower           7
Parody               5
Police               3
Josei                3
Thriller             2
Samurai              2
Name: Max1_G, dtype: int64
```

In [224]:
```
# Exportamos a .csv
vs_afinidad_5generos.to_csv('vs_afin_maxgen_VF.csv',sep=';')
```

Principales hallazgos

Existen 2 géneros que son ampliamente afines a los usuarios "Comedy" (18.738 usuarios) y "Action" (10.146 usuarios)

In [ ]:

Prueba - Fundamentos Data Science

Martes 19 de mayo de 2020, Santiago de Chile

Sakura SPA

Miembros de la Célula:

Susana Arce

Fabiola Aravena

Rodrigo Pereira

Administrador de Contrato: Gonzalo Seguel

Sponsor: Andrea Villaroel

Objetivo:

Nuestro sponsor posee un cátalogo de contenido audiovisual de 12.294 títulos de Anime, requiere poder efecturar recomendaciones de qué ver a los usuarios en base a otros Anime han sido de su gusto.

Propuesta:

Aplicación web en donde el usuario seleccione de una lista de Anime propuestos los que han sido de su gusto, y en base a ello se le entregue una lista de otras alternativas que sean a fin con sus preferencias.

Búsqueda de Posters asociados a los anime

Este registro, muestra la labor realizar para hacer una ingesta de la data asociada a url en donde se encuentran las portadas de cada título de anime, esta data adicional se ingesto para ser utilizada en la fase de producción de la plataforma.

In [1]:
```
# Importación de librerías para procesamiento de datos.
import pandas as pd
import numpy as np

import matplotlib.pyplot as plt
import seaborn as sns
plt.style.use('seaborn')

import factor_analyzer as factor
import missingno as msngo

import warnings
warnings.filterwarnings(action="ignore")
```

In [32]:
```
subgenre = ["Action","Adventure","Cars","Comedy","Dementia","Demons
","Drama","Ecchi","Fantasy","Game","Harem","Hentai","Historical","H
orror","Josei","Kids","Magic","MartialArts","Mecha","Military","Mus
ic","Mystery","Parody","Police","Psychological","Romance","Samurai"
,
"School","Sci-Fi","Seinen","Shoujo","ShoujoAi","Shounen","ShounenAi
","SliceofLife","Space","Sports","Supernatural","SuperPower","Thril
ler","Vampire","Yaoi","Yuri"]
```

Recogida de datos

Es importante indicar que existe un límite comercial de 5.000 consultas diarias a la API que nos proporciona los posters de anime

In [30]:
```
#Importamos los data sets de las ingestas anteriores realizadas
df_t1 = pd.read_csv("titulos_con_url_1.csv", sep=";")
df_t2 = pd.read_csv("titulos_con_url_2.csv", sep=";")
df_t3 = pd.read_csv("titulos_con_url_3.csv", sep=";")
df_t44 = pd.read_csv("titulos_con_url_44.csv", sep=";")
df_t45 = pd.read_csv("titulos_con_url_45.csv", sep=";")
df_t46 = pd.read_csv("titulos_con_url_46.csv", sep=";")
```

In [111]:
```
df_titulos_generos = pd.read_csv("titulos_generos_VF.csv", sep=";")
df_titulos_generos.head()
```

Out[111]:

	Unnamed: 0	index	anime_id	name	genre	type	episodes	rating	mem
0	0	5	32935	Haikyuu!!: Karasuno Koukou VS Shiratorizawa Ga...	Comedy, Drama, School, Shounen, Sports	TV	10.0	9.15	9
1	1	7	820	Ginga Eiyuu Densetsu	Drama, Military, Sci-Fi, Space	OVA	110.0	9.11	8
2	2	11	28851	Koe no Katachi	Drama, School, Shounen	Movie	1.0	9.05	10
3	3	33	28957	Mushishi Zoku Shou: Suzu no Shizuku	Adventure, Fantasy, Historical, Mystery, Seine...	Movie	1.0	8.75	3
4	4	37	31757	Kizumonogatari II: Nekketsu-hen	Action, Mystery, Supernatural, Vampire	Movie	1.0	8.73	3

5 rows × 53 columns

In [118]:
```
df_titulos_generos.shape
```

Out[118]: (7492, 54)

Procesamiento de Urls

Se analizan los datos de los registros de días anteriores

In [6]:
```
df_titulos_generos["url"] = None
```

In [112]:
```
df_t46["url"].isnull().value_counts()
```

Out[112]:
```
True     7805
False    4489
Name: url, dtype: int64
```

In [34]:
```
df_t46["new_url"]=None
```

In [52]:
```
df_t46["new_url"] = np.where( pd.isnull(df_t46["url"]),df_t1["url"]
,df_t46["url"])
df_t46["new_url"].isnull().value_counts()
```

Out[52]:
```
True     7805
False    4489
Name: new_url, dtype: int64
```

In [53]:
```
df_t46["new_url"] = np.where( pd.isnull(df_t46["new_url"]),df_t2["u
rl"],df_t46["new_url"])
df_t46["new_url"].isnull().value_counts()
```

Out[53]:
```
True     7805
False    4489
Name: new_url, dtype: int64
```

In [54]:
```
df_t46["new_url"] = np.where( pd.isnull(df_t46["new_url"]),df_t3["u
rl"],df_t46["new_url"])
df_t46["new_url"].isnull().value_counts()
```

Out[54]:
```
True     7805
False    4489
Name: new_url, dtype: int64
```

In [55]:
```
df_t46["new_url"] = np.where( pd.isnull(df_t46["new_url"]),df_t45["
url"],df_t46["new_url"])
df_t46["new_url"].isnull().value_counts()
```

Out[55]:
```
True     7805
False    4489
Name: new_url, dtype: int64
```

In [113]:
```
df_transfer = df_t46.loc[:,["anime_id","new_url"]]
df_transfer.head()
```

Out[113]:

	anime_id	new_url
0	32281	https://imdb-api.com/images/original/MV5BODRmZ...
1	5114	https://imdb-api.com/images/original/MV5BZmEzN...
2	28977	https://imdb-api.com/images/original/MV5BNzM4Y...
3	9253	https://imdb-api.com/images/original/MV5BYmJhM...
4	9969	https://imdb-api.com/images/original/MV5BNzM4Y...

In [114]:
```
df_transfer.shape
```

Out[114]:
```
(12294, 2)
```

In [115]:
```
# Se cargan los registros existentes al data set de df_titulos
df_titulos_generos = pd.merge(right=df_titulos_generos,left=df_tran
sfer)
df_titulos_generos.head()
```

Out[115]:

	anime_id	new_url	Unnamed: 0	index	name	
0	32935	https://imdb-api.com/images/original/MV5BNzQ1M...	0	5	Haikyuu!!: Karasuno Koukou VS Shiratorizawa Ga...	Cc D S Sh
1	820	NaN	1	7	Ginga Eiyuu Densetsu	D Militar Fi,
2	28851	https://imdb-api.com/images/original/MV5BZGRkO...	2	11	Koe no Katachi	D S Sh
3	28957	https://imdb-api.com/images/original/MV5BMjM5N...	3	33	Mushishi Zoku Shou: Suzu no Shizuku	Adve Fa Hist M S
4	31757	https://imdb-api.com/images/original/MV5BZjVjN...	4	37	Kizumonogatari II: Nekketsu-hen	A M Supern Va

5 rows × 54 columns

In [116]:
```
df_titulos_generos["new_url"].isnull().value_counts()
```

Out[116]:
```
True     4689
False    2803
Name: new_url, dtype: int64
```

In [117]:
```
df_titulos_generos.shape
```

Out[117]:
```
(7492, 54)
```

In [157]:
```
df_titulos = df_titulos_generos[df_titulos_generos["new_url"].isnul
l()].reset_index()
df_titulos.shape
```

Out[157]:
```
(4689, 55)
```

Ingestar imagen de los anime

In [10]:
```
#solo ejecutar una vez
#df_titulos["url"] = None
```

In [11]:
```
import json
import requests

result = 0 #solo ejecutar una vez --> cantidad de títulos con respu
esta no vacía
url_cant = 0 #solo ejecutar una vez --> cantidad de títulos respond
idos por la API
contador = 0 #solo ejecutar una vez -- cantidad de títulos consulta
dos
```

Ejecutar una celda por día

In [19]:
```
for name in df_titulos["name"]:
    year = 0
    namee = name.replace("/","%20")
    namee = name.replace(" ","%20")
    url = "https://imdb-api.com/ja/API/SearchSeries/k_h2JDsS1e/"+na
mee
    if list(df_titulos[df_titulos["name"] == name]["type"])[0] == "
Movie":
        url = "https://imdb-api.com/ja/API/SearchMovie/k_h2JDsS1e/"
+namee

    response = requests.request("GET", url)
 #   print(response.status_code)
    if response.status_code == 200:
        if json.loads(response.text)["results"] is not None:
            if json.loads(response.text)["results"] != []:
                result = result +1
                print(namee)
                print(url)
                for i in json.loads(response.text)["results"]:
                    df_titulos.loc[df_titulos["name"]==name,"url"]
= i["image"]
                    print(i["image"])
                    if  i["description"][1:5].isdigit():
                        yea = int(i["description"][1:5])
                        if yea > year:
                            year = yea
                            df_titulos.loc[df_titulos["name"]==name
,"url"] = i["image"]
#         print(df_titulos.loc[df_titulos["name"]==name,"url"])
#         print("-----------------------------------------------
----")
        url_cant = url_cant +1
    contador = contador +1
```

```
Bonobono%20(TV)
https://imdb-api.com/ja/API/SearchSeries/k_h2JDsS1e/Bonobono%20(TV
)
https://imdb-api.com/images/original/MV5BMDQzOTVhMDQtMTNkMi00MzE1L
Tg1ZjktYTYyY2U5OWZjZWMwXkEyXkFqcGdeQXVyNTY0NDkzNDc@._V1_Ratio0.727
3_AL_.jpg
FlashBack
https://imdb-api.com/ja/API/SearchSeries/k_h2JDsS1e/FlashBack
https://imdb-api.com/images/original/MV5BYjc4MzE2NzctNzQyMS00ZmRhL
ThhOTEtM2NkYjkzZTFjOWZiXkEyXkFqcGdeQXVyNjgzMjQ0MTA@._V1_Ratio0.727
3_AL_.jpg
Interlude
https://imdb-api.com/ja/API/SearchSeries/k_h2JDsS1e/Interlude
https://imdb-api.com/images/original/MV5BNjkyZGRjMmItOTY3YS00ZDc5L
WFiMGItMGI4Njk5ZTk3NTRlXkEyXkFqcGdeQXVyMTA1OTEwNjE@._V1_Ratio0.727
3_AL_.jpg
Hairy%20Tale
https://imdb-api.com/ja/API/SearchSeries/k_h2JDsS1e/Hairy%20Tale
https://imdb-api.com/images/original/MV5BMmE4NmE1MjctN2E5MC00MWMyL
WE0MTYtNjMyYjIwMmNhMGE3XkEyXkFqcGdeQXVyMTAwMzM3NDI3._V1_Ratio0.727
3_AL_.jpg
Piano
https://imdb-api.com/ja/API/SearchSeries/k_h2JDsS1e/Piano
https://imdb-api.com/images/original/MV5BNTBmZDFmNDctMWI3MS00MDE2L
ThiN2EtZDBjYzQxNGFiNWExXkEyXkFqcGdeQXVyNjU3MzA0NjE@._V1_Ratio0.727
3_AL_.jpg
Forsaken
https://imdb-api.com/ja/API/SearchSeries/k_h2JDsS1e/Forsaken
https://imdb-api.com/images/original/MV5BMDJhNDQ1MjktM2UzZi00M2M4L
WIzNDAtZDQwY2Q1NDllNmM0XkEyXkFqcGdeQXVyMjY0MTQ0NjY@._V1_Ratio0.727
3_AL_.jpg
Hashire!
https://imdb-api.com/ja/API/SearchSeries/k_h2JDsS1e/Hashire!
https://imdb-api.com/images/original/nopicture.jpg
Ijoku
https://imdb-api.com/ja/API/SearchSeries/k_h2JDsS1e/Ijoku
https://imdb-api.com/images/original/nopicture.jpg
Shusaku
https://imdb-api.com/ja/API/SearchSeries/k_h2JDsS1e/Shusaku
https://imdb-api.com/images/original/nopicture.jpg
Pendant
https://imdb-api.com/ja/API/SearchSeries/k_h2JDsS1e/Pendant
https://imdb-api.com/images/original/nopicture.jpg
```

In [20]:
```
print("Cant. de consultas ejecutadas:")
print(contador )
print("Cant. de títulos con conexión establecida por la API:")
print(url_cant)
print("Cant. de títulos con resultados no vacíos")
print(result)
print("Cant. último título consultado:")
print(name)
print("Ubicación último título consultado")
print(np.where(df_titulos["name"]==name))
```

```
Cant. de consultas ejecutadas:
4917
Cant. de títulos con conexión establecida por la API:
4850
Cant. de títulos con resultados no vacíos
11
Cant. último título consultado:
Sakura no Mori
Ubicación último título consultado
(array([4557]),)
```

In [21]:
```
df_titulos["url"].value_counts()
```

Out[21]:
```
https://imdb-api.com/images/original/nopicture.jpg
4
https://imdb-api.com/images/original/MV5BNTBmZDFmNDctMWI3MS00MDE2L
ThiN2EtZDBjYzQxNGFiNWExXkEyXkFqcGdeQXVyNjU3MzA0NjE@._V1_Ratio0.727
3_AL_.jpg    1
https://imdb-api.com/images/original/MV5BMmE4NmE1MjctN2E5MC00MWMyL
WE0MTYtNjMyYjIwMmNhMGE3XkEyXkFqcGdeQXVyMTAwMzM3NDI3._V1_Ratio0.727
3_AL_.jpg    1
https://imdb-api.com/images/original/MV5BMDJhNDQ1MjktM2UzZi00M2M4L
WIzNDAtZDQwY2Q1NDllNmM0XkEyXkFqcGdeQXVyMjY0MTQ0NjY@._V1_Ratio0.727
3_AL_.jpg    1
https://imdb-api.com/images/original/MV5BYjc4MzE2NzctNzQyMS00ZmRhL
ThhOTEtM2NkYjkzZTFjOWZiXkEyXkFqcGdeQXVyNjgzMjQ0MTA@._V1_Ratio0.727
3_AL_.jpg    1
https://imdb-api.com/images/original/MV5BMDQzOTVhMDQtMTNkMi00MzE1L
Tg1ZjktYTYyY2U5OWZjZWMwXkEyXkFqcGdeQXVyNTY0NDkzNDc@._V1_Ratio0.727
3_AL_.jpg    1
https://imdb-api.com/images/original/MV5BNjkyZGRjMmItOTY3YS00ZDc5L
WFiMGItMGI4Njk5ZTk3NTRlXkEyXkFqcGdeQXVyMTA1OTEwNjE@._V1_Ratio0.727
3_AL_.jpg    1
Name: url, dtype: int64
```

In [22]:
```
df_titulos["url"].isnull().value_counts()
```

Out[22]:
```
True     4548
False      10
Name: url, dtype: int64
```

La ingesta se debe realizar parcial por limitantes comerciales del API

In [175]:
```
# Escribir DataFrame a CSV por
df_titulos.to_csv('titulos_con_url_7.csv',sep=';')
```

In [27]:
```
df_titulos_fusion = pd.merge(right=df_titulos_generos, right_on="anime_id",
                             left=df_titulos.loc[:,["anime_id","url"]], left_on="anime_id",
                             how="right")
print(df_titulos_fusion.shape)
df_titulos_fusion.head()
```

```
(7492, 54)
```

Out[27]:

	anime_id	url_x	Unnamed: 0	url_y	name	genre	type	episodes	rating
0	820	None	0	NaN	Ginga Eiyuu Densetsu	Drama, Military, Sci-Fi, Space	OVA	110.0	9.11
1	30709	None	1	NaN	Kamisama Hajimemashita: Kako-hen	Comedy, Demons, Fantasy, Shoujo, Supernatural	OVA	4.0	8.64
2	12431	None	2	NaN	Uchuu Kyoudai	Comedy, Sci-Fi, Seinen, Slice of Life, Space	TV	99.0	8.59
3	17389	None	3	NaN	Kingdom 2nd Season	Action, Historical, Military, Seinen	TV	39.0	8.57
4	24687	None	4	NaN	Mushishi Zoku Shou: Odoro no Michi	Adventure, Fantasy, Historical, Mystery, Seine...	Special	1.0	8.54

5 rows × 54 columns

In [28]:
```
df_titulos_fusion["url"] = np.where( pd.isnull(df_titulos_fusion["u
rl_y"]),
                                    df_titulos_fusion["url_x"],
                                    df_titulos_fusion["url_y"])
df_titulos_fusion["url"].isnull().value_counts()
```

Out[28]:
```
True     4548
False    2944
Name: url, dtype: int64
```

In [29]:
```
df_titulos_fusion.columns
```

Out[29]:
```
Index(['anime_id', 'url_x', 'Unnamed: 0', 'url_y', 'name', 'genre'
, 'type',
       'episodes', 'rating', 'members', 'log_members', 'Action', '
Adventure',
       'Cars', 'Comedy', 'Dementia', 'Demons', 'Drama', 'Ecchi', '
Fantasy',
       'Game', 'Harem', 'Hentai', 'Historical', 'Horror', 'Josei',
'Kids',
       'Magic', 'MartialArts', 'Mecha', 'Military', 'Music', 'Myst
ery',
       'Parody', 'Police', 'Psychological', 'Romance', 'Samurai',
'School',
       'Sci-Fi', 'Seinen', 'Shoujo', 'ShoujoAi', 'Shounen', 'Shoun
enAi',
       'SliceofLife', 'Space', 'Sports', 'Supernatural', 'SuperPow
er',
       'Thriller', 'Vampire', 'Yaoi', 'Yuri', 'url'],
      dtype='object')
```

In [30]:
```
df_titulos_fusion.drop(['url_x', 'Unnamed: 0', 'url_y'],axis=1,inpl
ace=True)
```

In [31]:
```
# Exportamos a .csv los df de interacciones final
df_titulos_fusion.to_csv('titulos_generos_VF20200519.csv',sep=';')
```

Principales hallazgos:

Se encuentran posters para 2.944 animes, sin embargo 438 corresponde a imagenes vacías. Se observa que en todos los generos existen animes con posters encontrado positivamente.

In [35]:
```
df_titulos_fusion["url"].isnull().value_counts()
```

Out[35]:
```
True     4548
False    2944
Name: url, dtype: int64
```

In [36]:
```
for i in subgenre:
    print (i)
    print(df_titulos_fusion[df_titulos_fusion[i]==1]["url"].isnull(
).value_counts())
```

```
Action
True     1006
False     819
Name: url, dtype: int64
Adventure
True     856
False    706
Name: url, dtype: int64
Cars
False    29
True     15
Name: url, dtype: int64
Comedy
True     1696
False    1070
Name: url, dtype: int64
Dementia
False    79
True     51
Name: url, dtype: int64
Demons
True     137
False     59
Name: url, dtype: int64
Drama
True     599
False    585
Name: url, dtype: int64
Ecchi
True     243
False    153
Name: url, dtype: int64
Fantasy
True     848
False    524
Name: url, dtype: int64
Game
True     68
False    44
Name: url, dtype: int64
Harem
True     99
False    58
Name: url, dtype: int64
Hentai
True     829
False    174
Name: url, dtype: int64
```

```
Historical
True     237
False    211
Name: url, dtype: int64
Horror
True     128
False    110
Name: url, dtype: int64
Josei
True     23
False    11
Name: url, dtype: int64
Kids
True     389
False    222
Name: url, dtype: int64
Magic
True     318
False    198
Name: url, dtype: int64
MartialArts
True     105
False     64
Name: url, dtype: int64
Mecha
True     386
False    330
Name: url, dtype: int64
Military
True     178
False    135
Name: url, dtype: int64
Music
True     270
False    191
Name: url, dtype: int64
Mystery
True     166
False    117
Name: url, dtype: int64
Parody
True     170
False     78
Name: url, dtype: int64
Police
False    74
True     71
Name: url, dtype: int64
Psychological
False    67
True     47
Name: url, dtype: int64
Romance
True     465
```

```
False    405
Name: url, dtype: int64
Samurai
True     57
False    34
Name: url, dtype: int64
School
True     410
False    287
Name: url, dtype: int64
Sci-Fi
True     801
False    676
Name: url, dtype: int64
Seinen
True     172
False    139
Name: url, dtype: int64
Shoujo
True     240
False    173
Name: url, dtype: int64
ShoujoAi
True     19
False    19
Name: url, dtype: int64
Shounen
True     670
False    473
Name: url, dtype: int64
ShounenAi
True     29
False    18
Name: url, dtype: int64
SliceofLife
True     354
False    328
Name: url, dtype: int64
Space
True     179
False    104
Name: url, dtype: int64
Sports
False    172
True     168
Name: url, dtype: int64
Supernatural
True     331
False    226
Name: url, dtype: int64
SuperPower
True     167
False     93
Name: url, dtype: int64
```

```
Thriller
True     20
False    14
Name: url, dtype: int64
Vampire
True     33
False    24
Name: url, dtype: int64
Yaoi
True     18
False    10
Name: url, dtype: int64
Yuri
True     28
False    10
Name: url, dtype: int64
```

In [34]:
```
df_titulos_fusion["url"].value_counts()
```

Out[34]:
```
https://imdb-api.com/images/original/nopicture.jpg
438
https://imdb-api.com/images/original/MV5BMGZhOTQ1YmEtODI4NC00MjdkL
WFmNjktODIxMzk3OGUwMTk0XkEyXkFqcGdeQXVyMTkxNjUyNQ@@._V1_Ratio0.727
3_AL_.jpg                       10
https://imdb-api.com/images/original/MV5BZmJmYTIwMWUtMWI4ZC00NjA1L
Tk4OTMtZDgyZGE1ZjVlMWVjL2ltYWdlL2ltYWdlXkEyXkFqcGdeQXVyMjc4OTQ1OTA
@._V1_Ratio0.7273_AL_.jpg       8
https://imdb-api.com/images/original/MV5BNWFkNjQ1ZjAtZDMyZC00NWI3L
WE2ZDctZWM2MmUxMTE5MzcwXkEyXkFqcGdeQXVyMTk2MDc1MjQ@._V1_Ratio0.727
3_AL_.jpg                       6
https://imdb-api.com/images/original/MV5BNTk1OWE3MmItNDhlYi00NGM4L
TkwMWUtM2NkZWJiNjQyNGIxXkEyXkFqcGdeQXVyMTMxODk2OTU@._V1_Ratio0.727
3_AL_.jpg                       6
https://imdb-api.com/images/original/MV5BMTRmYWMzMTMtNTY2MC00NmEzL
ThiYWEtYThmZDNhNzgwNGEzXkEyXkFqcGdeQXVyMjc4OTQ1OTA@._V1_Ratio0.727
3_AL_.jpg                       6
https://imdb-api.com/images/original/MV5BZmFhMTQxZWUtNGVlZC00ZjFjL
WI2NjQtMTAwN2M3NzhmNTdiXkEyXkFqcGdeQXVyNjg3MDM4Mzc@._V1_Ratio0.727
3_AL_.jpg                       5
https://imdb-api.com/images/original/MV5BY2MwZWNkZTgtYTQ4Ny00NjgzL
WFjNjEtYjNhMjI2MzVjZWZjXkEyXkFqcGdeQXVyNjQ3MjI4NTk@._V1_Ratio0.772
7_AL_.jpg                       5
https://imdb-api.com/images/original/MV5BY2JlMThkZWEtNzdlYS00ZjVlL
WE2ZWYtZGU2MmIwMWFjMzBkXkEyXkFqcGdeQXVyNjgwNTk4Mg@@._V1_Ratio0.818
2_AL_.jpg                       5
https://imdb-api.com/images/original/MV5BN2MwNjQ0NzAtODVlYi00NDU5L
WFiZDQtMzMwOTBjOTZkODYxXkEyXkFqcGdeQXVyMTA3OTEyODI1._V1_Ratio0.727
3_AL_.jpg                       5
https://imdb-api.com/images/original/MV5BNjU1YjM2YzAtZWE2Ny00ZWNiL
WFkZWItMDJhMzJiNDQwMmI4XkEyXkFqcGdeQXVyNTU1MjgyMjk@._V1_Ratio0.727
3_AL_.jpg                       4
https://imdb-api.com/images/original/MV5BY2MxMmRmN2QtY2ZmZS00YjMyL
Tk1MTQtMmQ2ZDcwNTI0YzU3XkEyXkFqcGdeQXVyNDg0NDIxMTU@._V1_Ratio0.727
```

```
3_AL_.jpg                          4
https://imdb-api.com/images/original/MV5BNWZjNWE1MDAtZGZkNS00ZjY2L
WFhMWUtMWRlZjliMTQ0ZGM4XkEyXkFqcGdeQXVyOTgwMzk1MTA@._V1_Ratio0.727
3_AL_.jpg                          4
https://imdb-api.com/images/original/MV5BYjA3MWI5YzctMzc0ZC00YmMyL
WFiMGEtZWNhODgzOTdkNTJkXkEyXkFqcGdeQXVyMTEwNTA2NjEy._V1_Ratio1.181
8_AL_.jpg                          4
https://imdb-api.com/images/original/MV5BNzgwNzI3MTE3Ml5BMl5BanBnX
kFtZTcwMjc4NzIzMQ@@._V1_Ratio0.7273_AL_.jpg
4
https://imdb-api.com/images/original/MV5BMTcwMDEwMjI2OV5BMl5BanBnX
kFtZTcwMzkxMjE1MQ@@._V1_Ratio0.7273_AL_.jpg
4
https://imdb-api.com/images/original/MV5BOGI2MmUyODgtOWQ5NS00OWY2L
WE3NzctNzAyN2Y3ZjUxMzI0XkEyXkFqcGdeQXVyNjExODE1MDc@._V1_Ratio0.727
3_AL_.jpg                          4
https://imdb-api.com/images/original/MV5BNmU3MDRkN2MtN2VmYS00NDhmL
TgxYWEtNjVkNDExNGQ5NmRkXkEyXkFqcGdeQXVyMTE3NTc4NDk@._V1_Ratio0.727
3_AL_.jpg                          3
https://imdb-api.com/images/original/MV5BMTYwNjM3MDk0Nl5BMl5BanBnX
kFtZTcwNDc5NzIzMQ@@._V1_Ratio0.7273_AL_.jpg
3
https://imdb-api.com/images/original/MV5BNGJmMjVlZWYtZjZmOS00NWUyL
WJlZGQtYzE2MTc5NzRkNzgwXkEyXkFqcGdeQXVyNjExODE1MDc@._V1_Ratio0.727
3_AL_.jpg                          3
https://imdb-api.com/images/original/MV5BYmIyMGIyN2ItMWY0My00OTI2L
TkzODctNDU4YjdhYmY1ZWUyXkEyXkFqcGdeQXVyNjg5NDY4MDY@._V1_Ratio0.863
6_AL_.jpg                          3
https://imdb-api.com/images/original/MV5BNWIwODg4N2QtNzQyYy00YWYwL
WI5MGQtYzk3ZjY1NTNhZmMyXkEyXkFqcGdeQXVyMjc4OTQ1OTA@._V1_Ratio0.680
0_AL_.jpg                          3
https://imdb-api.com/images/original/MV5BZDRkZDYzMDQtMjJkMi00ZTE4L
TljZmMtY2E1NTIzNWQzNjkyXkEyXkFqcGdeQXVyNjExODE1MDc@._V1_Ratio0.727
3_AL_.jpg                          3
https://imdb-api.com/images/original/MV5BMmQ2YjYxMjItM2U3MS00NWEzL
TgzYmMtOGZlZjBlM2ZjYzE1XkEyXkFqcGdeQXVyOTY0MDUzMDg@._V1_Ratio1.772
7_AL_.jpg                          3
https://imdb-api.com/images/original/MV5BYzQ1YjM5YWQtNzJiMi00YjhlL
WE2YzItMzhhZTQ5N2QxNDZhXkEyXkFqcGdeQXVyMjY0MzgwMTc@._V1_Ratio0.727
3_AL_.jpg                          3
https://imdb-api.com/images/original/MV5BNWQ5NzEyNzUtYmFjMC00ZGViL
WIwZjEtMzcwNWIzOTNlZTA3XkEyXkFqcGdeQXVyNTc3MDU1MTU@._V1_Ratio0.727
3_AL_.jpg                          3
https://imdb-api.com/images/original/MV5BYzRkMDNiM2YtZDNiYS00MzgzL
ThkNGMtZDAwZWRlMTViZDI4XkEyXkFqcGdeQXVyOTA3MTMyOTk@._V1_Ratio0.727
3_AL_.jpg                          3
https://imdb-api.com/images/original/MV5BZTQ4NDI5OTMtNTdiMS00NDRkL
TljMjktYTM1MGFhNTc4NjU3L2ltYWdlL2ltYWdlXkEyXkFqcGdeQXVyMzQ2MTY3MDQ
@._V1_Ratio0.7273_AL_.jpg      3
https://imdb-api.com/images/original/MV5BYTk3MGM5ZWMtMDMxNi00ZTdlL
ThhOWMtNTczMmMyNGQ5YzI2XkEyXkFqcGdeQXVyMTkxOTE4Mzc@._V1_Ratio0.727
3_AL_.jpg                          3
https://imdb-api.com/images/original/MV5BOWM4ZDAxMzgtZjAyNi00MmEyL
WEwYTgtNjE4MWMwOTFhOTIwXkEyXkFqcGdeQXVyMjc4OTQ1OTA@._V1_Ratio0.727
```

```
3_AL_.jpg                       3

...
https://imdb-api.com/images/original/MV5BNjI5YzRlNWMtODFkMS00NjIxL
WExNjctNGMzODY2NGI3M2Y4XkEyXkFqcGdeQXVyMjc4OTQ1OTA@._V1_Ratio0.727
3_AL_.jpg                       1
https://imdb-api.com/images/original/MV5BMTY4NDExMzI0NV5BMl5BanBnX
kFtZTgwNzA3ODg2NDE@._V1_Ratio0.7273_AL_.jpg
1
https://imdb-api.com/images/original/MV5BOTZlNzZmNzctYWNiMi00NTUxL
Tk1NDMtZGRkZGExNzMwZTRmXkEyXkFqcGdeQXVyNDg4MjkzNDk@._V1_Ratio1.772
7_AL_.jpg                       1
https://imdb-api.com/images/original/MV5BODk5N2NiMWUtNGExZC00MmVhL
TkzYmQtOGM1NmMwYWMzODllXkEyXkFqcGdeQXVyMjI5MjU5OTI@._V1_Ratio0.727
3_AL_.jpg                       1
https://imdb-api.com/images/original/MV5BNTVlYzRkZTEtMTZiMi00YWM1L
WI3OGUtYWQzMWRmMWMxMWRlXkEyXkFqcGdeQXVyNjUwMTQ4NjE@._V1_Ratio0.727
3_AL_.jpg                       1
https://imdb-api.com/images/original/MV5BNGE4YzhlYjgtNzJkMy00ZDQyL
WJmOTMtNGVhZjcwYTE1MjdiXkEyXkFqcGdeQXVyMzgxODM4NjM@._V1_Ratio0.727
3_AL_.jpg                       1
https://imdb-api.com/images/original/MV5BOGRlNTllMmYtNTcyYy00MGJjL
WFiY2UtMDE2NWY3ZmRkNjhmXkEyXkFqcGdeQXVyMzU3MDU3NjI@._V1_Ratio0.727
3_AL_.jpg                       1
https://imdb-api.com/images/original/MV5BNzUxZWY2ZjMtYTIzNy00Y2NiL
ThlZWUtOWE5MmFlNjNlMGVlXkEyXkFqcGdeQXVyNzYxMTExMzI@._V1_Ratio0.727
3_AL_.jpg                       1
https://imdb-api.com/images/original/MV5BMjAwZjEwODUtYTc0ZC00YTA3L
WE4ODktYzNmN2FiYTM2MTNlXkEyXkFqcGdeQXVyMzUwMDU2MjM@._V1_Ratio0.727
3_AL_.jpg                       1
https://imdb-api.com/images/original/MV5BNTQyZTBkMTgtMWMzMy00NTExL
WE2NGEtMDBlMDBmMzEyNWI3XkEyXkFqcGdeQXVyMTA3OTEyODI1._V1_Ratio0.727
3_AL_.jpg                       1
https://imdb-api.com/images/original/MV5BZmIxNTQ4ZDgtMTI1NC00YjZlL
Tg4OWMtOGI0OWI3Zjg3Y2UxXkEyXkFqcGdeQXVyMTA1OTEwNjE@._V1_Ratio1.000
0_AL_.jpg                       1
https://imdb-api.com/images/original/MV5BMGEyMDFkNWYtMTUxOS00ODFlL
WFmZjAtNmY0YzI2YjIyYWUxXkEyXkFqcGdeQXVyNTYxNjI1OTY@._V1_Ratio2.045
5_AL_.jpg                       1
https://imdb-api.com/images/original/MV5BMTE0M2M0ZmItMjEwMy00YmUwL
Tk4ZjUtZjk0ZTRmOTQxMzdhXkEyXkFqcGdeQXVyMTA3OTEyODI1._V1_Ratio0.727
3_AL_.jpg                       1
https://imdb-api.com/images/original/MV5BZGI0ZjMwMzAtYjRmNi00YWIyL
TkyNmQtNmFjZjYxNmIwMDVhL2ltYWdlXkEyXkFqcGdeQXVyNTIyOTIyMzQ@._V1_Ra
tio0.7273_AL_.jpg               1
https://imdb-api.com/images/original/MV5BYzEwZjUwZDEtN2Y0YS00ZWMwL
WIwMTUtOTFmNDU1ZTQwYzYyXkEyXkFqcGdeQXVyNjc3NzUwNTg@._V1_Ratio0.818
2_AL_.jpg                       1
https://imdb-api.com/images/original/MV5BMGU2MTI2MzktYjRiMS00ODhlL
Tk0YzUtNWUzN2RiMTFhMjVkXkEyXkFqcGdeQXVyMTEzOTk4Mzkw._V1_Ratio0.727
3_AL_.jpg                       1
https://imdb-api.com/images/original/MV5BMTU2MDg0Njk4MF5BMl5BanBnX
kFtZTgwMTY0OTIyMTE@._V1_Ratio0.8182_AL_.jpg
1
```

```
https://imdb-api.com/images/original/MV5BMjMwMTY5NGYtNmVkNy00ZjgzL
WE2YzUtNjBlZjlmMGJlYTFlXkEyXkFqcGdeQXVyNjI4NDUxOTc@._V1_Ratio0.727
3_AL_.jpg                       1
https://imdb-api.com/images/original/MV5BMzhiOThlN2ItYWI2Ny00NTBjL
TkzNjUtNGJkZmJhNmU4OTRhXkEyXkFqcGdeQXVyNTAyODkwOQ@@._V1_Ratio0.727
3_AL_.jpg                       1
https://imdb-api.com/images/original/MV5BMjAzMzY4ODcwOV5BMl5BanBnX
kFtZTgwMzkwMjY0NDE@._V1_Ratio1.3182_AL_.jpg
1
https://imdb-api.com/images/original/MV5BMTNlZWU1YjEtM2FkMS00ODNhL
WFhMDMtM2E0NWFkNGI0MDI1XkEyXkFqcGdeQXVyMjQ2MTk1OTE@._V1_Ratio0.727
3_AL_.jpg                       1
https://imdb-api.com/images/original/MV5BNjg5ZDQyOTItNjdlZi00YzFjL
WI1NmEtOWYzMGZmYmI4ODVjXkEyXkFqcGdeQXVyNjQyMjcwNDM@._V1_Ratio0.727
3_AL_.jpg                       1
https://imdb-api.com/images/original/MV5BYTZmMjM4NDQtZmRhNi00ODIzL
Tk4ZDAtNDRlY2U4YzI1NGZkXkEyXkFqcGdeQXVyNjY1MzM5Nzg@._V1_Ratio1.772
7_AL_.jpg                       1
https://imdb-api.com/images/original/MV5BMjQ3M2VjZWItNmIwYS00NjE5L
Tg0NTYtZmM3ZmFmNWIzMjQ1XkEyXkFqcGdeQXVyMzM4MjM0Nzg@._V1_Ratio0.727
3_AL_.jpg                       1
https://imdb-api.com/images/original/MV5BMzhkNzc4YmUtZWM1OC00YmJhL
WFhNGItNGQ3ZDY5M2RlMjY2XkEyXkFqcGdeQXVyNTM3MDMyMDQ@._V1_Ratio0.727
3_AL_.jpg                       1
https://imdb-api.com/images/original/MV5BYjg0N2QwZDAtMTIyZC00ODU4L
TllMmQtMTU3Mzc4ZWMwOTZlXkEyXkFqcGdeQXVyNTY2MzQ3MDE@._V1_Ratio0.727
3_AL_.jpg                       1
https://imdb-api.com/images/original/MV5BMzJjZWRhNDItNGY4ZS00NWY1L
TllZTgtZTE2NDY1NWU1ODIxXkEyXkFqcGdeQXVyMzkzODQwMA@@._V1_Ratio1.318
2_AL_.jpg                       1
https://imdb-api.com/images/original/MV5BOWU3ZjhiNGMtYzM1YS00MWQzL
ThmY2UtYjAxYTllM2U3Yjk5XkEyXkFqcGdeQXVyMjMxMDM2NjY@._V1_Ratio0.727
3_AL_.jpg                       1
https://imdb-api.com/images/original/MV5BMTM1ZjMyZTYtZGRlOS00ZjkzL
WExN2QtZDQ4YmRmODQxZmEyXkEyXkFqcGdeQXVyNjU0NTI0Nw@@._V1_Ratio0.727
3_AL_.jpg                       1
https://imdb-api.com/images/original/MV5BNjkwODUzMzY0NF5BMl5BanBnX
kFtZTcwMDE5NjAyMQ@@._V1_Ratio0.7273_AL_.jpg
1
Name: url, Length: 2202, dtype: int64
```

Prueba - Fundamentos Data Science

Martes 19 de mayo de 2020, Santiago de Chile

Sakura SPA

Miembros de la Célula:

Susana Arce

Fabiola Aravena

Rodrigo Pereira

Administrador de Contrato: Gonzalo Seguel

Sponsor: Andrea Villaroel

Objetivo:

Nuestro sponsor posee un cátalogo de contenido audiovisual de 12.294 títulos de Anime, requiere poder efecturar recomendaciones de qué ver a los usuarios en base a otros Anime han sido de su gusto.

Propuesta:

Aplicación web en donde el usuario seleccione de una lista de Anime propuestos los que han sido de su gusto, y en base a ello se le entregue una lista de otras alternativas que sean a fin con sus preferencias.

Modelamiento Predictivo.

En un principio, tomamos la estrategia de clasificar los usuarios por títulos, separados por los diferentes tipos de series o películas. Teníamos 14 mini set de datos por los que se corrieron modelos de K-means. Tuvimos el problema que para algunas segmentaciones, como movies, no llegábamos a un valor óptimo de clusters, por lo que la estrategia no funcionaba bien para clasificar todas nuestras sub categorías. Para el modelamiento de apriori, tuvimos problemas de memoria para su ejecución. Probamos en diferentes máquinas, locales y remotas, pero no tuvimos éxito. Nuestra teoría es que era tal la cantidad de ítems a clasificar que necesitaba muchos recursos.

Por este motivo, decidimos clasificar a los usuarios por géneros, para realizar una recomendación de títulos.

In [315]:
```
# Importación de lMibrerías para procesamiento de datos.
import pandas as pd
import numpy as np
from sklearn.cluster import KMeans
from apyori import apriori

import matplotlib.pyplot as plt
import seaborn as sns
plt.style.use('seaborn')


import random
import pickle

from sklearn.metrics import silhouette_samples, silhouette_score
from sklearn.metrics import pairwise_distances_argmin_min

%matplotlib inline
from mpl_toolkits.mplot3d import Axes3D
plt.rcParams['figure.figsize'] = (6, 4)
plt.style.use('ggplot')

from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

import warnings
warnings.filterwarnings(action="ignore")
```

Función para mostrar resultados de algoritmo apriori.

In [112]:
```
def inspect(results):
    rh          = [tuple(result[2][0][0]) for result in results]
    lh          = [tuple(result[2][0][1]) for result in results]
    supports    = [result[1] for result in results]
    confidences = [result[2][0][2] for result in results]
    lifts       = [result[2][0][3] for result in results]
    return list(zip(rh, lh, supports, confidences, lifts))
```

In [113]:
```
rs=123
```

Importamos el dataset que está conformado por usuarios y géneros transpuestos. Se realizó un puntaje de afinidad para cada usuario con cada género. Son los valores que se observan a continuación.

In [145]:
```
df_generos= pd.read_csv("vs_afinidad_generos_VF.csv", sep=";")
```

In [146]:
```
df_generos.head()
```

Out[146]:

	user_id	Action	Adventure	Cars	Comedy	Dementia	Demons	Drama	Ecchi	Fantasy	...
0	1	9	0	0	18	0	3	3	12	6	...
1	3	18	23	0	23	0	0	9	0	23	...
2	4	3	0	0	6	0	0	0	0	6	...
3	6	0	0	0	3	0	0	0	0	0	...
4	7	103	69	0	204	0	17	38	67	81	...

5 rows × 44 columns

Cambiando valores NaN por 0, para utilizar como entrada de el algoritmo K-Means.

In [149]:
```
df_generos = df_generos.fillna(0.0)
```

Pasando valores a una matriz

In [154]:
```
user = df_genero_col
user = user.values
X = df_generos
X = X.values
```

In [155]:
```
X
```

Out[155]:
```
array([[    1,     9,     0, ...,     0,     0,     0],
       [    3,    18,    23, ...,     0,     0,     0],
       [    4,     3,     0, ...,     0,     0,     0],
       ...,
       [73511,     0,     4, ...,     0,     0,     0],
       [73513,     0,     0, ...,     0,     0,     0],
       [73515,    91,    43, ...,     0,     0,     5]], dtype=int64)
```

Dividiendo el data set en un 70 % de entrenamiento y 30% de validación.

In [181]:
```
train_X, test_X = train_test_split(X, test_size=0.3, random_state = rs)
```

In [182]:
```
len(train_X)
```

Out[182]:
```
33812
```

In [183]:
```
len(test_X)
```

Out[183]:
```
14491
```

Eliminando user_id de la matriz, para dejar solo los géneros.

In [185]:
```
test_X = test_X[:, 1:44]
```

In [186]:
```
test_X
```

Out[186]:
```
array([[36, 48,  0, ...,  0,  0,  3],
       [12, 16,  0, ...,  0,  0,  0],
       [ 0,  8,  0, ...,  0,  0,  0],
       ...,
       [24, 25,  0, ...,  8,  0,  0],
       [60, 58,  0, ...,  5,  0,  0],
       [ 0, 15,  0, ...,  0,  0,  0]], dtype=int64)
```

In [188]:
```
train_X
```

Out[188]:
```
array([[59199,    10,     5, ...,     0,     0,     0],
       [ 9104,    14,     5, ...,     0,     0,     0],
       [33387,     0,     0, ...,     0,     0,     0],
       ...,
       [26846,     5,     0, ...,     0,     0,     0],
       [42754,    22,    36, ...,     0,     0,     0],
       [23887,   210,   166, ...,     0,     0,     0]], dtype=int64)
```

In [189]:
```
train_X = train_X[:, 1:44]
```

In [190]:
```
train_X
```

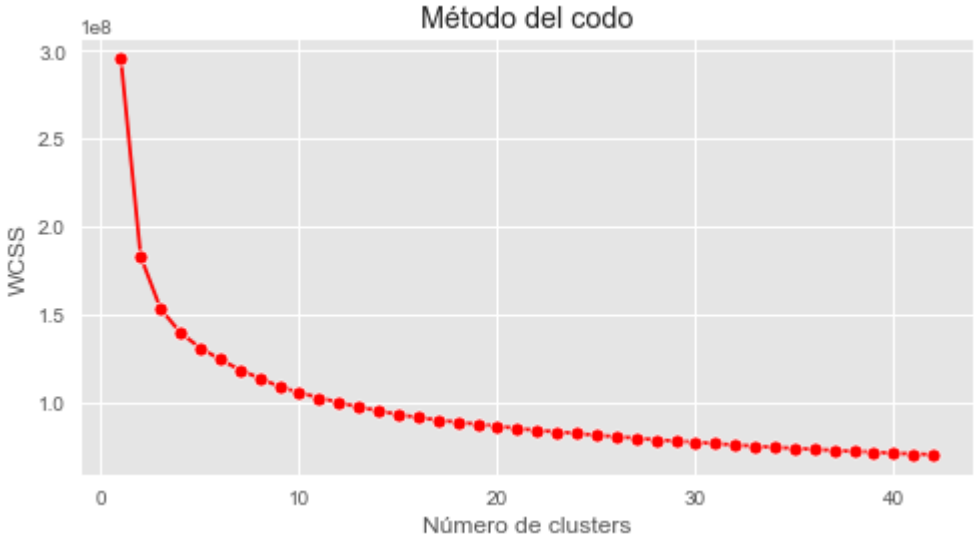
Out[190]:
```
array([[ 10,   5,   0, ...,   0,   0,   0],
       [ 14,   5,   0, ...,   0,   0,   0],
       [  0,   0,   0, ...,   0,   0,   0],
       ...,
       [  5,   0,   0, ...,   0,   0,   0],
       [ 22,  36,   0, ...,   0,   0,   0],
       [210, 166,   0, ...,   0,   0,   0]], dtype=int64)
```

Utilizaremos el **método del Codo** para encontrar el número óptimo de Clusters. Este ciclo ajustará el algoritmo K-Means a nuestros datos y después calculará la suma de cuadrados dentros del cluster y será añadido a la lista wcss.

Se utilizará **inertia_** que calcula la suma de distancias cuadradas de la muestra a su centro de agrupación más cercano.

In [191]:
```
wcss = []
for i in range(1,43):
    model_kmeans = KMeans(n_clusters=i, init ='k-means++', max_iter=300, n_ini
t=10, random_state = rs)
    train = np.array(train_X)
    model_kmeans.fit(train_X)
    wcss.append(model_kmeans.inertia_)
```

In [192]:
```
plt.figure(figsize=(8,4))
sns.lineplot(range(1,43), wcss,marker='o',color='red')
plt.title('Método del codo')
plt.xlabel('Número de clusters')
plt.ylabel('WCSS')
plt.show()
```



Definiremos 20 Clusters. Para evaluar cuántos clusters agruparán mejor se calculara Silouette Score. Valores cercanos a 1 nos indicarán buenos resultados.

In [193]:
```
model_kmeans = KMeans(n_clusters = 20, init = 'k-means++', max_iter=300 , n_in
it=10, random_state =rs)
```

In [194]:
```
y_kmeans = model_kmeans.fit_predict(test_X)
```

In [195]:
```
print(y_kmeans)
```

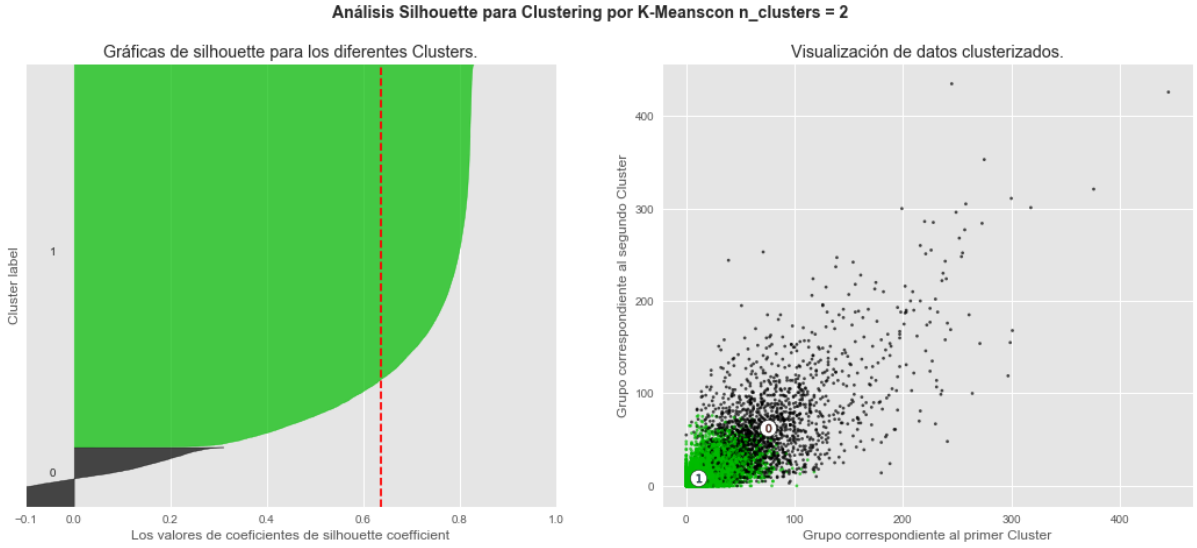
```
[ 9  7  7 ...  4 10 12]
```

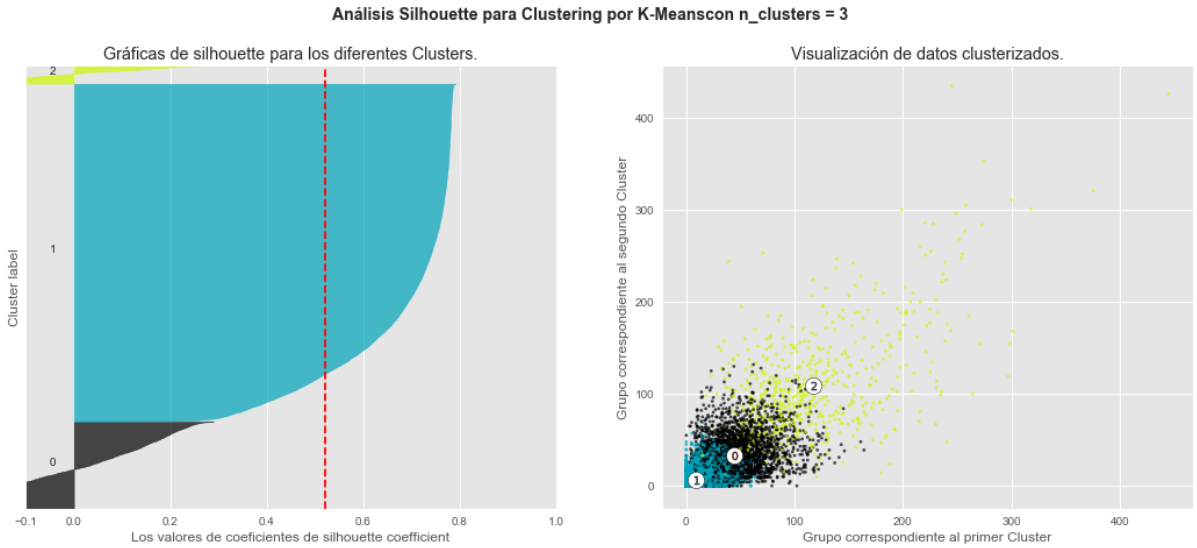
De acuerdo al puntaje y las gráficas, observamos que para 2 Clusters se obtiene el mejor puntaje. En el análisis exploratorio inferimos que esto se trata de títulos populares y de nicho, por lo que nos parece esperable este resultado.

In [313]:
```
# Puntaje Silhouette
import matplotlib.cm as cm
for n_clusters in range(2,21):
    # Creando subplot con una fila y dos columnas
    fig, (ax1, ax2) = plt.subplots(1, 2)
    fig.set_size_inches(18, 7)


    ax1.set_xlim([-0.1, 1])
    ax1.set_ylim([0, len(test_X) + (n_clusters + 1) * 10])


    clusterer = KMeans(n_clusters=n_clusters, random_state=rs)
    cluster_labels = clusterer.fit_predict(test_X)

    silhouette_avg = silhouette_score(test_X, cluster_labels)
    print("Para n_clusters =", n_clusters,
          "El promedio silhouette_score es :", silhouette_avg)

    sample_silhouette_values = silhouette_samples(test_X, cluster_labels)

    y_lower = 20
    for i in range(n_clusters):
        ith_cluster_silhouette_values = \
            sample_silhouette_values[cluster_labels == i]

        ith_cluster_silhouette_values.sort()

        size_cluster_i = ith_cluster_silhouette_values.shape[0]
        y_upper = y_lower + size_cluster_i

        color = cm.nipy_spectral(float(i) / n_clusters)
        ax1.fill_betweenx(np.arange(y_lower, y_upper),
                          0, ith_cluster_silhouette_values,
                          facecolor=color, edgecolor=color, alpha=0.7)

        ax1.text(-0.05, y_lower + 0.5 * size_cluster_i, str(i))

        y_lower = y_upper + 10

    ax1.set_title("Gráficas de silhouette para los diferentes Clusters.")
    ax1.set_xlabel("Los valores de coeficientes de silhouette coefficient")
    ax1.set_ylabel("Cluster label")

    ax1.axvline(x=silhouette_avg, color="red", linestyle="--")

    ax1.set_yticks([])
    ax1.set_xticks([-0.1, 0, 0.2, 0.4, 0.6, 0.8, 1])


    colors = cm.nipy_spectral(cluster_labels.astype(float) / n_clusters)
    ax2.scatter(test_X[:, 0], test_X[:, 1], marker='.', s=30, lw=0, alpha=0.7,
                c=colors, edgecolor='k')


    centers = clusterer.cluster_centers_
```

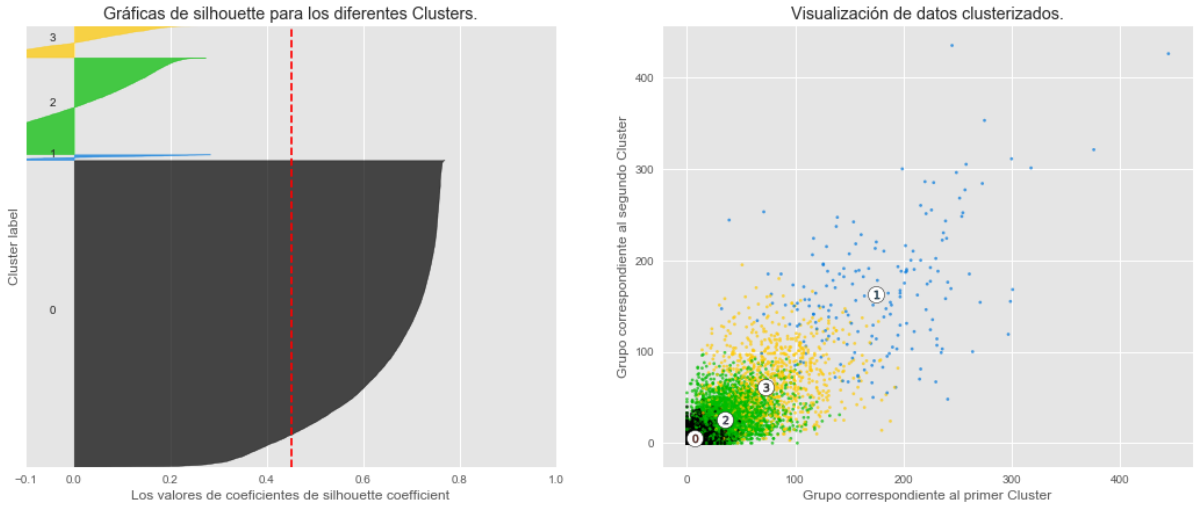
```
    ax2.scatter(centers[:, 0], centers[:, 1], marker='o',
                c="white", alpha=1, s=200, edgecolor='k')

    for i, c in enumerate(centers):
        ax2.scatter(c[0], c[1], marker='$%d$' % i, alpha=1,
                    s=50, edgecolor='k')

    ax2.set_title("Visualización de datos clusterizados.")
    ax2.set_xlabel("Grupo correspondiente al primer Cluster")
    ax2.set_ylabel("Grupo correspondiente al segundo Cluster")

    plt.suptitle(("Análisis Silhouette para Clustering por K-Means"
                  "con n_clusters = %d" % n_clusters),
                 fontsize=14, fontweight='bold')

plt.show()
```

```
Para n_clusters = 2 El promedio silhouette_score es : 0.637538083730868
Para n_clusters = 3 El promedio silhouette_score es : 0.5215960648905454
Para n_clusters = 4 El promedio silhouette_score es : 0.4514251810301966
Para n_clusters = 5 El promedio silhouette_score es : 0.435515626447497
Para n_clusters = 6 El promedio silhouette_score es : 0.4164179719105466
Para n_clusters = 7 El promedio silhouette_score es : 0.4152583793284396
Para n_clusters = 8 El promedio silhouette_score es : 0.3592289156566957
Para n_clusters = 9 El promedio silhouette_score es : 0.3245885157216861
Para n_clusters = 10 El promedio silhouette_score es : 0.3196283810355856
Para n_clusters = 11 El promedio silhouette_score es : 0.3235502991793944
Para n_clusters = 12 El promedio silhouette_score es : 0.3104763150955149
Para n_clusters = 13 El promedio silhouette_score es : 0.3048040019276014
Para n_clusters = 14 El promedio silhouette_score es : 0.27280619432325687
Para n_clusters = 15 El promedio silhouette_score es : 0.25655578996645906
Para n_clusters = 16 El promedio silhouette_score es : 0.24734742559355682
Para n_clusters = 17 El promedio silhouette_score es : 0.22534579307109903
Para n_clusters = 18 El promedio silhouette_score es : 0.2406114653673257
Para n_clusters = 19 El promedio silhouette_score es : 0.21106616497172614
Para n_clusters = 20 El promedio silhouette_score es : 0.22144285155944746
```

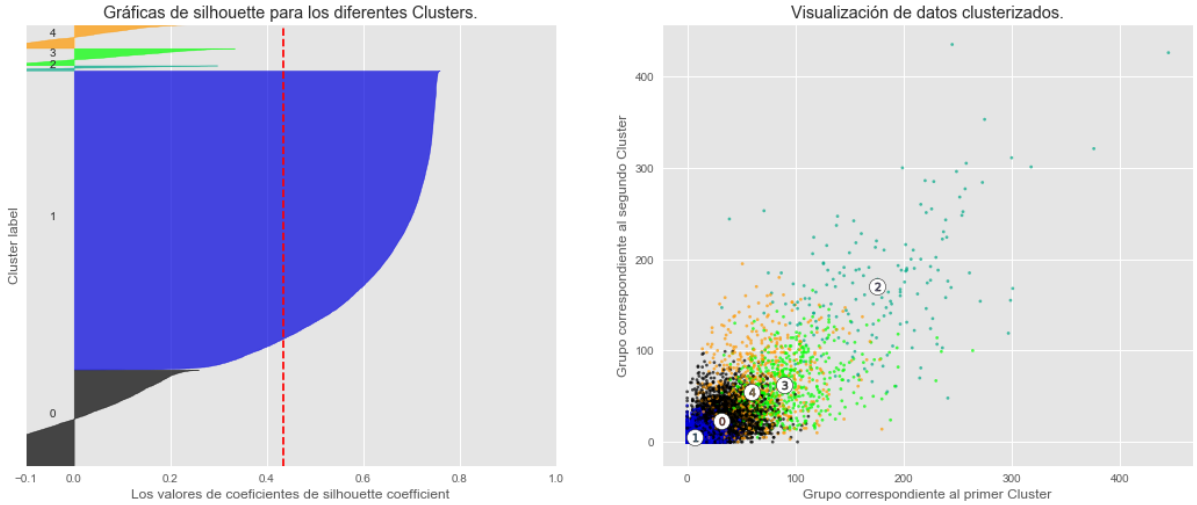




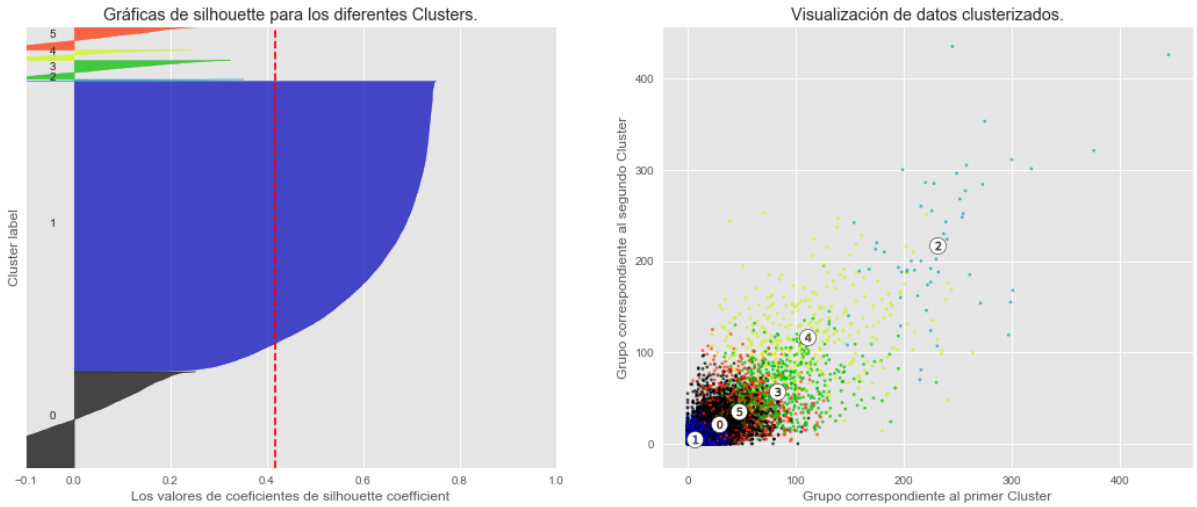
Análisis Silhouette para Clustering por K-Meanscon n_clusters = 4



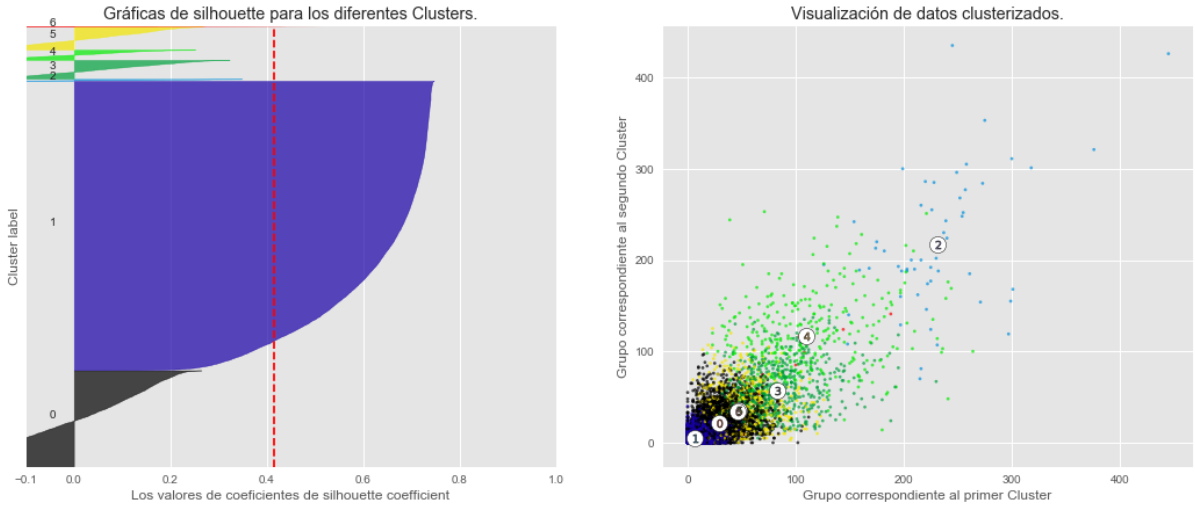
Análisis Silhouette para Clustering por K-Meanscon n_clusters = 5



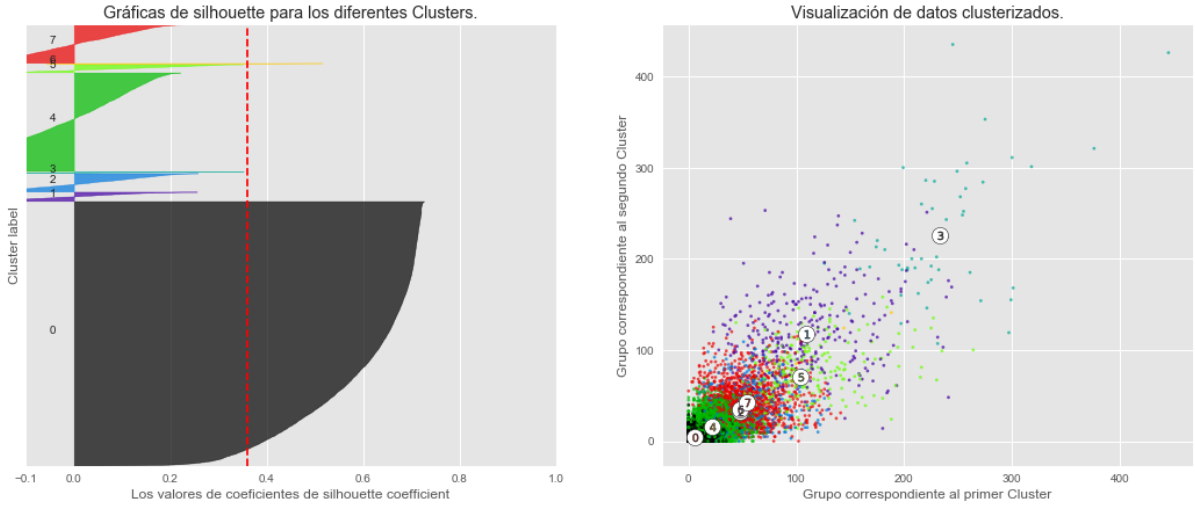
Análisis Silhouette para Clustering por K-Meanscon n_clusters = 6



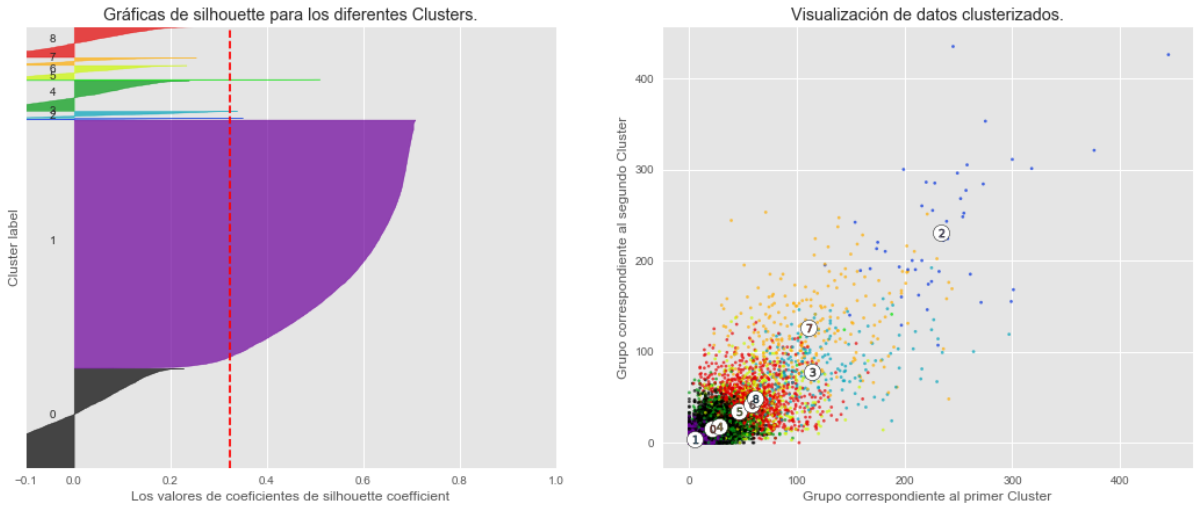
Análisis Silhouette para Clustering por K-Meanscon n_clusters = 7



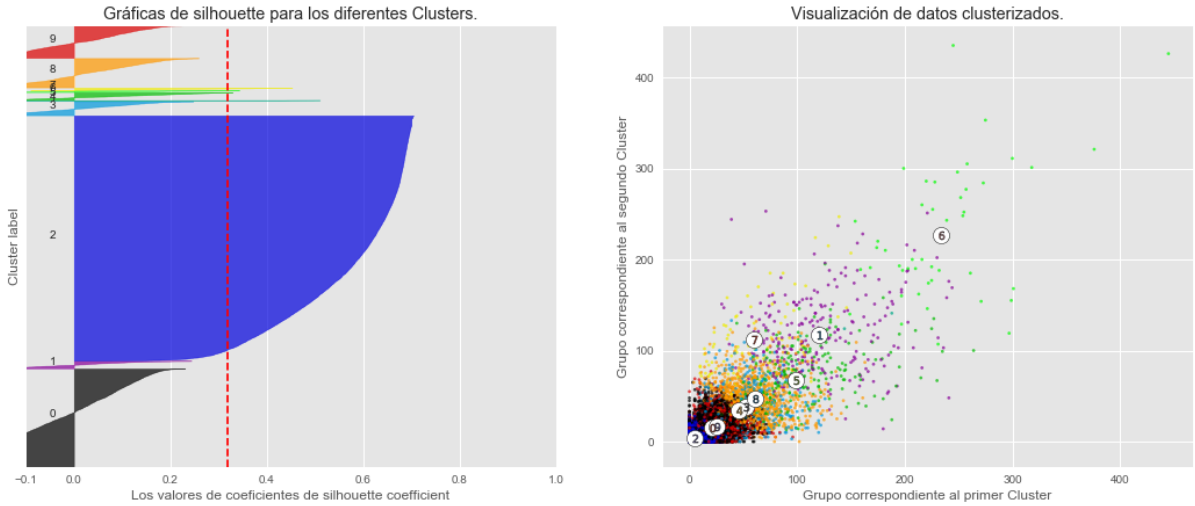
Análisis Silhouette para Clustering por K-Meanscon n_clusters = 8



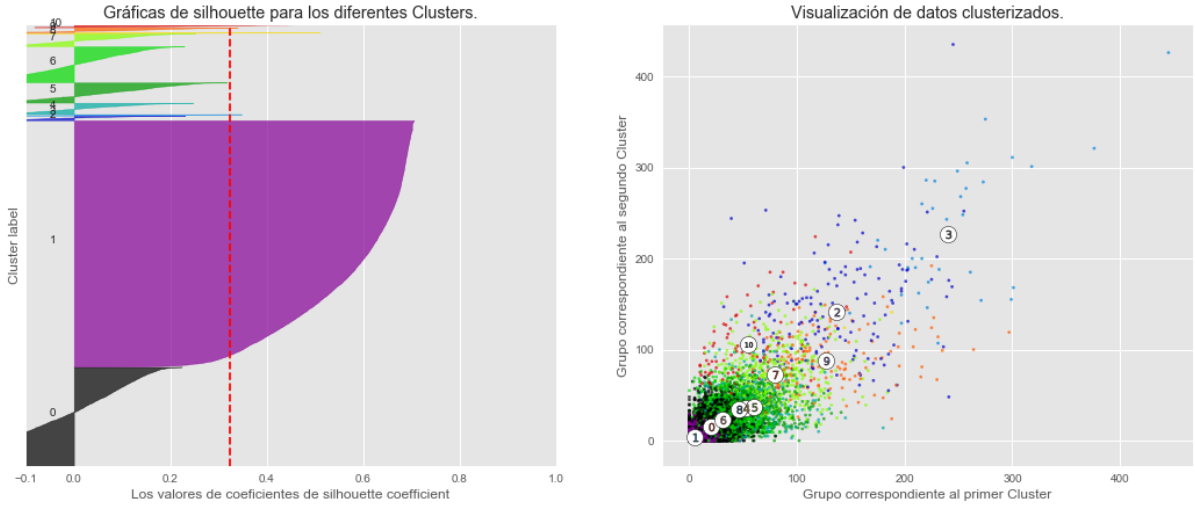
Análisis Silhouette para Clustering por K-Meanscon n_clusters = 9



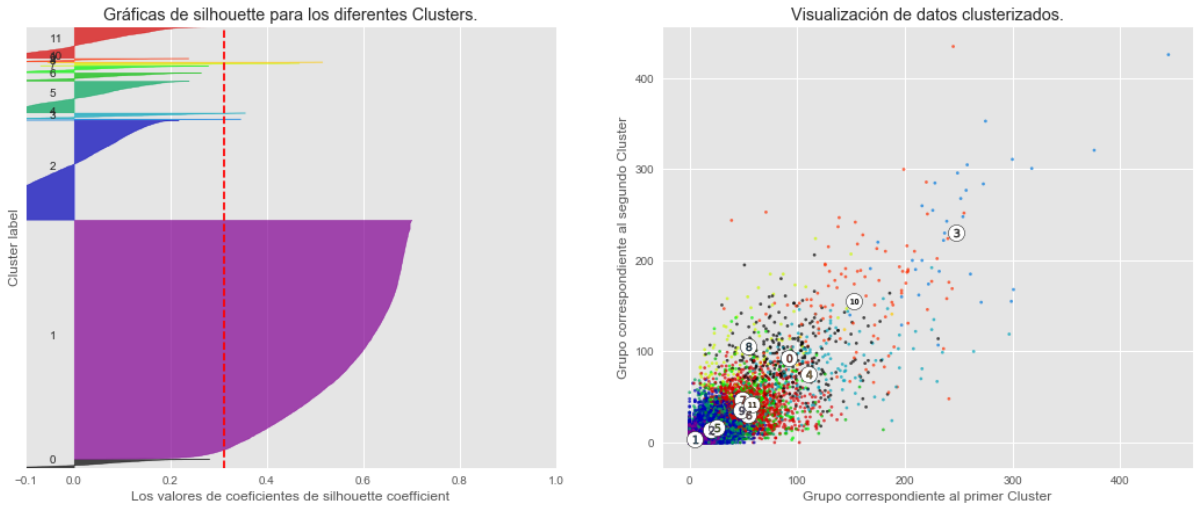
Análisis Silhouette para Clustering por K-Meanscon n_clusters = 10



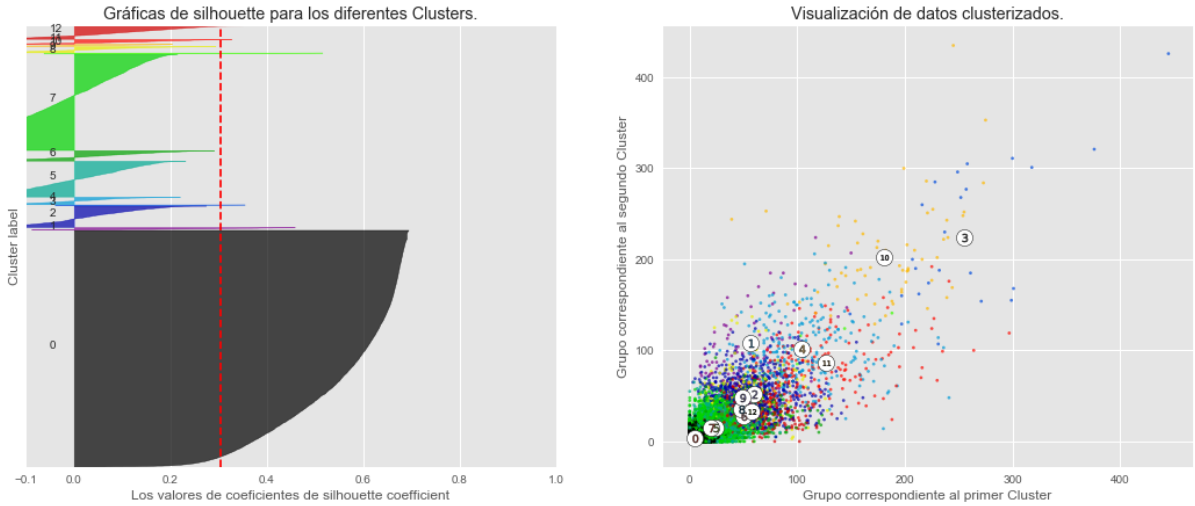
Análisis Silhouette para Clustering por K-Meanscon n_clusters = 11



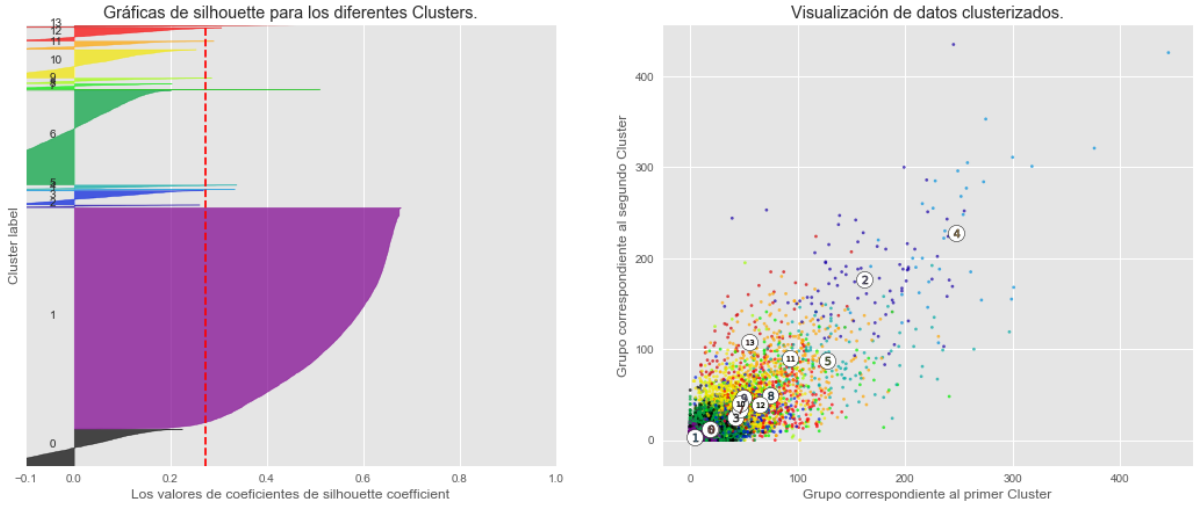
Análisis Silhouette para Clustering por K-Meanscon n_clusters = 12



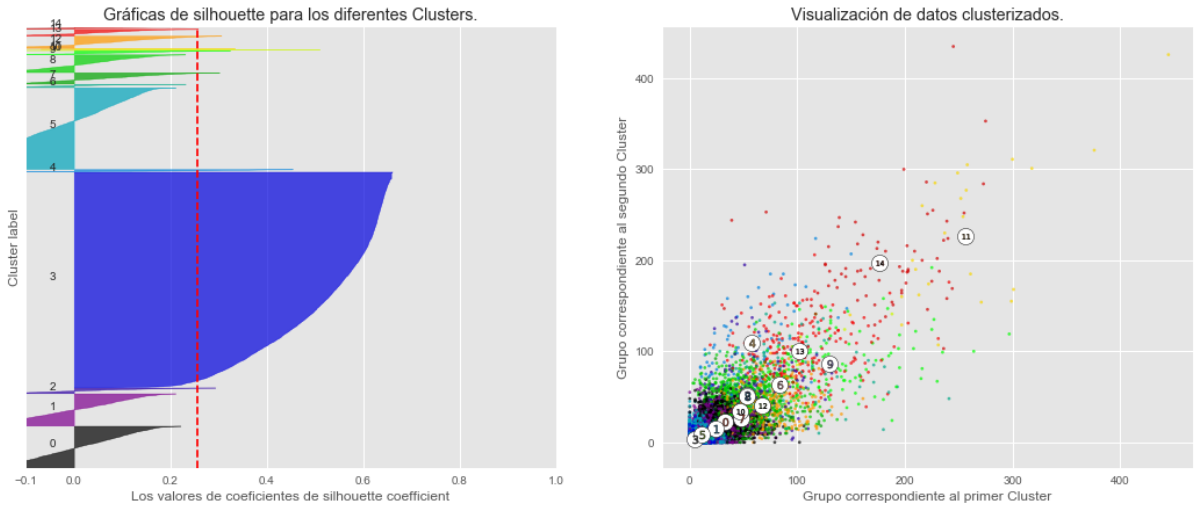
Análisis Silhouette para Clustering por K-Meanscon n_clusters = 13



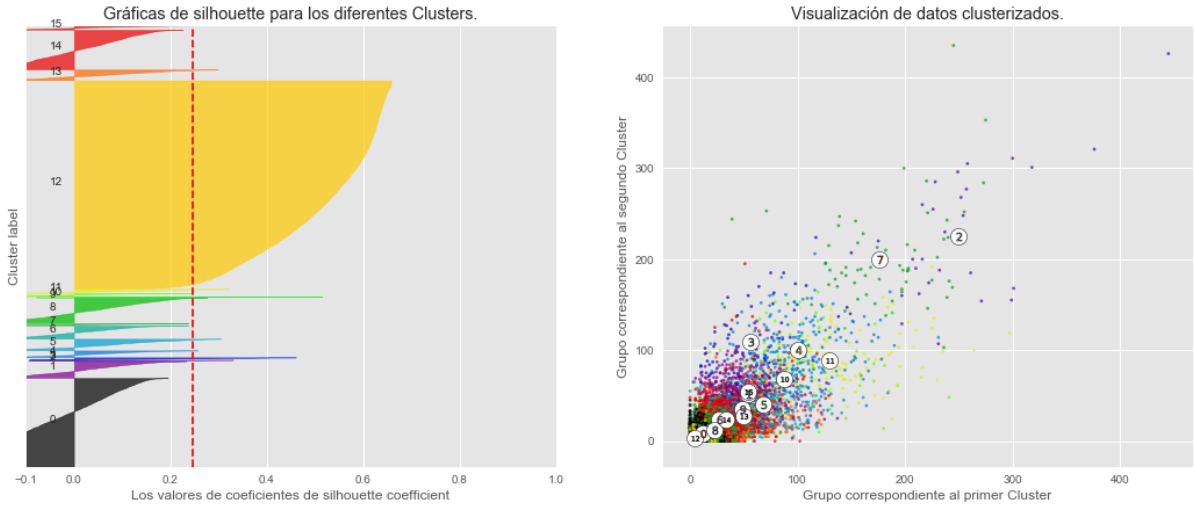
Análisis Silhouette para Clustering por K-Meanscon n_clusters = 14



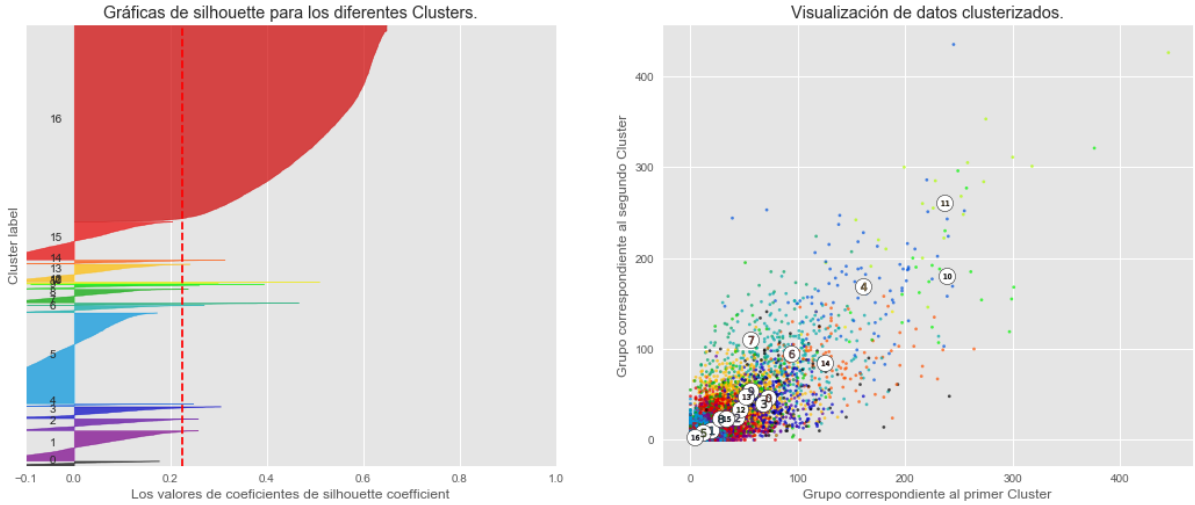
Análisis Silhouette para Clustering por K-Meanscon n_clusters = 15



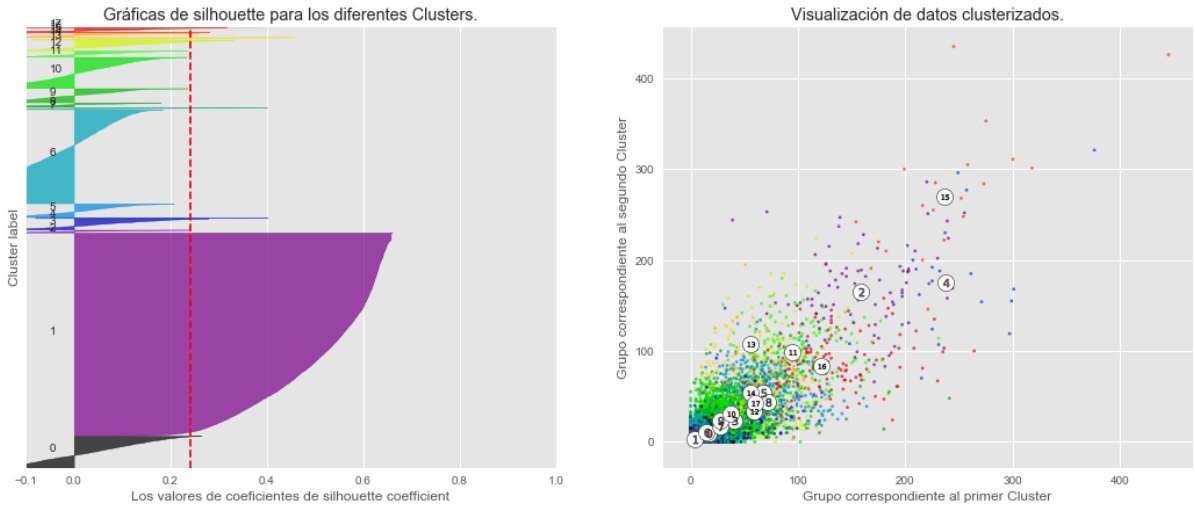
Análisis Silhouette para Clustering por K-Meanscon n_clusters = 16

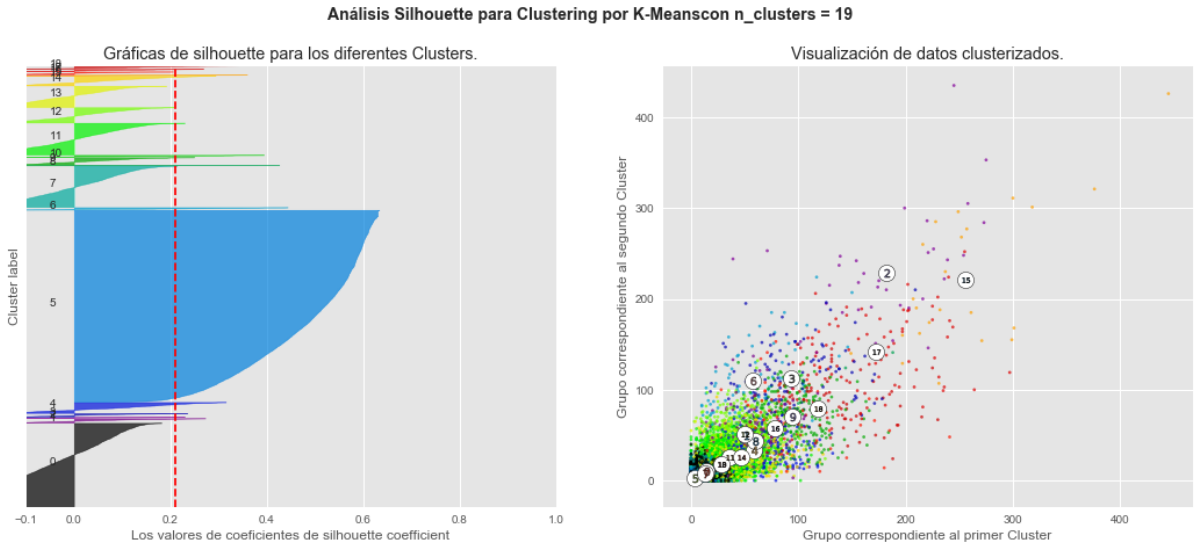


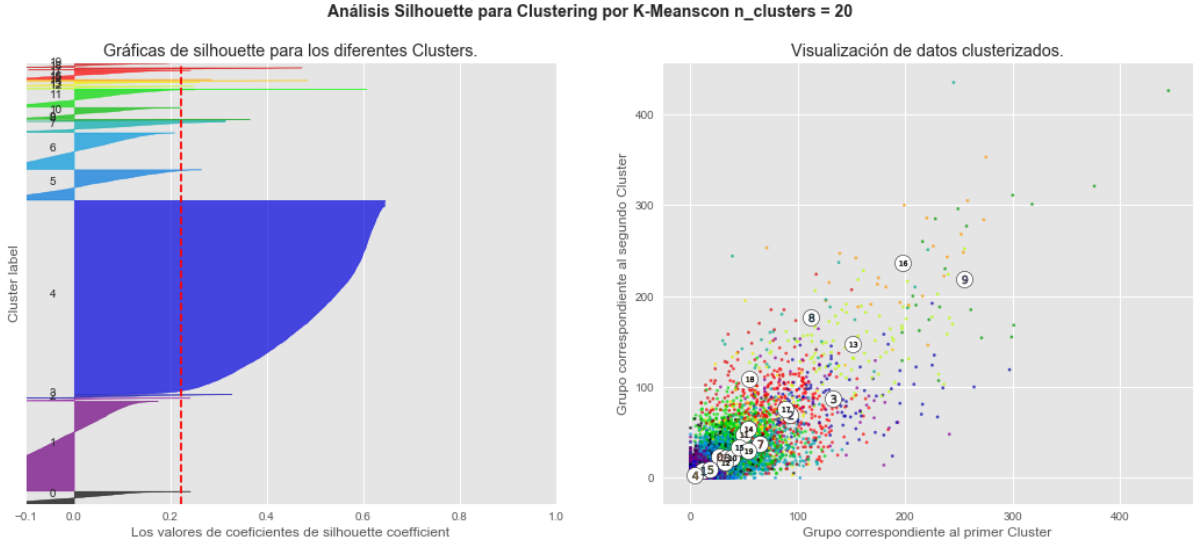
Análisis Silhouette para Clustering por K-Meanscon n_clusters = 17



Análisis Silhouette para Clustering por K-Meanscon n_clusters = 18







Se exporta modelo realizado con la selección de 20 Clusters por el método del codo.

In [317]:
```
pickle.dump(model_kmeans, open('model_kmeans_20.sav', 'wb'))
```

Según las métricas de silhouette_score escogimos 7 clusters para nuestra estrategia.

In [318]:
```
model_kmeans_7 = KMeans(n_clusters = 7, init = 'k-means++', max_iter=300 , n_i
nit=10, random_state =rs)
```

In [319]:
```
pickle.dump(model_kmeans_7, open('model_kmeans_7.sav', 'wb'))
```

Modelamiento Apriori

La librería **Apriori** que usaremos requiere nuestro set de datos como una lista de listas, donde el set de datos completo es una gran lista y cada transaccion en el dataset es una lista interna dentro de la lista gigante. El set de datos de entrada contiene un campo apriori, que contiene una lista con los géneros que ha visto cada usuario.

In [302]:
```
df_generos_1= pd.read_csv("vs_apriori_generos_VF_woNone.csv", sep=";")
```

In [303]:
```
df_generos_1.head()
```

Out[303]:

	user_id	Action	Adventure	Cars	Comedy	Dementia	Demons	Drama	Ecchi	Fantasy	...
0	1	Action	NaN	NaN	Comedy	NaN	Demons	Drama	Ecchi	Fantasy	...
1	3	Action	Adventure	NaN	Comedy	NaN	NaN	Drama	NaN	Fantasy	...
2	4	Action	NaN	NaN	Comedy	NaN	NaN	NaN	NaN	Fantasy	...
3	6	NaN	NaN	NaN	Comedy	NaN	NaN	NaN	NaN	NaN	...
4	7	Action	Adventure	NaN	Comedy	NaN	Demons	Drama	Ecchi	Fantasy	...

5 rows × 45 columns

Seleccionamos solo la columna que tiene las preferencias de cada usuario.

In [306]:
```
observation_ = []

observation_ = df_generos_1["aprioris"]
```

In [307]:
```
observation_
```

Out[307]:
```
0        ['Action', 'Comedy', 'Demons', 'Drama', 'Ecchi...
1        ['Action', 'Adventure', 'Comedy', 'Drama', 'Fa...
2        ['Action', 'Comedy', 'Fantasy', 'Historical', ...
3        ['Comedy', 'Historical', 'Mystery', 'School', ...
4        ['Action', 'Adventure', 'Comedy', 'Demons', 'D...
                              ...
48298    ['Action', 'Adventure', 'Cars', 'Comedy', 'Dra...
48299    ['Adventure', 'Comedy', 'Drama', 'Ecchi', 'Fan...
48300    ['Adventure', 'Comedy', 'Drama', 'Kids', 'Magi...
48301       ['Drama', 'Horror', 'Psychological', 'Sci-Fi']
48302    ['Action', 'Adventure', 'Comedy', 'Demons', 'D...
Name: aprioris, Length: 48303, dtype: object
```

Los parámetros a usar en el algoritmo apriori son:

- **min_support**: Este parámetro es usado para seleccionar los items con valores de soporte más grandes que el valor especificado por el parámetro.
- **the min_confidence**: filtra aquellas reglas que tienen confianza mayor que el umbral de confianza especificado en el parámetro.
- **the min_lift**: especifica el valor de elevación mínimo para las reglas preseleccionadas.
- **the min_length**: especifica el número mínimo de items que quieres en tus reglas.

Utilizamos diferentes valores en los parámetros para ver cuáles se adaptaban mejor a nuestro set de datos.

In [88]:
```
associations = apriori(observation_, min_length = 3, min_support = 0.001, min_
confidence = 0.01, min_lift = 2)
```

In [89]:
```
associations = list(associations)
```

In [90]:
```
print(len(associations))
```

```
1106
```

In [91]:
```
resultDataFrame=pd.DataFrame(inspect(associations),
                columns=['ID1','ID2','support','confidence','lift'])
```

In [92]:
```
resultDataFrame
```

Out[92]:

	ID1	ID2	support	confidence	lift
0	(Action,)	(Adventure,)	0.030660	0.172604	2.019414
1	(Action,)	(AdventureComedy,)	0.023173	0.130457	2.143079
2	(AdventureComedyEcchi,)	(Action,)	0.001159	0.684211	3.851903
3	(AdventureDemonsGame,)	(Action,)	0.001604	1.000000	5.629704
4	(Action,)	(AdventureEcchi,)	0.002228	0.012544	5.212689
...	...	...	...	...	...
1101	(ComedyEcchi,)	(Supernatural, RomanceSchool, nan, FantasyHare...	0.001248	0.076503	61.311475
1102	(Action,)	(nan, MartialArts, Shounen, SuperPower, Fantasy)	0.002406	0.013547	5.428643
1103	(Adventure,)	(Sci-Fi, Military, Space, nan, DramaMecha)	0.001070	0.012513	7.389276
1104	(AdventureComedy,)	(nan, MartialArts, Shounen, SuperPower, Fantasy)	0.002406	0.039531	15.840828
1105	(Action,)	(AdventureComedy, nan, MartialArts, Shounen, S...	0.002406	0.013547	5.629704

1106 rows × 5 columns

In [93]:
```
resultDataFrame.to_csv('C:\\apriori_generos_00010012.csv', header=False, index=False, sep=';')
```

Para los parámetros anteriores tuvimos muy pocos resultados de asociaciones, por lo que variamos los parámetros para obtener más asociaciones. El parámetro que afecta más la cantidad de resultados es el soporte.

In [102]:
```
associations2 = apriori(observation_, min_length = 3, min_support = 0.0005, min_confidence = 0.01, min_lift = 2)
```

In [103]:
```
associations2 = list(associations2)
```

In [108]:
```
print(len(associations2))
```

2444

In [234]:
```
resultDataFrame2=pd.DataFrame(inspect(associations2),
                columns=['ID1','ID2','support','confidence','lift'])
```

In [106]:
```
resultDataFrame2
```

Out[106]:

	ID1	ID2	support	confidence	lift
0	(Action,)	(Adventure,)	0.030660	0.172604	2.019414
1	(Action,)	(AdventureComedy,)	0.023173	0.130457	2.143079
2	(AdventureComedyEcchi,)	(Action,)	0.001159	0.684211	3.851903
3	(AdventureComedyGame,)	(Action,)	0.000802	1.000000	5.629704
4	(AdventureComedyMystery,)	(Action,)	0.000535	0.857143	4.825461
...	...	...	...	...	...
2439	(Military,)	(Romance, Sci-Fi, Music, Space, nan)	0.000624	0.025271	40.505415
2440	(AdventureComedy,)	(Ecchi, RomanceSchool, nan, Drama, FantasyHare...	0.000891	0.014641	16.427526
2441	(Action,)	(AdventureComedy, nan, MartialArts, Shounen, S...	0.002406	0.013547	5.629704
2442	(HaremMecha,)	(Comedy, Sci-FiShounen, Space, nan, Police, Ac...	0.000624	1.000000	1602.857143
2443	(ActionMecha,)	(Romance, Sci-Fi, Military, Music, Space, nan)	0.000624	0.100000	160.285714

2444 rows × 5 columns

In [109]:
```
resultDataFrame2.to_csv('C:\\apriori_generos_000050012.csv', header=False, index=False, sep=';')
```

Observamos que con un pequeño ajuste en el valor de soporte obtuvimos muchos más resultados.

In [ ]:
```

```

Prueba - Fundamentos Data Science

Martes 19 de mayo de 2020, Santiago de Chile

Sakura SPA

Miembros de la Célula:

Susana Arce

Fabiola Aravena

Rodrigo Pereira

Administrador de Contrato: Gonzalo Seguel

Sponsor: Andrea Villaroel

Objetivo:

Nuestro sponsor posee un cátalogo de contenido audiovisual de 12.294 títulos de Anime, requiere poder efecturar recomendaciones de qué ver a los usuarios en base a otros Anime han sido de su gusto.

Propuesta:

Aplicación web en donde el usuario seleccione de una lista de Anime propuestos los que han sido de su gusto, y en base a ello se le entregue una lista de otras alternativas que sean a fin con sus preferencias.

Recomendador de Películas

El presente código procesa el resultado de modelamiento, entregado en formato .csv y genera una lista de animes sugeridos.

Finalmente se incluye un prototipo de algoritmo que se implementará en la plataforma web.

In [450]:
```
# Importación de librerías para procesamiento de datos.
import pandas as pd
import numpy as np

import matplotlib.pyplot as plt
import seaborn as sns
plt.style.use('seaborn')

import re
import random as random

import requests
from ipywidgets import Image

import warnings
warnings.filterwarnings(action="ignore")
```

In [489]:
```
subgenre = ["Action","Adventure","Cars","Comedy","Dementia","Demons
","Drama","Ecchi","Fantasy","Game","Harem","Hentai","Historical","H
orror","Josei","Kids","Magic","MartialArts","Mecha","Military","Mus
ic","Mystery","Parody","Police","Psychological","Romance","Samurai"
,
"School","Sci-Fi","Seinen","Shoujo","ShoujoAi","Shounen","ShounenAi
","SliceofLife","Space","Sports","Supernatural","SuperPower","Thril
ler","Vampire","Yaoi","Yuri"]
```

Recogida de datos

In [232]:
```
#Importamos los data sets para su uso
# Los archivos pueden ser descargados desde:
# https://drive.google.com/open?id=1TlhsxxaTENAO6xIw-EvvZhRAuJaJtPF
B

df_titulos_VF = pd.read_csv("titulos_generos_VF20200519.csv", sep="
;")
df_interacciones = pd.read_csv("interacciones_generos_VF.csv", sep=
";")
vs_afinidad_generos = pd.read_csv('vs_afinidad_generos_VF.csv',sep=
';')
vs_generos_max = pd.read_csv('vs_afin_maxgen_VF.csv',sep=';')
```

In [3]:
```
df_titulos_VF.shape
```

Out[3]: (7492, 53)

In [116]:
```
df_titulos_VF.sort_values(["rating","log_members"],ascending=False,
inplace=True)
df_titulos_VF.head()
```

Out[116]:

	Unnamed: 0	anime_id	url	name	genre
4689	4689	32935	https://imdb-api.com/images/original/MV5BNzQ1M...	Haikyuu!!: Karasuno Koukou VS Shiratorizawa Ga...	Comedy, Drama, School, Shounen, Sports
0	0	820	NaN	Ginga Eiyuu Densetsu	Drama, Military, Sci-Fi, Space
4690	4690	28851	https://imdb-api.com/images/original/MV5BZGRkO...	Koe no Katachi	Drama, School, Shounen
3672	3672	26259	NaN	Mienu Me ni Kanjita Kumotoriyama no Asahi	Drama, Kids
4691	4691	28957	https://imdb-api.com/images/original/MV5BMjM5N...	Mushishi Zoku Shou: Suzu no Shizuku	Adventure, Fantasy, Historical, Mystery, Seine...

5 rows × 53 columns

In [4]:
```
df_interacciones.shape
```

Out[4]: (775258, 57)

In [5]:
```
vs_afinidad_generos.shape
```

Out[5]: (48303, 44)

In [6]:
```
vs_generos_max.shape
```

Out[6]: (48303, 51)

Estos data sets fueron procesados anteriormente en la fase procesamiento, para ser traspasados a modelamiento. Ahora se utilizarán para la predicción final.

In [56]:
```
# Importamos la predicción apriori

predic_apriori = pd.read_csv("apriori_generos_000050012.csv", sep="
;",
                             header=None,
                              names=["id1","id2","support","confiden
ce","lift"])
```

In [57]:
```
predic_apriori.head()
```

Out[57]:

	id1	id2	support	confidence	lift
0	('Action',)	('Adventure',)	0.030660	0.172604	2.019414
1	('Action',)	('AdventureComedy',)	0.023173	0.130457	2.143079
2	('AdventureComedyEcchi',)	('Action',)	0.001159	0.684211	3.851903
3	('AdventureComedyGame',)	('Action',)	0.000802	1.000000	5.629704
4	('AdventureComedyMystery',)	('Action',)	0.000535	0.857143	4.825461

Procesamiento del resultado de predicción para utilizarlo

In [58]:
```
predic_apriori.sort_values(["confidence","support"],ascending=False
,inplace=True)
predic_apriori.head()
```

Out[58]:

	id1	id2	support	confidence	lift
348	('HaremMusic',)	('Shoujo',)	0.002496	1.0	20.000000
1278	('HaremMusic',)	('Shoujo', 'nan')	0.002496	1.0	20.000000
232	('DementiaPsychological',)	('Romance',)	0.001961	1.0	12.169197
1076	('DementiaPsychological',)	('Romance', 'nan')	0.001961	1.0	12.169197
219	('ComedyRomanceSchoolSeinen',)	('SliceofLife',)	0.001783	1.0	19.411765

In [182]:
```
def limpieza_id1(fila):
    id1 = fila["id1"]
    id1 = id1.replace("'nan'","")
    id1 = id1.replace("(","").replace(")","").replace(",","").repla
ce("'","").replace(" ","")

    id1 = id1.replace("MartialArts","Tmp1")
    id1 = id1.replace("Sci-Fi","Tmp2")
    id1 = id1.replace("ShoujoAi","Tmp3")
    id1 = id1.replace("ShounenAi","Tmp4")
    id1 = id1.replace("SliceofLife","Tmp5")
```

```
    id1 = id1.replace("SuperPower","Tmp6")


    id1 = re.findall('[A-Z][^A-Z]*',id1)

    if len(id1)==1:
        id1_final = id1[0]
        id1_final = id1_final.replace("Tmp1","MartialArts")
        id1_final = id1_final.replace("Tmp2","Sci-Fi")
        id1_final = id1_final.replace("Tmp3","ShoujoAi")
        id1_final = id1_final.replace("Tmp4","ShounenAi")
        id1_final = id1_final.replace("Tmp5","SliceofLife")
        id1_final = id1_final.replace("Tmp6","SuperPower")

    else:
        id1_final = None

    return id1_final



def limpieza_id2(fila):
    id2 = fila.id2
    id2 = id2.replace("'nan'","")
    id2 = id2.replace("(","").replace(")","").replace(",","").repla
ce("'","").replace(" ","")


    id2 = id2.replace("MartialArts","Tmp1")
    id2 = id2.replace("Sci-Fi","Tmp2")
    id2 = id2.replace("ShoujoAi","Tmp3")
    id2 = id2.replace("ShounenAi","Tmp4")
    id2 = id2.replace("SliceofLife","Tmp5")
    id2 = id2.replace("SuperPower","Tmp6")

    id2 = re.findall('[A-Z][^A-Z]*',id2)

    id2_final=[]

    for k in id2[0:3]:
        k = k.replace("Tmp1","MartialArts")
        k = k.replace("Tmp2","Sci-Fi")
        k = k.replace("Tmp3","ShoujoAi")
        k = k.replace("Tmp4","ShounenAi")
        k = k.replace("Tmp5","SliceofLife")
        k = k.replace("Tmp6","SuperPower")
        id2_final.append(k)


    return id2_final


predic_apriori["id1pro"] = predic_apriori.apply((limpieza_id1), axi
s=1)
```

```
predic_apriori["id2pro"] = predic_apriori.apply((limpieza_id2), axis=1)
```

In [183]:
```
predic_apriori.head()
```

Out[183]:

	id1	id2	support	confidence	lift	id1pr
348	('HaremMusic',)	('Shoujo',)	0.002496	1.0	20.000000	Non
1278	('HaremMusic',)	('Shoujo', 'nan')	0.002496	1.0	20.000000	Non
232	('DementiaPsychological',)	('Romance',)	0.001961	1.0	12.169197	Non
1076	('DementiaPsychological',)	('Romance', 'nan')	0.001961	1.0	12.169197	Non
219	('ComedyRomanceSchoolSeinen',)	('SliceofLife',)	0.001783	1.0	19.411765	Non

---

Se obtienen 2 columnas nuevas procesadas que relacionan directamente el genero sugerido por el modelamiento con algún genero existente, el cual se utilizará para hacer el match con el género más afin de cada usuario

---

Selección de usuario

Vamos a generar el supuesto flujo del usuario en la plataforma, considerando las posibilidades de que sea un usuario nuevo o un usuario registrado

In [313]:
```
t_user = random.choice(["N","E"]) # N: Usuario Nuevo E: Usuario Existente
```

In [329]:
```
# Simular de Ingreso al sitio

if t_user is "N":
    seleccion = random.sample(df_interacciones["anime_id"].tolist(),10) #simula la seleccion del usuario nuevo
elif t_user is "E":
    usuario = random.sample(df_interacciones["user_id"].tolist(),1)[0] #simula el ingreso del user_id
```

Búsqueda del género afin

Determinamos el género o los géneros más afin para los usuarios nuevos o registrados

In [363]:
```
def max_genre_nuevo(seleccion):
# Determina el género de máxima afinidad con el usuario a partir de
l
#indicador de afinidad construído y aplicado a su selección de títu
los entregada

    data_new_user = pd.merge(left = pd.DataFrame({"anime_id":selecc
ion}),left_on="anime_id",
                             right = df_titulos_VF , right_on ="ani
me_id")
    data_new_user["user_id"] = "N001" #Para posteriormente construi
r una base de las interacciones de usuarios no registrados
    tmp = pd.pivot_table(data_new_user,values=subgenre,index=["user
_id"],aggfunc=sum).fillna(0)
    tmp1= np.argmax(tmp.iloc[0])
    max1=[]
    max1.append(tmp1)
    return max1

def max_genres(usuario):
# Determina el género de máxima afinidad con el usuario a partir de
l
#indicador de afinidad construído y aplicado a su registro historic
o
    max1 = vs_generos_max[vs_generos_max["user_id"] == usuario].loc
[:,"Maximos"].tolist()[0]
    max1 = max1.replace("[","").replace("]","").replace("'","").rep
lace(" ","").split(",")
    return max1
```

Se realiza el cruce de la predicción y luego con los animes a recomendar.

In [354]:
```
def recomendar_genres(max1):
# Obtiene el o los géneros entregados como predicción por el algori
tmo aprioro
    rec_genre = []
    for i in max1:
        for k in predic_apriori[predic_apriori["id1pro"]==i].loc[:,
"id2pro"].tolist()[0]:
            rec_genre.append(k)
    return rec_genre

def recomendar_anime(rec_genre):
# A partir de la predicción de género, se selecciona la lista de pe
lículas a sugerir.

    animes_rec = []
    for i in rec_genre:
        #tmp = df_titulos_VF[df_titulos_VF[i]==1].loc[:,"anime_id"]
#Exponemos método alternativo para asegurar visualización de imagen
es.
        tmp = df_titulos_VF[~df_titulos_VF["url"].isnull()][df_titu
los_VF[i]==1].loc[:,"anime_id"]
        for j in tmp:
            animes_rec.append(j)
    return animes_rec

def filtrar_anime(animes_rec,usuario):
# Sólo aplica a usuarios existentes, para no recomendar películas y
a vistas.
    animes_rec_vf = animes_rec
    animes_visto = df_interacciones[df_interacciones["user_id"]==us
uario]["anime_id"]
    for i in animes_visto:
        try:
            animes_rec.remove(i)
        except ValueError:
            pass
    return animes_rec_vf
```

Se realiza un simulador de la función de producción, que actua de setup general para operar en las funciones anteriormente definidas

In [470]:
```
def produccion(t_user,
               usuario=None,
               seleccion=None):
    print("-------------------------------------------------------
------------------")
    if t_user is "N":
        max1 = max_genre_nuevo(seleccion)
        print("Hola nuevo usuario, bienvenido a nuestro recomendado
r de animes")
        print("Gracias por indicarnos tus títulos de referencia par
a iniciar la búsqueda")
    if t_user is "E":
        max1 = max_genres(usuario)
        print("Hola otra vez!")
        text ="¿Cómo estas usuario "+str(usuario)+" ?"
        print(text)

    text = "Te gusta el "+max1[0]+", porque no pruebas nuestras sug
erencias!"
    print(text)

    rec1 = recomendar_genres(max1)
    reco = recomendar_anime(rec1)

    if t_user is "N":
        VF = reco
    if t_user is "E":
        VF= filtrar_anime(reco,usuario)
    print("\n")
    print("\n")

    for n,i in enumerate(VF[0:10]):
        anime_recomendado = df_titulos_VF[df_titulos_VF["anime_id"]
==i]
        titulo = anime_recomendado.loc[:,"name"].tolist()[0]
        formato = anime_recomendado.loc[:,"type"].tolist()[0]
        rating = anime_recomendado.loc[:,"rating"].tolist()[0]
        poster = anime_recomendado.loc[:,"url"].tolist()[0]
        print("Sugerencia " + str(n+1))
        print(titulo)
        print("Es una excelente "+ formato +" evaluada con un "+ st
r(rating))
        if n ==0:
            imagen = Image(value=requests.get(poster).content)
        print(poster)
        print("\n")
    print("-------------------------------------------------------
------------------")
    return imagen
```

In [ ]:
```
```

Prototipo para presentación

Genera en primera instancia de forma aleatoria las posibilidades de usuarios nuevos o usuario existente. Además genera una lista de títulos seleccionados para simular el ingreso que debería realizar un nuevo usuario

In [493]:
```
t_user = random.choice(["N","E"]) # N: Usuario Nuevo E: Usuario Exi
stente
usuario = random.sample(df_interacciones["user_id"].tolist(),1)[0],
seleccion = random.sample(df_interacciones["anime_id"].tolist(),10)
```

In [494]:
```
produccion(t_user,usuario,seleccion)
```

```
--------------------------------------------------------------------
--------
Hola otra vez!
¿Cómo estas usuario (39066,) ?
Te gusta el Adventure, porque no pruebas nuestras sugerencias!



Sugerencia 1
Vampire Hunter
Es una excelente OVA evaluada con un 6.83
https://imdb-api.com/images/original/MV5BZjhkZTRlNTQtNTVkMy00NTgxL
Tk5N2EtMzZlODczZWE2ZTNhL2ltYWdlL2ltYWdlXkEyXkFqcGdeQXVyMzM4MjM0Nzg
@._V1_Ratio0.8182_AL_.jpg


Sugerencia 2
Jie Mo Ren
Es una excelente ONA evaluada con un 6.65
https://imdb-api.com/images/original/nopicture.jpg


Sugerencia 3
Final Fantasy
Es una excelente OVA evaluada con un 6.25
https://imdb-api.com/images/original/nopicture.jpg


Sugerencia 4
Happy World!
Es una excelente OVA evaluada con un 6.11
https://imdb-api.com/images/original/MV5BY2E3MTNmNDctYTdmNi00NDY4L
WE4NTctMmU1ZjQ5ZTczNDk1XkEyXkFqcGdeQXVyMTA2NDI0NDM0._V1_Ratio0.727
3_AL_.jpg
```

```
Sugerencia 5
DragonBlade
Es una excelente Movie evaluada con un 5.95
https://imdb-api.com/images/original/MV5BYTQxZjA1YzktN2JiZS00OTdiL
ThlMmYtNTAyMTg3OWNhYzhiXkEyXkFqcGdeQXVyODE0OTY5MDg@._V1_Ratio0.727
3_AL_.jpg


Sugerencia 6
Garo: Guren no Tsuki
Es una excelente TV evaluada con un 5.96
https://imdb-api.com/images/original/MV5BNTRmYjFiNTMtMjdhMC00ZmI0L
TgyNTQtNTJlN2E5MmZhODc3XkEyXkFqcGdeQXVyNDQxNjIwNTI@._V1_Ratio0.727
3_AL_.jpg


Sugerencia 7
GO-GO Tamagotchi!
Es una excelente TV evaluada con un 7.17
https://imdb-api.com/images/original/MV5BYTRmYmQxNjQtZTE2MS00NTUxL
TgxNWItNDZjOWQyYzFjYmNiXkEyXkFqcGdeQXVyNzkwODIyMzc@._V1_Ratio0.727
3_AL_.jpg


Sugerencia 8
Hyoutan Suzume
Es una excelente Movie evaluada con un 5.0
https://imdb-api.com/images/original/nopicture.jpg


Sugerencia 9
Jim Button
Es una excelente TV evaluada con un 6.1
https://imdb-api.com/images/original/nopicture.jpg


Sugerencia 10
Kanimanji Engi
Es una excelente Movie evaluada con un 4.79
https://imdb-api.com/images/original/nopicture.jpg


--------------------------------------------------------------------
--------
```