HERNNDEZ_GONZALEZ_RICARDO_PARAMONT

June 18, 2021

proyecto # Análisis de Videojuegos con sus ventas y calificaciones ## Diplomado Ciencia de Datos - modulo III

Profesor: José Gustavo Fuentes Cabrera
Hernández González Ricardo Paramont

DataSet: Video_Games_Sales_as_at_22_Dec_2016.csv

0.1 Dependencias

[1]:
```
import numpy as np
import pandas as pd

from sklearn.preprocessing import MinMaxScaler,StandardScaler
from sklearn.decomposition import PCA
from sklearn.manifold import MDS,TSNE
from sklearn.cluster import AgglomerativeClustering,KMeans
from sklearn.mixture import GaussianMixture
from sklearn.metrics import silhouette_score
from sklearn.feature_selection import VarianceThreshold

from varclushi import VarClusHi

from scipy.stats import chisquare
from scipy.stats import kruskal
from statsmodels.stats.multicomp import MultiComparison

from itertools import chain
from functools import reduce

import matplotlib.pyplot as plt
import seaborn as sns
import cufflinks as cf
import plotly.express as px
import plotly.graph_objects as go

cf.go_offline()
```

```
pd.set_option('display.max_columns',None)
```

#

Procesamiento de Datos

0.2 Campos de información

Name: Nombre del videojuego.
Platform: Consola de videojuegos.
Year of release: Año de publicación del videojuego.
Genre: Género del videojuego.
Publisher: Empresa que publica el videojuego.
NA_sales: Ventas en undades en Norteamérica.
EU_sales: Ventas en unidades en Europa.
JP_sales: Ventas en unidades en Japón.
Other_sales: Ventas en otras partes del mundo.
Gloal_sales: Ventas mundiales totales.
Critic_score: Calificación agregada compilada por Metacritic.
Critic_count: Número de críticas usadas para el valor de Critic_Score.
User_score: Calificación dada por usuarios de Metacritic.
User_count: Número de criticas usadas para el valor de User_score.
Developer: Empresa desarrolladora del videojuego.
Rating: Rating según ESRB.

0.3 Lectura de Datos

[2]:
```
df = pd.read_csv('Video_Games_Sales_as_at_22_Dec_2016.csv')
print(f'Tamaño del dataset: {df.shape}')
df.head()
```

```
Tamaño del dataset: (6811, 16)
```

[2]:

	name	platform	year_of_release	genre	publisher	na_sales	eu_sales	jp_sales	other_sales	global_sales	critic_score
0	Wii Sports	Wii	2006.0	Sports	Nintendo	41.36	28.96	3.77	8.45	82.53	76.0
1	Mario Kart Wii	Wii	2008.0	Racing	Nintendo						
2	Wii Sports Resort	Wii	2009.0	Sports	Nintendo						
3	New Super Mario Bros.	DS	2006.0	Platform	Nintendo						
4	Wii Play	Wii	2006.0	Misc	Nintendo						

```
1     15.68     12.76      3.79         3.29         35.52          82.0
2     15.61     10.93      3.28         2.95         32.77          80.0
3     11.28      9.14      6.50         2.88         29.80          89.0
4     13.96      9.18      2.93         2.84         28.92          58.0

   critic_count  user_score  user_count developer rating
0          51.0         8.0       322.0  Nintendo      E
1          73.0         8.3       709.0  Nintendo      E
2          73.0         8.0       192.0  Nintendo      E
3          65.0         8.5       431.0  Nintendo      E
4          41.0         6.6       129.0  Nintendo      E
```

0.4 Definición de variables

[3]:
```
varc =␣
 ↪['year_of_release','na_sales','eu_sales','jp_sales','other_sales','global_sales',
       'critic_score','critic_count','user_score','user_count']
vard = ['platform','genre','publisher','rating','name','developer']
```

0.5 Completitud

[4]:
```
def completitud(df):
    comp=pd.DataFrame(df.isnull().sum())
    comp.reset_index(inplace=True)
    comp=comp.rename(columns={"index":"columna",0:"total"})
    comp["completitud"]=round((1-comp["total"]/df.shape[0])*100, 3)
    comp=comp.sort_values(by="completitud",ascending=True)
    comp.reset_index(drop=True,inplace=True)
    return comp
```

[5]:
```
completitud(df)
```

[5]:
```
             columna  total  completitud
0               name      0        100.0
1           platform      0        100.0
2   year_of_release      0        100.0
3              genre      0        100.0
4          publisher      0        100.0
5           na_sales      0        100.0
6           eu_sales      0        100.0
7           jp_sales      0        100.0
8        other_sales      0        100.0
9       global_sales      0        100.0
10      critic_score      0        100.0
11      critic_count      0        100.0
12        user_score      0        100.0
```

```
13       user_count      0        100.0
14        developer      0        100.0
15           rating      0        100.0
```

Afortunadamente, todas las columnas cuentas con registros completos.

0.6 Duplicados

[6]:
```
df.duplicated().sum()
```

[6]:
```
0
```

No hay duplicados.

0.7 Análisis Exploratorio

0.7.1 Continuas

[7]:
```
df[varc].describe()
```

[7]:
```
       year_of_release     na_sales     eu_sales     jp_sales  other_sales  \
count      6811.000000  6811.000000  6811.000000  6811.000000  6811.000000
mean       2007.451916     0.395068     0.236532     0.063900     0.082838
std           4.202178     0.968207     0.687961     0.287742     0.270124
min        1985.000000     0.000000     0.000000     0.000000     0.000000
25%        2004.000000     0.060000     0.020000     0.000000     0.010000
50%        2007.000000     0.150000     0.060000     0.000000     0.020000
75%        2011.000000     0.390000     0.210000     0.010000     0.070000
max        2016.000000    41.360000    28.960000     6.500000    10.570000

       global_sales  critic_score  critic_count   user_score    user_count
count   6811.000000   6811.000000   6811.000000  6811.000000   6811.000000
mean       0.778520     70.236970     28.954339     7.182866    174.907796
std        1.965288     13.858778     19.235502     1.439574    588.005932
min        0.010000     13.000000      3.000000     0.500000      4.000000
25%        0.110000     62.000000     14.000000     6.500000     11.000000
50%        0.290000     72.000000     25.000000     7.500000     27.000000
75%        0.750000     80.000000     39.000000     8.200000     88.000000
max       82.530000     98.000000    113.000000     9.600000  10665.000000
```

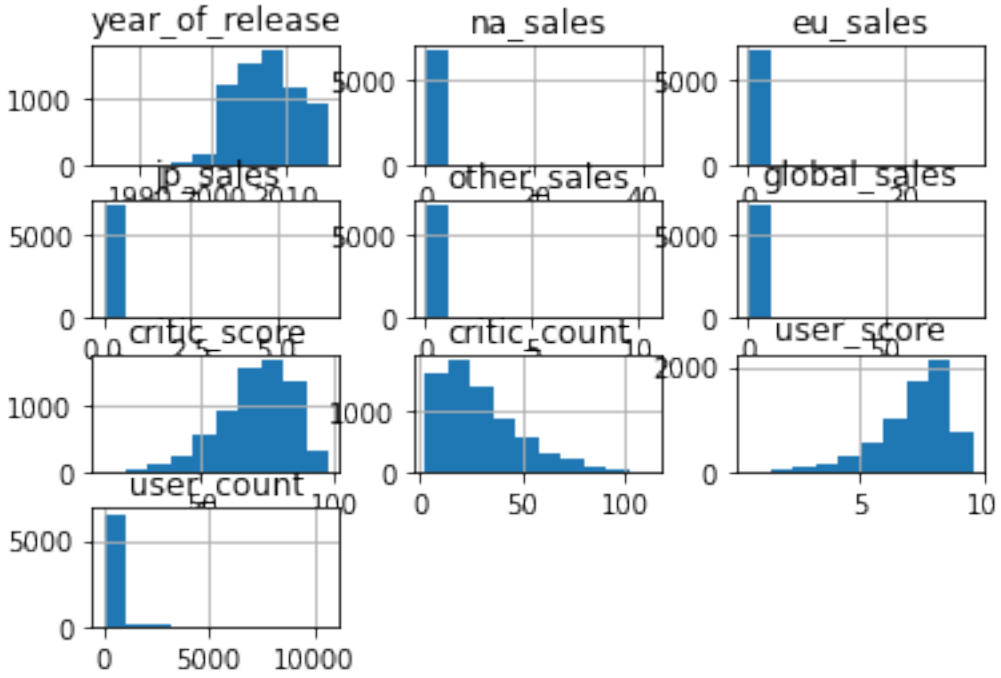
Se confirma que todas las variables contienen valores dentro de rangos esperados, según el valor min y max.

Distribuciones

[8]:
```
df[varc].hist()
```

[8]:
```
array([[<AxesSubplot:title={'center':'year_of_release'}>,
        <AxesSubplot:title={'center':'na_sales'}>,
```

```
 <AxesSubplot:title={'center':'eu_sales'}>],
[<AxesSubplot:title={'center':'jp_sales'}>,
 <AxesSubplot:title={'center':'other_sales'}>,
 <AxesSubplot:title={'center':'global_sales'}>],
[<AxesSubplot:title={'center':'critic_score'}>,
 <AxesSubplot:title={'center':'critic_count'}>,
 <AxesSubplot:title={'center':'user_score'}>],
[<AxesSubplot:title={'center':'user_count'}>, <AxesSubplot:>,
 <AxesSubplot:>]], dtype=object)
```



0.7.2 Categóricas

[9]:
```
df[vard].describe()
```

[9]:

	platform	genre	publisher	rating	name	developer
count	6811	6811	6811	6811	6811	6811
unique	16	12	262	7	4365	1289
top	PS2	Action	Electronic Arts	T	Madden NFL 07	EA Canada
freq	1140	1630	944	2367	8	149

Por el número de valores únicos, no se considerarán: * name * developer

Frecuencia

[10]:
```
def my_pie_count(df,col,title=""):
    aux=pd.DataFrame(df[col].value_counts()).reset_index().
 ↪rename(columns={"index":"conteo"})
    fig=aux.
 ↪iplot(kind='pie',labels='conteo',values=col,title=title,asFigure=True,theme="white")
    return fig
```

[11]:
```
for v in vard[:-2]:
    display(my_pie_count(df,v,v))
```

Normalización

[12]:
```
def normalizar(df,v,umbral=0.05):
    aux = df[v].value_counts(1).to_frame().reset_index()
    aux['mapa'] = np.where(aux[v]<umbral,'OTROS',aux['index'])
    mas_grande = aux.head(1)['index'].values[0]
    if aux.loc[aux['mapa']=='OTROS'][v].sum()<umbral:
        aux['mapa'] = aux['mapa'].replace({'OTROS':mas_grande})
    return v,dict(aux[['index','mapa']].values.tolist())
```

[13]:
```
mapa_norm = list(map(lambda v:normalizar(df,v),vard[:-2]))
```

[14]:
```
for v,d in mapa_norm:
    df[f'n_{v}'] = df[v].replace(d)
```

[15]:
```
varn = df.filter(like='n_').columns.tolist()
```

[16]:
```
df[varn].describe()
```

[16]:

	n_platform	n_genre	n_publisher	n_rating
count	6811	6811	6811	6811
unique	10	9	4	4
top	OTROS	Action	OTROS	T
freq	1147	1630	4880	2370

0.8 Extremos

[17]:
```
#Buscando extremos segun percentiles 1 y 99
for v,li,ls in df[varc].describe(percentiles=[0.01,0.99]).T[['1%','99%']].
 ↪reset_index().values:
    df[f'ex_{v}'] = ((df[v]<li)|(df[v]>ls)).astype(int)
df['ex_'] = df.filter(like='ex_').max(axis=1)
```

[18]:
```
df['ex_'].value_counts(True)
#1 significa que es valor extremo
```

[18]:
```
0    0.933196
1    0.066804
Name: ex_, dtype: float64
```

[19]:
```
#Borrando extremos
df = df.loc[df['ex_']==0].reset_index(drop=True).drop(df.filter(like='ex_').
 ↪columns,axis=1)
```

0.9 Varianza Baja

[20]:
```
vt = VarianceThreshold(threshold=0.1)
vt.fit(df[varc])
```

[20]:
```
VarianceThreshold(threshold=0.1)
```

[21]:
```
varianza_peq = [v for v,nu in zip(varc,vt.get_support()) if not nu]
varianza_peq
```

[21]:
```
['eu_sales', 'jp_sales', 'other_sales']
```

[22]:
```
varc = [v for v in varc if v not in varianza_peq]
df.drop(varianza_peq,axis=1,inplace=True)
```

0.10 Multicolinealidad

[23]:
```
vc = VarClusHi(df=df,feat_list=varc)
vc.varclus()
rs = vc.rsquare
rs = rs.sort_values(by=['Cluster','RS_Ratio']).reset_index(drop=True)
rs['id'] = rs.groupby('Cluster').cumcount()+1
rs
```

[23]:

	Cluster	Variable	RS_Own	RS_NC	RS_Ratio	id
0	0	na_sales	0.957161	0.097049	4.744313e-02	1
1	0	global_sales	0.957161	0.148158	5.028961e-02	2
2	1	user_score	0.766536	0.068925	2.507463e-01	1
3	1	critic_score	0.766536	0.157168	2.769989e-01	2
4	2	user_count	0.686188	0.084277	3.426927e-01	1
5	2	critic_count	0.686188	0.113745	3.540874e-01	2
6	3	year_of_release	1.000000	0.078028	2.408365e-16	1

Se observa que se tienen realmente sólo cuatro dimensiones: * Las ventas * Las calificaciones * El número de críticas dadas * El año de publicación

[24]:
```
var_reduced = sorted(rs.loc[rs['id']==1]['Variable'])
```

[25]:
```
var_reduced
```

[25]: ['na_sales', 'user_count', 'user_score', 'year_of_release']

0.11 Correlación

[26]:
```
#Todas la continuas
df[varc].corr().iplot(kind="heatmap",colorscale='spectral')
```

[27]:
```
#Variables que no tienen multicolinealidad
df[var_reduced].corr().iplot(kind="heatmap",colorscale='spectral')
```

#

Previzualización en Espacio Reducido

0.12 Cambio de espacios

[28]:
```
# Nos quedamos con las variables continuas que no tienen multicolinealidad
X = df[var_reduced].sample(1000, random_state=42).reset_index(drop=True)
df_sample = df.sample(1000, random_state=42).reset_index(drop=True)
```

0.12.1 PCA $\mathcal{X} \to \mathcal{X}_p$

[29]:
```
sc = StandardScaler()
pca = PCA(n_components=3)
Xp = pd.DataFrame(pca.fit_transform(sc.fit_transform(X)))
print(pca.explained_variance_ratio_.cumsum())
Xp
```

```
[0.35739883 0.66874017 0.87200323]
```

[29]:

	0	1	2
0	-1.283642	-0.328662	0.066393
1	1.723013	-0.911155	0.323300
2	1.127463	-0.397752	-1.216471
3	-0.660271	0.133089	0.369320
4	-0.647706	0.377395	-0.333292
..	…	…	…
995	-0.013410	2.187113	2.250715
996	1.756697	-0.365954	-1.103841
997	-1.396537	-0.297954	0.282960
998	-2.048679	0.062540	0.664543

```
999 -0.231619 -0.416373  0.677337

[1000 rows x 3 columns]
```

0.12.2 MDS $\mathcal{X} \to \mathcal{X}_m$

[30]:
```
sc = MinMaxScaler()
mds = MDS(n_components=3,n_jobs=-1)
Xm = pd.DataFrame(mds.fit_transform(sc.fit_transform(X)))
Xm
```

[30]:
```
            0         1         2
0    0.100138  0.108551 -0.306903
1   -0.085969 -0.349088  0.317549
2    0.204417 -0.145949  0.232096
3   -0.120136  0.064121 -0.087155
4    0.039266  0.129997 -0.096851
..        …         …         …
995 -0.395482  0.423687 -0.249313
996  0.134405 -0.236225  0.370528
997  0.057831  0.116557 -0.329307
998 -0.008885  0.226212 -0.459738
999 -0.149345 -0.072278 -0.033854

[1000 rows x 3 columns]
```

0.12.3 t-SNE $\mathcal{X} \to \mathcal{X}_t$

[31]:
```
sc = MinMaxScaler()
tsne = TSNE(n_components=3,n_jobs=-1,perplexity=15)
Xt = pd.DataFrame(tsne.fit_transform(sc.fit_transform(X)))
Xt
```

[31]:
```
             0          1         2
0    -7.579919  -9.031008  2.350855
1    11.410842   2.717801  6.913116
2     3.123317   0.952022  5.816360
3    -3.114307   4.243768 -4.993948
4    -6.112155  -2.074870 -1.407376
..         …          …         …
995   1.958578 -13.326956  2.080255
996   6.320858   0.297090  5.798705
997  -7.428191 -10.290636  1.385277
998  -3.242615 -13.954642 -0.403981
999   2.824702   7.565819 -6.850651
```

```
[1000 rows x 3 columns]
```

0.13 Visualización preeliminar

0.13.1 Vectores

[32]:
```
Xp.iplot(kind='scatter3d',x=0,y=1,z=2,mode='markers',color='purple')
```

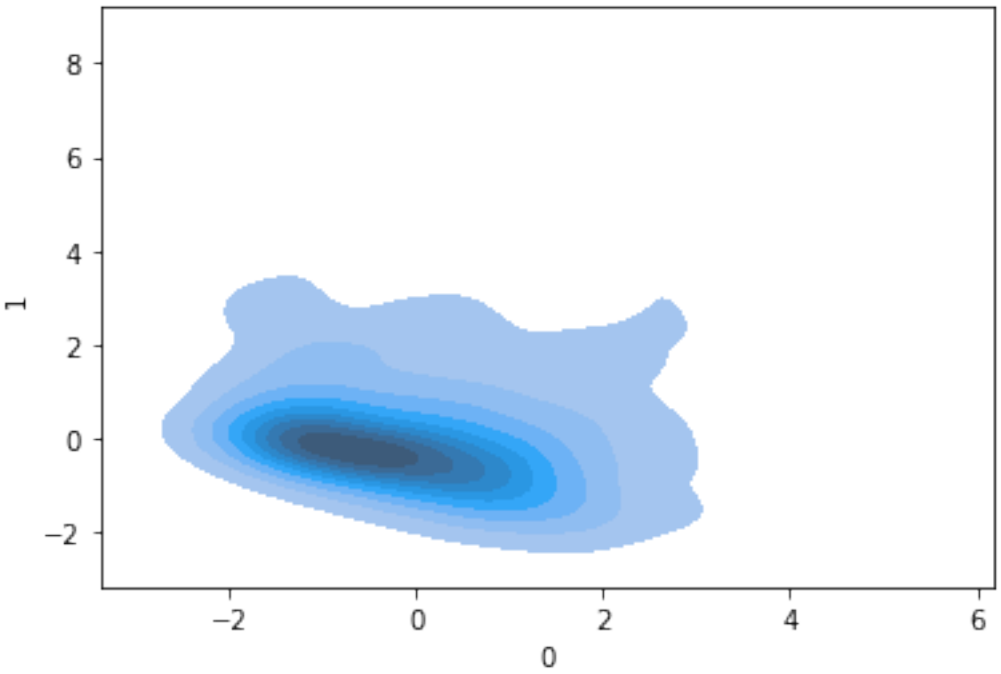
[33]:
```
Xm.iplot(kind='scatter3d',x=0,y=1,z=2,mode='markers',color='purple')
```

[34]:
```
Xt.iplot(kind='scatter3d',x=0,y=1,z=2,mode='markers',color='purple')
```

0.13.2 Densidad

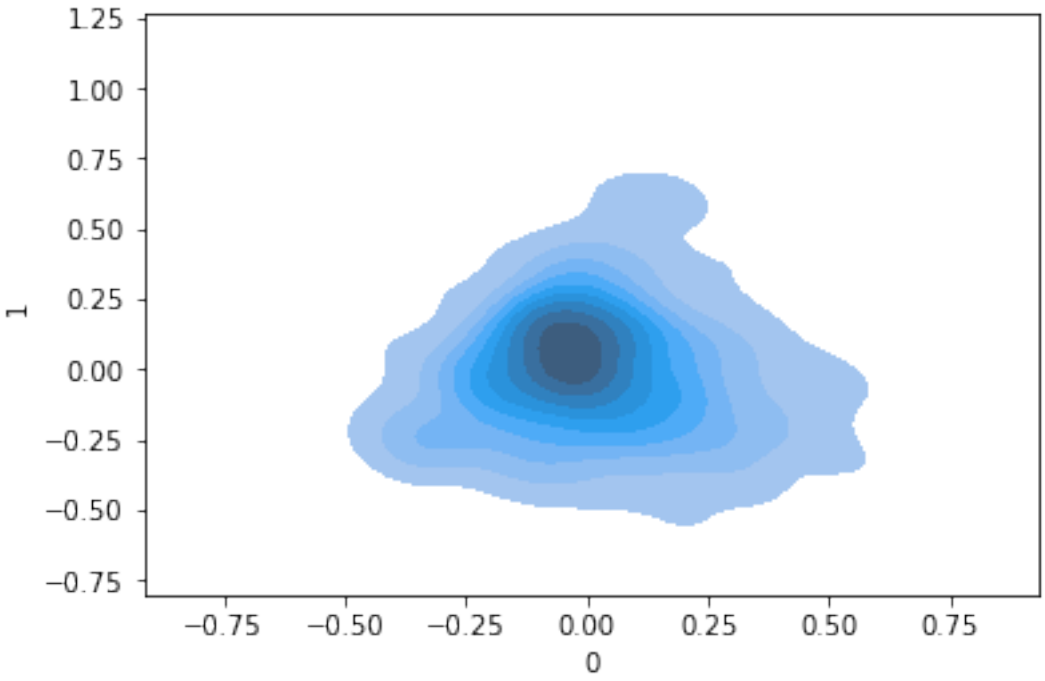
[35]:
```
sns.kdeplot(data=Xp,x=0,y=1,fill=True)
```

[35]:
```
<AxesSubplot:xlabel='0', ylabel='1'>
```



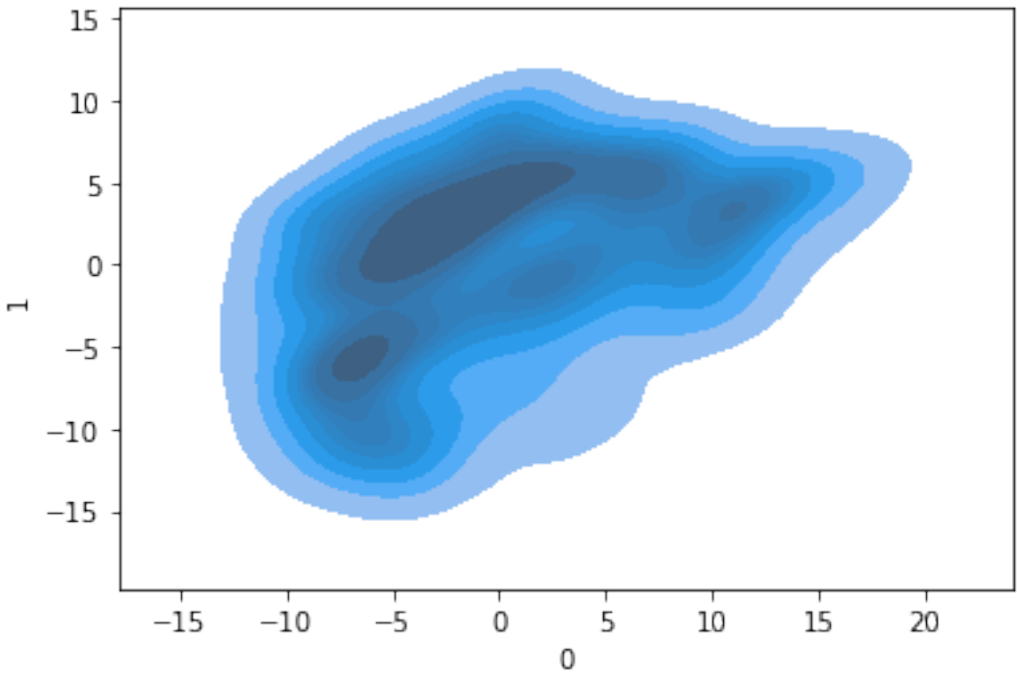
[36]:
```
sns.kdeplot(data=Xm,x=0,y=1,fill=True)
```

[36]:
```
<AxesSubplot:xlabel='0', ylabel='1'>
```



[37]:
```
sns.kdeplot(data=Xt,x=0,y=1,fill=True)
```

[37]: <AxesSubplot:xlabel='0', ylabel='1'>



No se aprecia facilmente la presencia de clusters claramente diferenciados. Al parecer todos los datos seconcentran en un único cúmulo , sin embargo, la gráfica de densidad con el método t-SNE podría indicar laexistencia de al menos tres grupos.

#

Clustering

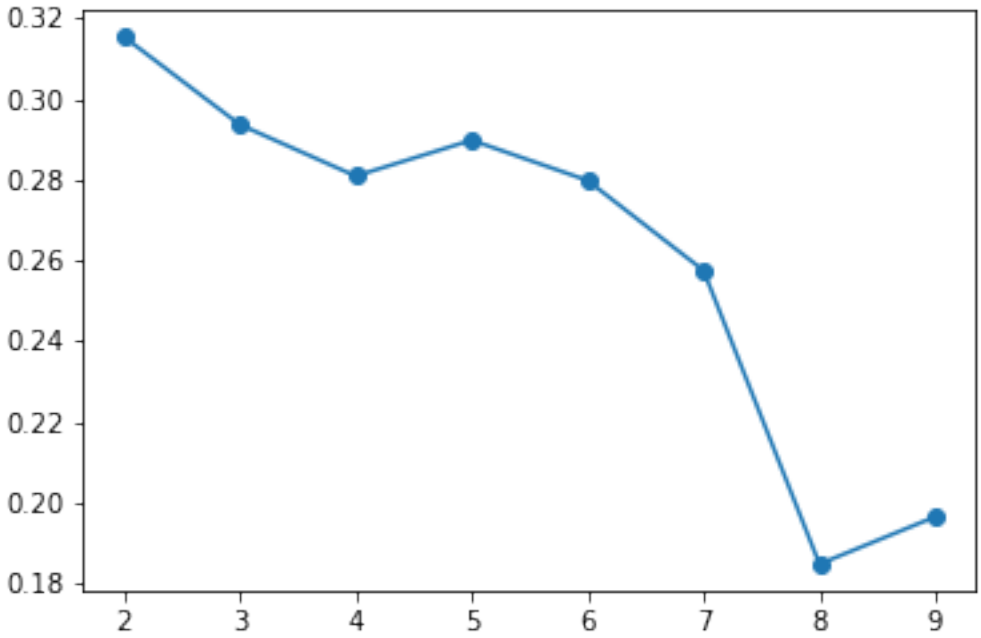
---

[38]:
```
sc = MinMaxScaler()
Xs = pd.DataFrame(sc.fit_transform(X),columns=var_reduced)
```

0.14 Aglomerativo

[39]:
```
sil = pd.DataFrame(map(lambda k:(k,silhouette_score(Xs,
                                                    ␣
 ↪AgglomerativeClustering(n_clusters=k).fit_predict(Xs))),
                  range(2,10)),columns=['k','sil'])
plt.plot(sil['k'],sil['sil'],marker='o')
```

[39]: `[<matplotlib.lines.Line2D at 0x7ff036dc9610>]`

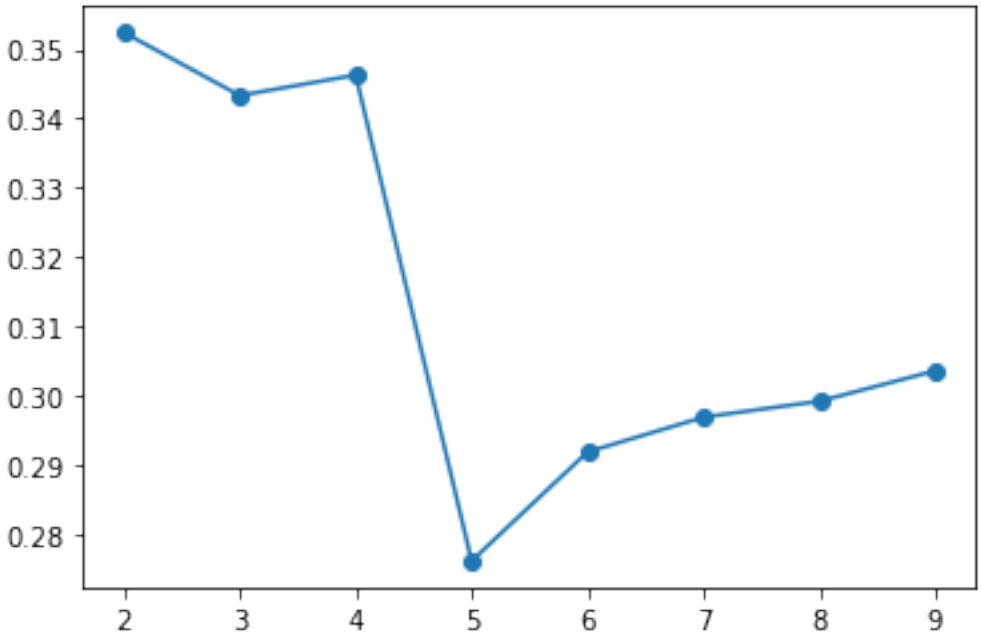


[40]:
```
k = 3
tipo = 'agg'
agg = AgglomerativeClustering(n_clusters=k)
df_sample[f'cl_{tipo}']=Xp[f'cl_{tipo}']=Xm[f'cl_{tipo}']=Xt[f'cl_{tipo}'] =agg.
 ↪fit_predict(Xs[var_reduced])
```

0.15 K-medias

[41]:
```
sil = pd.DataFrame(map(lambda k:(k,silhouette_score(Xs,
                                                    ␣
 ↪KMeans(n_clusters=k,max_iter=1000).fit_predict(Xs))),
                  range(2,10)),columns=['k','sil'])
plt.plot(sil['k'],sil['sil'],marker='o')
```

[41]:
```
[<matplotlib.lines.Line2D at 0x7ff036e23820>]
```

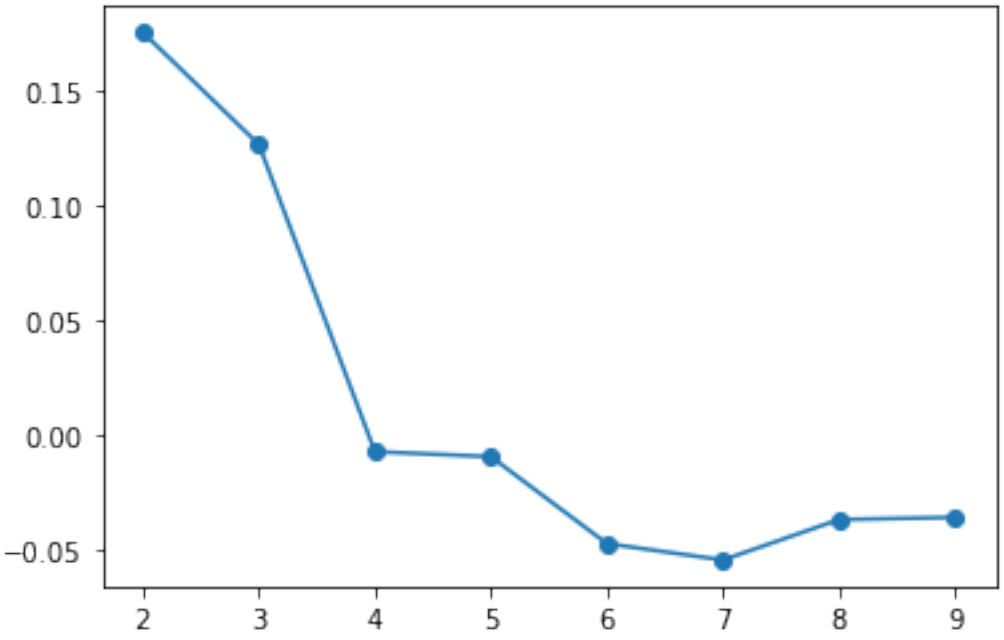


[42]:
```
k = 4
tipo = 'kme'
kme = KMeans(n_clusters=k,max_iter=1000)
df_sample[f'cl_{tipo}']=Xp[f'cl_{tipo}']=Xm[f'cl_{tipo}']=Xt[f'cl_{tipo}'] =kme.
 ↪fit_predict(Xs[var_reduced])
```

0.16 Modelos Gaussianos Mixtos

[43]:
```
sil = pd.DataFrame(map(lambda k:(k,silhouette_score(Xs,
                                                    ␣
 ↪GaussianMixture(n_components=k,max_iter=1000).fit_predict(Xs))),
                  range(2,10)),columns=['k','sil'])
plt.plot(sil['k'],sil['sil'],marker='o')
```

[43]:
```
[<matplotlib.lines.Line2D at 0x7ff036b28a00>]
```



[44]:
```
k = 4
tipo = 'gmm'
gmm = GaussianMixture(n_components=k,max_iter=1000)
df_sample[f'cl_{tipo}']=Xp[f'cl_{tipo}']=Xm[f'cl_{tipo}']=Xt[f'cl_{tipo}'] =gmm.
 ↪fit_predict(Xs[var_reduced])
```

0.17 Selección final

[45]:
```
varcl = sorted(df_sample.filter(like='cl_'))
for v in varcl:
    Xp[v] = Xp[v].astype(str)
    Xm[v] = Xm[v].astype(str)
    Xt[v] = Xt[v].astype(str)
    df_sample[v] = df_sample[v].astype(str)
```

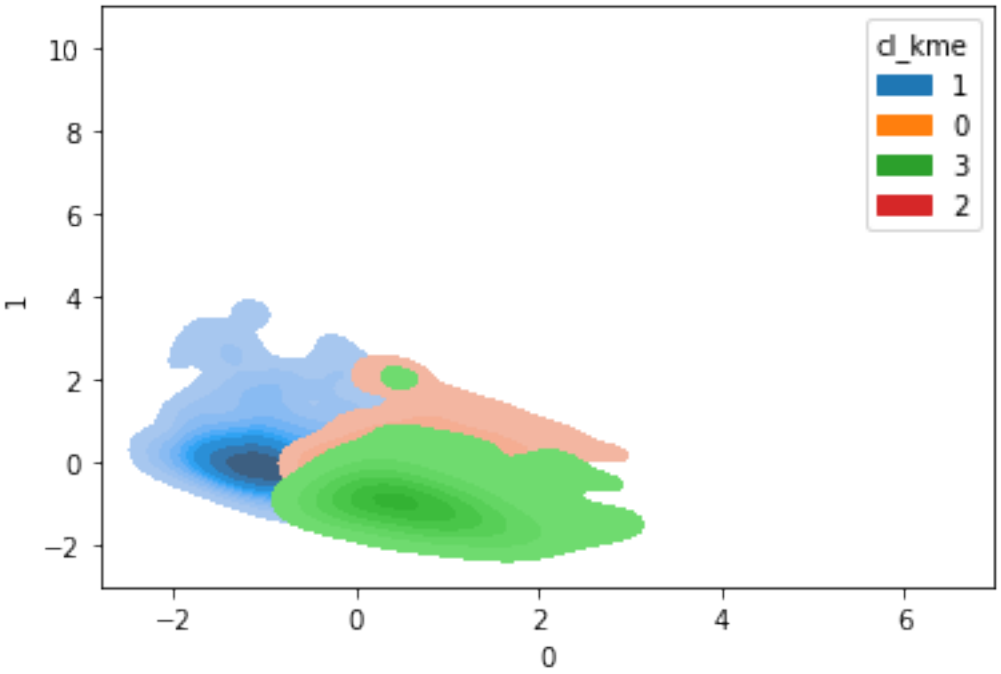
```
pd.DataFrame(map(lambda cl:
 ↪(cl,silhouette_score(Xs,df_sample[cl])),varcl),columns=['cluster','sil']).
 ↪iplot(kind='bar',categories='cluster')
```

Se selecciona K-means

0.18 Visualización con cluster

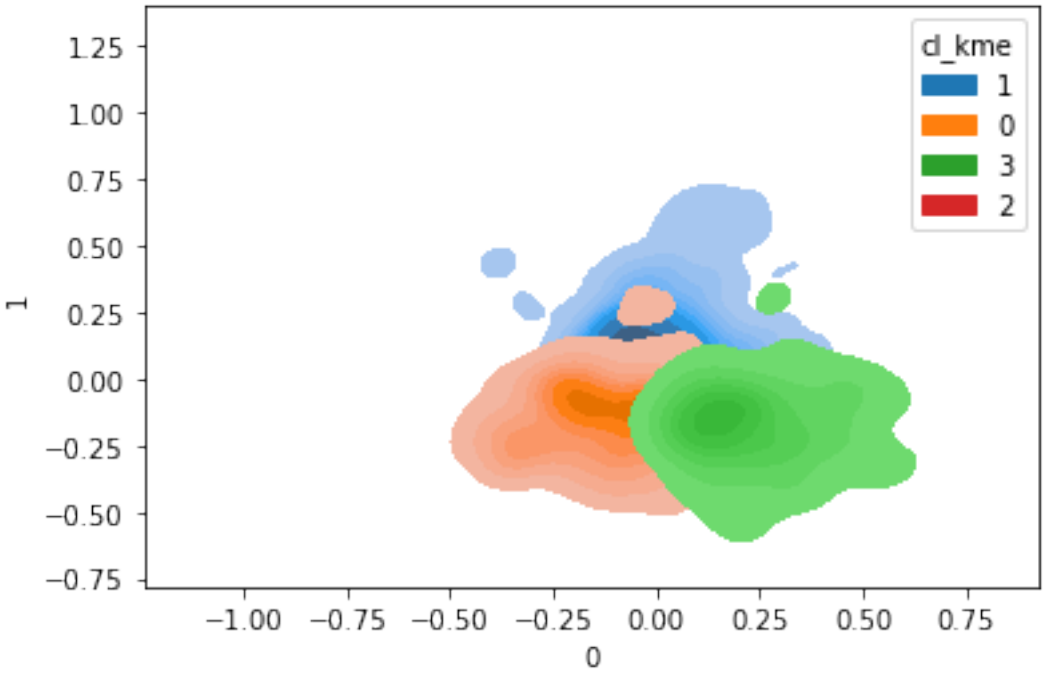
[46]:
```
Xp.iplot(kind='scatter3d',x=0,y=1,z=2,mode='markers',categories='cl_kme')
sns.kdeplot(data=Xp,x=0,y=1,fill=True,hue='cl_kme')
```

[46]:
```
<AxesSubplot:xlabel='0', ylabel='1'>
```



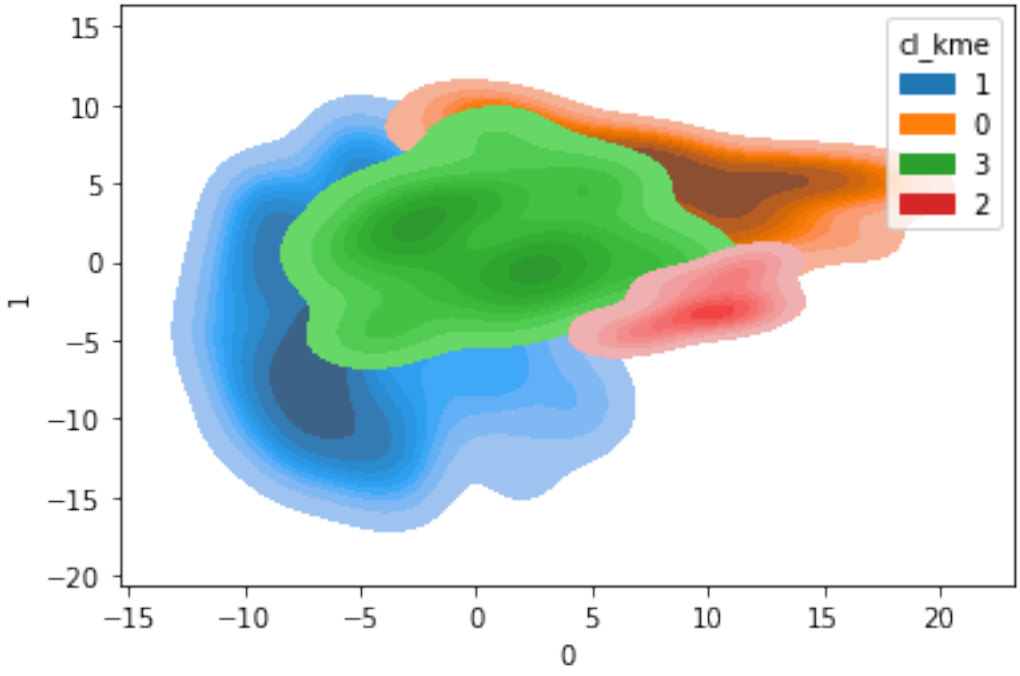
[47]:
```
Xm.iplot(kind='scatter3d',x=0,y=1,z=2,mode='markers',categories='cl_kme')
sns.kdeplot(data=Xm,x=0,y=1,fill=True,hue='cl_kme')
```

[47]:
```
<AxesSubplot:xlabel='0', ylabel='1'>
```



[48]:
```
Xt.iplot(kind='scatter3d',x=0,y=1,z=2,mode='markers',categories='cl_kme')
sns.kdeplot(data=Xt,x=0,y=1,fill=True,hue='cl_kme')
```

[48]: <AxesSubplot:xlabel='0', ylabel='1'>



0.19 Perfilamiento

0.19.1 Continuo

[49]:
```
for v in var_reduced:
    print(v)
    display(MultiComparison(df_sample[v],df_sample['cl_kme']).tukeyhsd().
 ↪summary())
    plt.figure()
    sns.boxplot(data=df_sample,y=v,x='cl_kme')
```

```
na_sales
```

```
<class 'statsmodels.iolib.table.SimpleTable'>
```

```
user_count
```

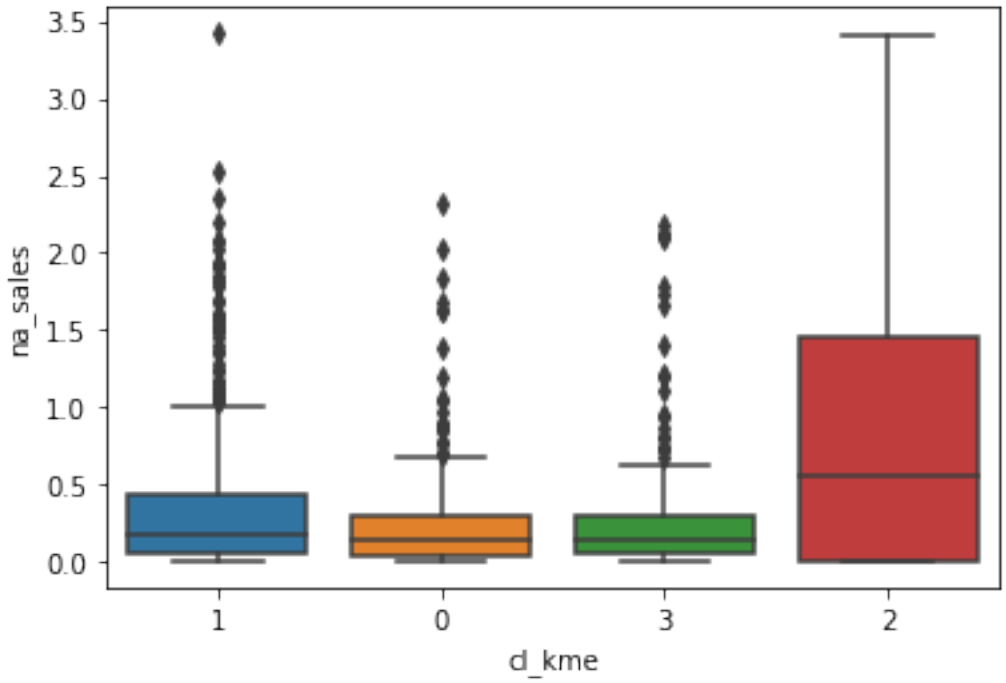
```
<class 'statsmodels.iolib.table.SimpleTable'>
```

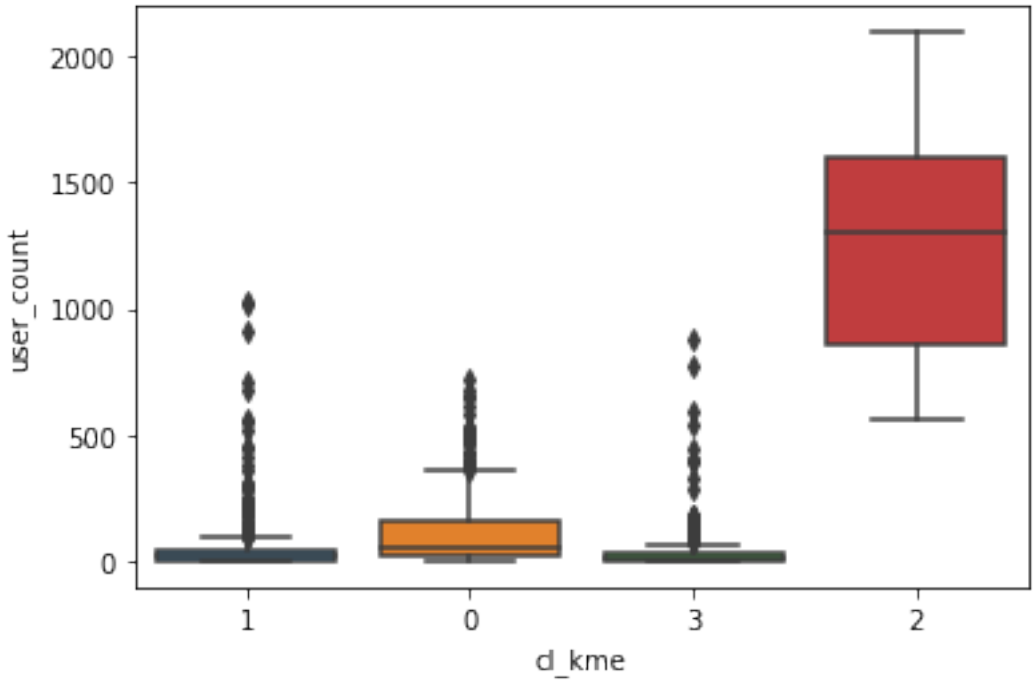
```
user_score
```

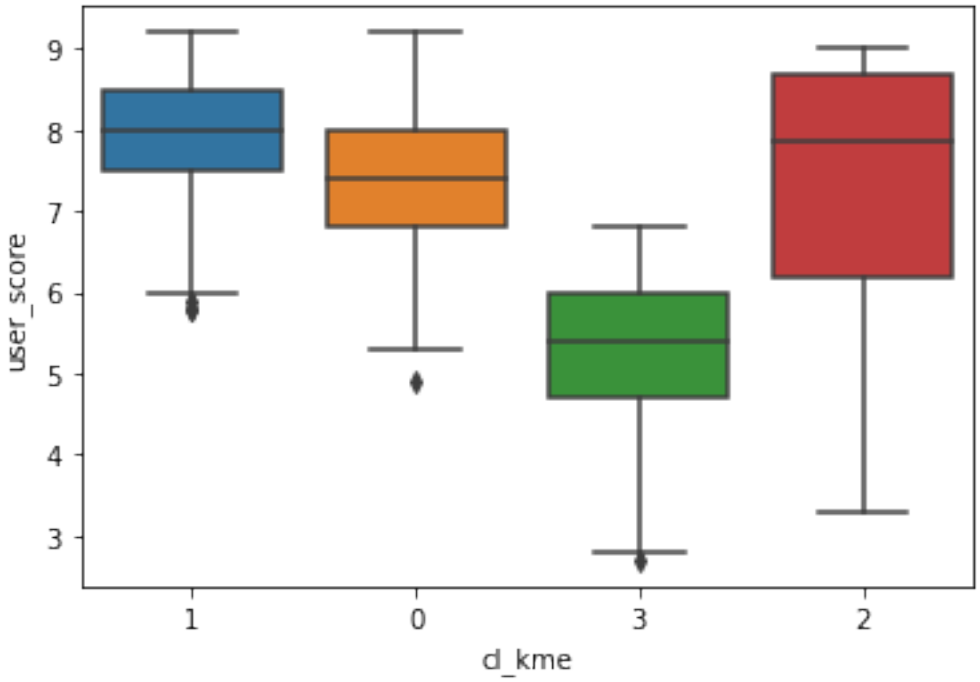
```
<class 'statsmodels.iolib.table.SimpleTable'>
```

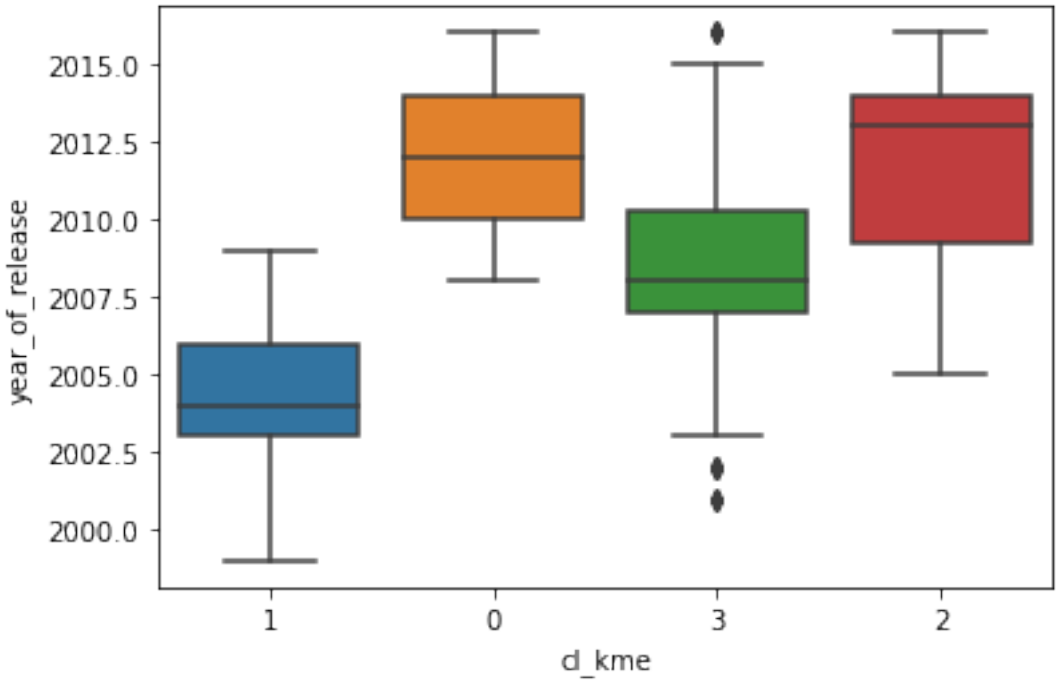
```
year_of_release
```

```
<class 'statsmodels.iolib.table.SimpleTable'>
```




2000
1500
1000
500
0
user_count
1
0
3
2
d_kme


9
8
7
6
5
4
3
user_score
1
0
3
2
d_kme



0.19.2 Discreto

[50]:
```
for v in varn:
    piv = df_sample.pivot_table(index='cl_kme',
                      columns=v,
                      values='user_score',
                      aggfunc='count',
                      fill_value=0)
    observadas = [[np.round(x/sum(l)*100,0) for x in l] for l in piv.values]

    esperada = list(np.round((df[v].value_counts().sort_index()/len(df)*100),0))

    display(pd.DataFrame(enumerate(map(lambda obs:np.
↪round(chisquare(f_exp=esperada,
                                                                    f_obs=obs).
↪pvalue,4),observadas)),columns=['cluster','p-value']).assign(variable=v))

    piv['t'] = piv.sum(axis=1)
    for c in piv.columns:
        if c!='t':
            piv[c]/=piv['t']

    piv.drop('t',axis=1,inplace=True)
```

```
    esp = pd.DataFrame({'total':esperada}).T
    esp.columns = piv.columns

    pd.concat([piv.apply(lambda x:np.round(x*100,0)),esp]).
↪iplot(kind='bar',barmode='stack')
```

```
   cluster  p-value    variable
0        0   0.0000  n_platform
1        1   0.0001  n_platform
2        2   0.0000  n_platform
3        3   0.0000  n_platform
```

```
   cluster  p-value variable
0        0   0.2733  n_genre
1        1   0.8331  n_genre
2        2   0.0000  n_genre
3        3   0.3325  n_genre
```

```
   cluster  p-value     variable
0        0   0.3797  n_publisher
1        1   0.9489  n_publisher
2        2   0.2644  n_publisher
3        3   0.6349  n_publisher
```

```
   cluster  p-value  variable
0        0   0.0912  n_rating
1        1   0.4783  n_rating
2        2   0.0000  n_rating
3        3   0.3927  n_rating
```

0.20 Arquetipos

[51]:
```
display(df_sample[var_reduced+['cl_kme']].groupby('cl_kme').mean())
display(df_sample[var_reduced].mean().to_frame().T)
```

```
        na_sales   user_count  user_score  year_of_release
cl_kme
0       0.246207   122.509579    7.329119      2011.988506
1       0.345287    52.887129    7.965149      2004.388119
2       0.900263  1245.684211    7.271053      2011.763158
3       0.270357    53.938776    5.285714      2008.494898
```

```
   na_sales  user_count  user_score  year_of_release
0   0.32583     116.591      7.2476         2007.457
```

Se observa que el año es la variable que más diferencias marca entre todos los grupos, a excepción del cluster3 y 1. Es por esto mismo, que la única variable categórica que puede dividir efectivamente en clusters aldataset es la de las plataformas, ya que cada consola de videojuegos tiene un periodo de vida respecto a lapublicación de videojuegos.

- Cluster 0 : Videojuegos publicados alrededor del año 2008, con calificaciones bajas (5.2 en promedio),bajo número de críticas (54 en promedio) y el más bajo promedio de ventas (0.27 millones de copias).De forma general, este cluster representa a juegos con muy poco éxito.
- Cluster 1 : Videojuegos publicados alrededor del año 2011, con calificaciones medias (7.1 en promedio),número medio de críticas (119 en promedio) pero ventas bajas (0.24 millones de copias), las más bajasde entre los clusters de hecho. Este cluster se compone de los juegos más nuevos del dataset, que apesar de tener buenas calificaciones, resultaron ser un fracaso en ventas.
- Cluster 2 : Videojuegos publicados alrededor del año 2004, con calificaciones muy altas (8.0 en prome-dio), pero con bajo número de críticas y de ventas. Este cluster representa juegos de años tempranosque recibieron muy altas calificaciones, pero no una base amplia de jugadores, ya haya sido por un bajoéxito de ventas o porque el mercado de los videojuegos no era tan grande como hoy en día.
- Cluster 3 : Videojuegos publicados alrededor del 2011, con calificaciones medias (7.3 en promedio) yun gran número de críticas por parte de usuarios junto con ventas. Este cluster representa el de losvideojuegos más exitosos en ventas y en popularidad. Son juegos que a pesar de no destacar mucho ensus calificaciones, pudieron asegurar una amplia base de fans.

Debido a que la variable de año de publicación tiene una influencia tan grande en la formación de grupos,se decide volver a clusterizar los datos omitiendo dicha variable, para intentar obtener una mejor agrupaciónen función del éxito de los juegos.

0.21 Gráfico Radial

[52]:
```
Xs['cl_kme'] = df_sample['cl_kme']
```

[53]:
```
aux = Xs[var_reduced+['cl_kme']].groupby('cl_kme').mean()
aux
```

[53]:

cl_kme	na_sales	user_count	user_score	year_of_release
0	0.071990	0.056839	0.712172	0.764030
1	0.100961	0.023447	0.810023	0.316948
2	0.263235	0.595532	0.703239	0.750774
3	0.079052	0.023951	0.397802	0.558523

[54]:
```
fig = go.Figure()

for i,row in aux.iterrows():
    fig.add_trace(go.Scatterpolar(r=row.values,
                                  theta=var_reduced,
                                  fill='toself',
                                  name=f'cluster {i}'))
fig.show()
```

#

Clustering excluyendo la variable de Año

---

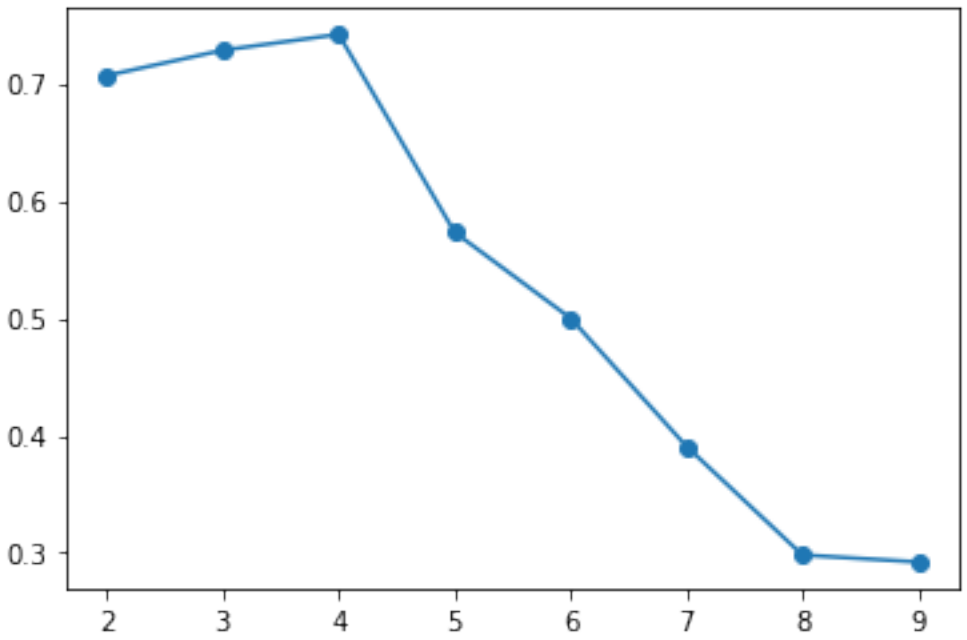
[55]:
```
var_reduced = var_reduced[:-1]
var_reduced
```

[55]:
```
['na_sales', 'user_count', 'user_score']
```

0.21.1 Aglomerativo

[56]:
```
sil = pd.DataFrame(map(lambda k:(k,silhouette_score(Xs,
                                                    ␣
 ↪AgglomerativeClustering(n_clusters=k).fit_predict(Xs))),
                  range(2,10)),columns=['k','sil'])
plt.plot(sil['k'],sil['sil'],marker='o')
```

[56]:
```
[<matplotlib.lines.Line2D at 0x7fefd02898e0>]
```



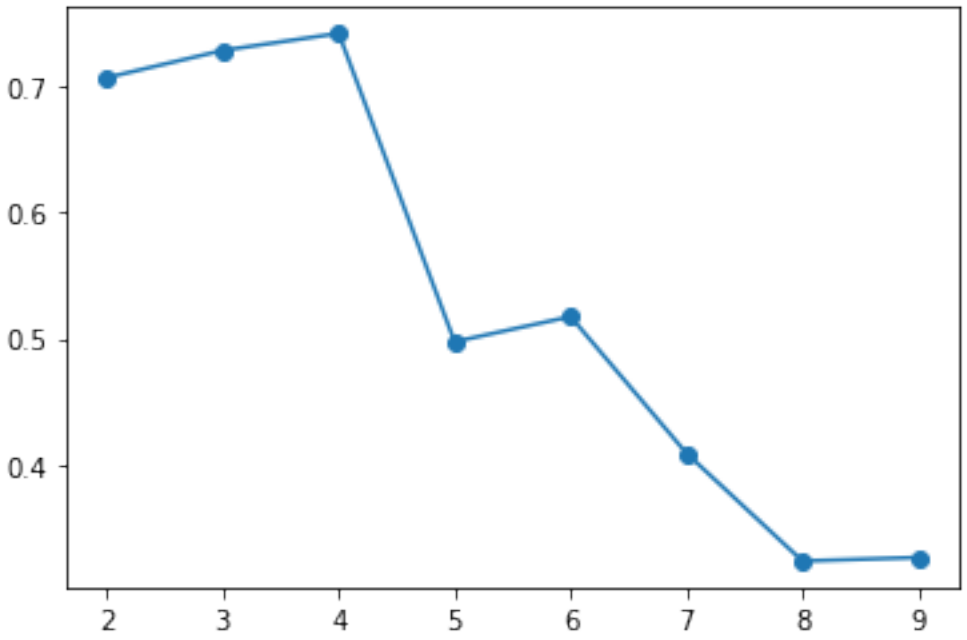
[57]:
```
k = 4
tipo = 'agg'
agg = AgglomerativeClustering(n_clusters=k)
```

```
df_sample[f'cl_{tipo}']=Xp[f'cl_{tipo}']=Xm[f'cl_{tipo}']=Xt[f'cl_{tipo}'] =agg.
↪fit_predict(Xs[var_reduced])
```

0.21.2 K-medias

[58]:
```
sil = pd.DataFrame(map(lambda k:(k,silhouette_score(Xs,
                                                    ␣
↪KMeans(n_clusters=k,max_iter=1000).fit_predict(Xs))),
                   range(2,10)),columns=['k','sil'])
plt.plot(sil['k'],sil['sil'],marker='o')
```

[58]:
```
[<matplotlib.lines.Line2D at 0x7fefd0323490>]
```



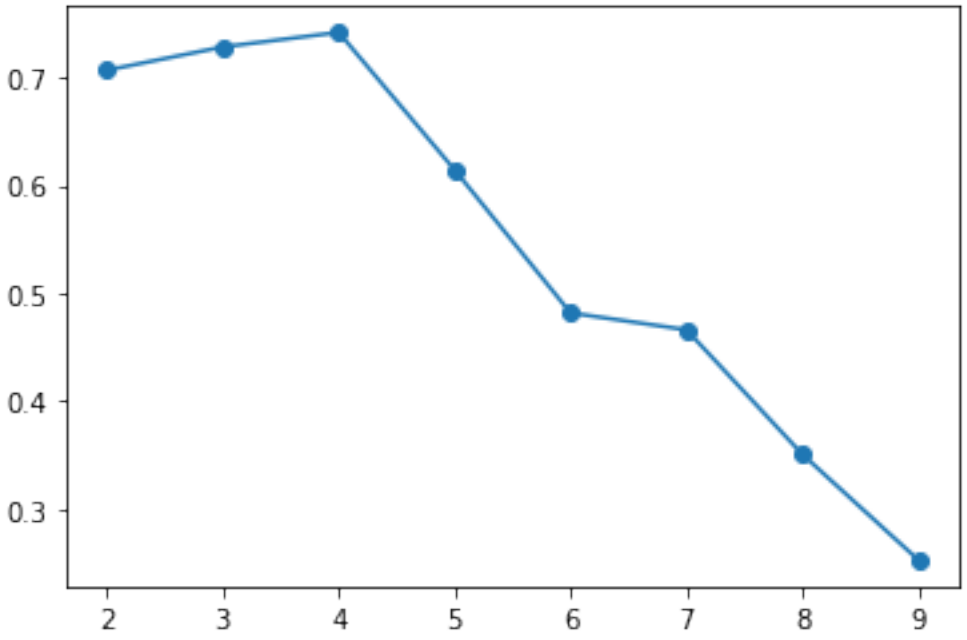
[59]:
```
k = 4
tipo = 'kme'
kme = KMeans(n_clusters=k,max_iter=1000)
df_sample[f'cl_{tipo}']=Xp[f'cl_{tipo}']=Xm[f'cl_{tipo}']=Xt[f'cl_{tipo}'] =kme.
↪fit_predict(Xs[var_reduced])
```

0.21.3 Modelos Gaussianos Mixtos

[60]:
```
sil = pd.DataFrame(map(lambda k:(k,silhouette_score(Xs,
                                                    ␣
↪GaussianMixture(n_components=k,max_iter=1000).fit_predict(Xs))),
                   range(2,10)),columns=['k','sil'])
```

```
plt.plot(sil['k'],sil['sil'],marker='o')
```

[60]: [<matplotlib.lines.Line2D at 0x7ff036e941c0>]



[61]:
```
k = 4
tipo = 'gmm'
gmm = GaussianMixture(n_components=k,max_iter=1000)
df_sample[f'cl_{tipo}']=Xp[f'cl_{tipo}']=Xm[f'cl_{tipo}']=Xt[f'cl_{tipo}'] =gmm.
 ↪fit_predict(Xs[var_reduced])
```

0.21.4 Selección final

[62]:
```
varcl = sorted(df_sample.filter(like='cl_'))
for v in varcl:
    Xp[v] = Xp[v].astype(str)
    Xm[v] = Xm[v].astype(str)
    Xt[v] = Xt[v].astype(str)
    df_sample[v] = df_sample[v].astype(str)

pd.DataFrame(map(lambda cl:
 ↪(cl,silhouette_score(Xs,df_sample[cl])),varcl),columns=['cluster','sil']).
 ↪iplot(kind='bar',categories='cluster')
```

Se selecciona el agrupamiento realizado por el módelo Agrupamiento por su valor en el estadígrafo de silueta.

0.22 Gráfico Radial

[63]:
```
Xs['cl_agg'] = df_sample['cl_agg']
```

[64]:
```
aux = Xs[var_reduced+['cl_agg']].groupby('cl_agg').mean()
aux
```

[64]:

cl_agg	na_sales	user_count	user_score
0	0.070674	0.027614	0.793755
1	0.064584	0.034079	0.419118
2	0.099334	0.559153	0.720085
3	0.571111	0.141736	0.755077

[65]:
```
fig = go.Figure()

for i,row in aux.iterrows():
    fig.add_trace(go.Scatterpolar(r=row.values,
                                  theta=var_reduced,
                                  fill='toself',
                                  name=f'cluster {i}'))
fig.show()
```

0.22.1 Arquetipos

- Cluster 0 : Concentra videojuegos con malas ventas y una baja cantidad de críticas, pero buenas cali-ficaciones.
- Cluster 1 : Concentra los videojuegos con el menor éxito, bajas ventas, baja cantidad de críticas y malascalificaciones.
- Cluster 2 : Concentra videojuegos con una enorme cantidad de críticas y muy buenas calificaciones,pero sorpresivamente, bajas ventas.
- Cluster 3 : Concentra videojuegos que tiene un número relativamente bajo de críticas, pero buenascalificaciones y excelenetes ventas.

[ ]:
```

```