analisis

December 10, 2018

Trabajo Final

```
<h2 style="text-align: center; font-weight: bold; font-size: 20px; padding: 10px 20px; backgroun
<div style="width: 450px; margin: 0 auto 30px;">
    <div style="overflow: hidden; border-bottom: 1px solid #d2d2d2;">
        <div style="float: left; width: 300px; padding-left: 8px;">
            <p style="font-family: 'Open Sans', sans-serif; font-size: 18px; font-weight: bold;
        </div>
        <div style="float: left; width: 150px;">
            <p style="font-family: 'Open Sans', sans-serif; font-size: 18px; font-weight: bold;
        </div>
    </div>
    <div style="overflow: hidden; border-bottom: 1px solid #d2d2d2;">
        <div style="float: left; width: 300px; padding-left: 8px;">
            <p style="font-family: 'Open Sans', sans-serif; font-size: 16px; color: #363636; lin
        </div>
        <div style="float: left; width: 150px;">
            <p style="font-family: 'Open Sans', sans-serif; font-size: 16px; color: #363636; lin
        </div>
    </div>
     <div style="overflow: hidden; border-bottom: 1px solid #d2d2d2; ">
        <div style="float: left; width: 300px; padding-left: 8px;">
            <p style="font-family: 'Open Sans', sans-serif; font-size: 16px; color: #363636; lin
        </div>
        <div style="float: left; width: 150px;">
            <p style="font-family: 'Open Sans', sans-serif; font-size: 16px; color: #363636; lin
        </div>
    </div>
```

```
In [1]: import pandas as pd
        import numpy as np
        import matplotlib
        import seaborn as sns
        import sklearn
        from matplotlib import pyplot as plt
```

```
import plotly.offline as py
from plotly import tools
py.init_notebook_mode(connected=True)
import plotly.graph_objs as go
from plotly import tools

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

```
In [2]: apps = pd.read_csv('data/AppleStore.csv')
        desc = pd.read_csv('data/appleStore_description.csv')
```

Con millones de aplicaciones en la actualidad, el siguiente conjunto de datos se ha convertido en la clave para obtener las mejores aplicaciones en la tienda de aplicaciones iOS. Este conjunto de datos contiene más de 7000 detalles de aplicaciones móviles de Apple iOS. Los datos se extrajeron de la API de búsqueda de iTunes en el sitio web de Apple Inc.

Fecha de recolección de datos (de API); Julio 2017

Dimensión del conjunto de datos; 7197 filas y 18 columnas (17 columnas + 1 de descripcion que está en un archivo aparte).

```
In [3]: apps.shape

Out[3]: (7197, 17)

In [4]: apps.dtypes

Out[4]: Unnamed: 0            int64
        id                    int64
        track_name           object
        size_bytes            int64
        currency             object
        price               float64
        rating_count_tot      int64
        rating_count_ver      int64
        user_rating         float64
        user_rating_ver     float64
        ver                  object
        cont_rating          object
        prime_genre          object
        sup_devices.num       int64
        ipadSc_urls.num       int64
        lang.num              int64
        vpp_lic               int64
        dtype: object
```

vpp_lic: *The Apple Volume Purchase Program (VPP) is a service that allows organizations that have registered for the Apple VPP to purchase iOS apps in bulk, but not at discounted prices.*

After making a bulk purchase, the organization receives redemption codes for each app bought. The organization can then distribute app codes to individual users, who use the codes to "purchase" the app from the Apple App Store.

source: https://searchmobilecomputing.techtarget.com/definition/Apple-Volume-Purchase-Program-Apple-VPP

In [5]:
```
# borramos la primer columna, Unnamed: 0, que en el file original funciona como indice
apps.drop('Unnamed: 0', axis=1, inplace=True)
```

In [6]:
```
apps.head()
```

Out[6]:
```
          id                                    track_name  size_bytes  \
0  281656475                               PAC-MAN Premium   100788224
1  281796108                     Evernote - stay organized   158578688
2  281940292   WeatherBug - Local Weather, Radar, Maps, Alerts   100524032
3  282614216  eBay: Best App to Buy, Sell, Save! Online Shop...   128512000
4  282935706                                         Bible    92774400

  currency  price  rating_count_tot  rating_count_ver  user_rating  \
0      USD   3.99             21292                26          4.0
1      USD   0.00            161065                26          4.0
2      USD   0.00            188583              2822          3.5
3      USD   0.00            262241               649          4.0
4      USD   0.00            985920              5320          4.5

   user_rating_ver     ver cont_rating   prime_genre  sup_devices.num  \
0              4.5   6.3.5          4+         Games               38
1              3.5   8.2.2          4+  Productivity               37
2              4.5   5.0.0          4+       Weather               37
3              4.5  5.10.0         12+      Shopping               37
4              5.0   7.5.1          4+     Reference               37

   ipadSc_urls.num  lang.num  vpp_lic
0                5        10        1
1                5        23        1
2                5         3        1
3                5         9        1
4                5        45        1
```

In [7]:
```
# remove repeated columns from description file
desc.drop(['track_name','size_bytes'], axis=1,inplace=True)
```

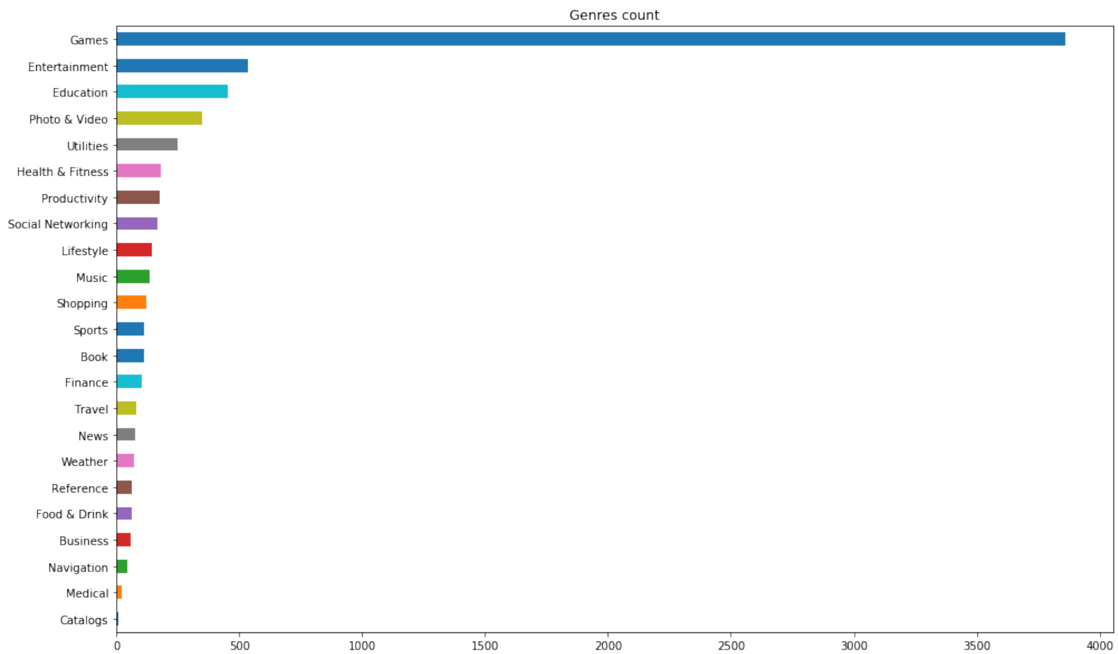
In [8]:
```
desc.head()
```

Out[8]:
```
          id                                           app_desc
0  281656475  SAVE 20%, now only $3.99 for a limited time!\n...
1  281796108  Let Evernote change the way you organize your ...
2  281940292  Download the most popular free weather app pow...
3  282614216  The eBay app is the best way to find anything ...
4  282935706  On more than 250 million devices around the wo...
```

0.1 Análisis de categorias

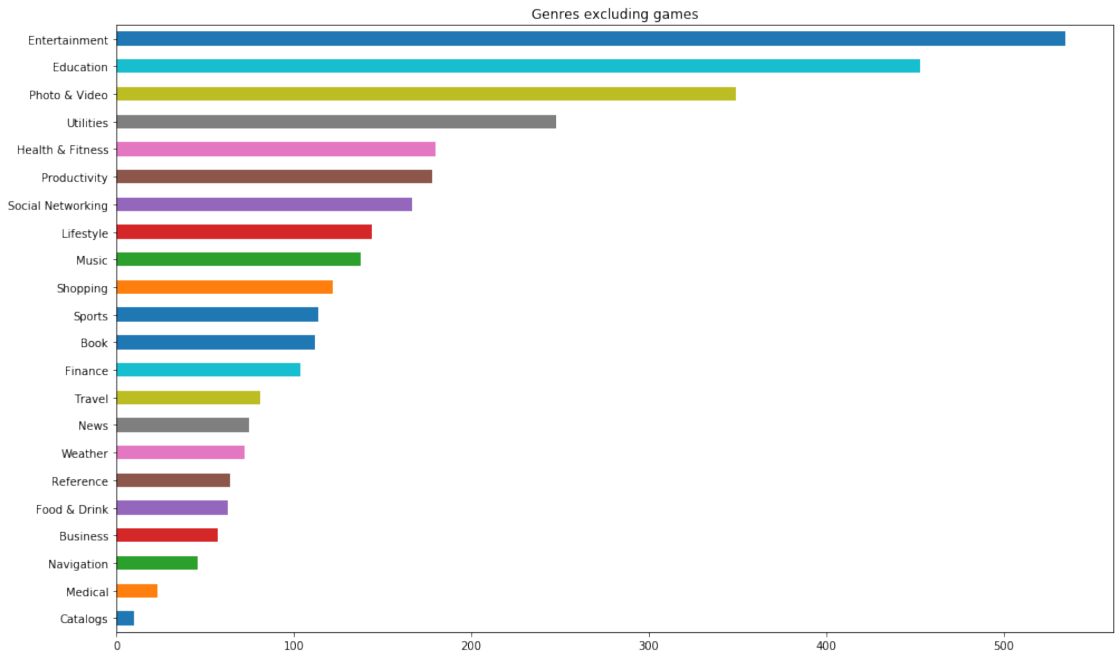
```
In [9]: apps.prime_genre.value_counts(ascending=True).plot(kind='barh', figsize=(16,10)).set_tit
```

```
Out[9]: Text(0.5, 1.0, 'Genres count')
```




```
In [10]: #mismo plot filtrando games
         apps[apps['prime_genre'] != 'Games'].prime_genre.value_counts(ascending=True).plot(kind
```

```
Out[10]: Text(0.5, 1.0, 'Genres excluding games')
```



0.2 Analisis de precios

```
In [11]: apps.currency.value_counts()
```

```
Out[11]: USD    7197
         Name: currency, dtype: int64
```

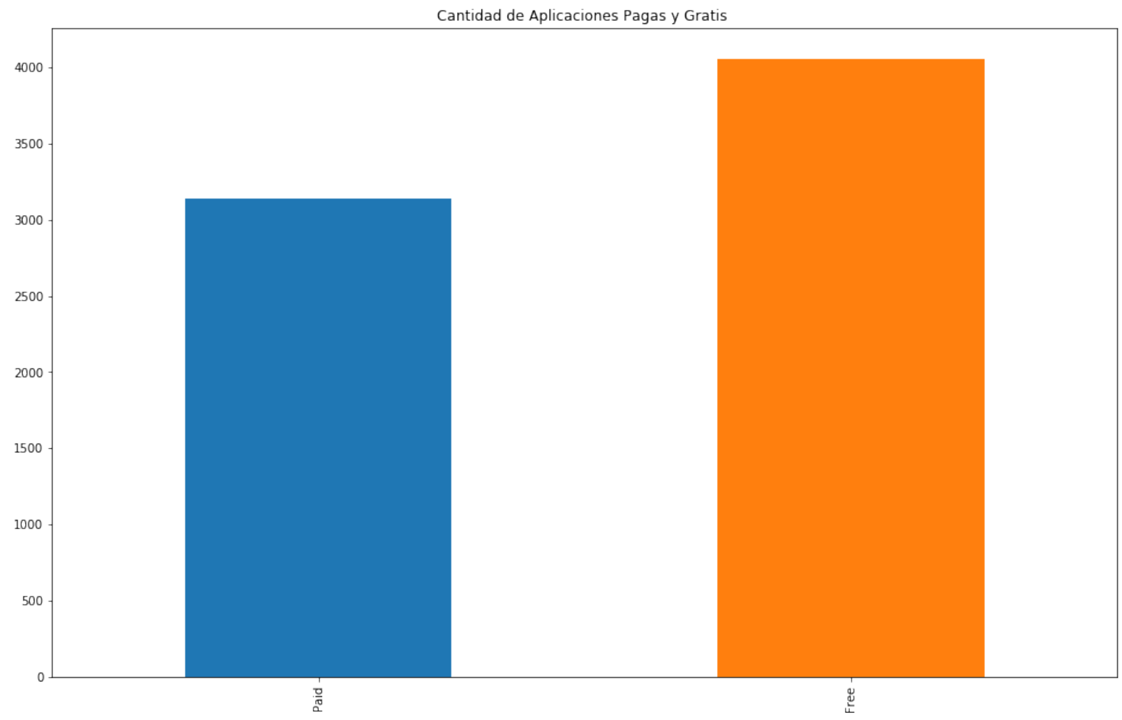
Como todos los precios estan en USD, se puede descartar esta columna.

```
In [12]: apps.drop('currency', axis=1, inplace=True)
```

0.3 Apps pagas vs gratuitas

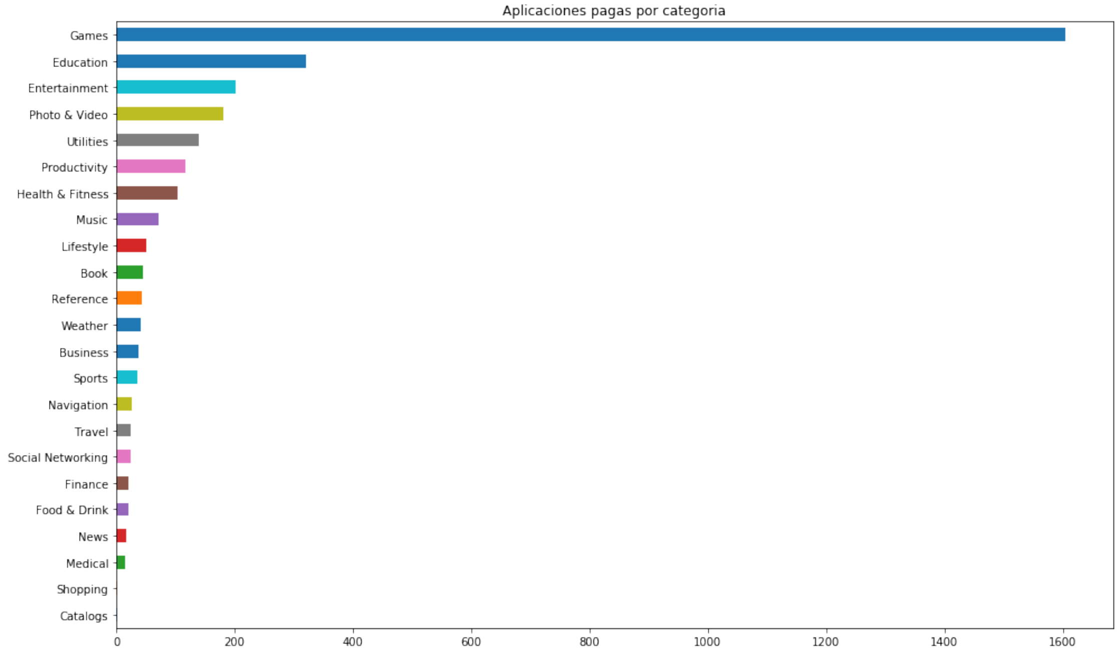
```
In [13]: def paid(x):
             if x>0:
                 return 'Paid'
             else :
                 return'Free'

         apps['category']= apps.price.apply(lambda x : paid(x))

         apps['category'].value_counts(ascending=True).plot(kind='bar', figsize=(16,10)).set_tit
```

```
Out[13]: Text(0.5, 1.0, 'Cantidad de Aplicaciones Pagas y Gratis ')
```




```
In [14]: # plot generos de apps pagas
         apps[apps['price'] > 0].prime_genre.value_counts(ascending=True).plot(kind='barh', figs
```

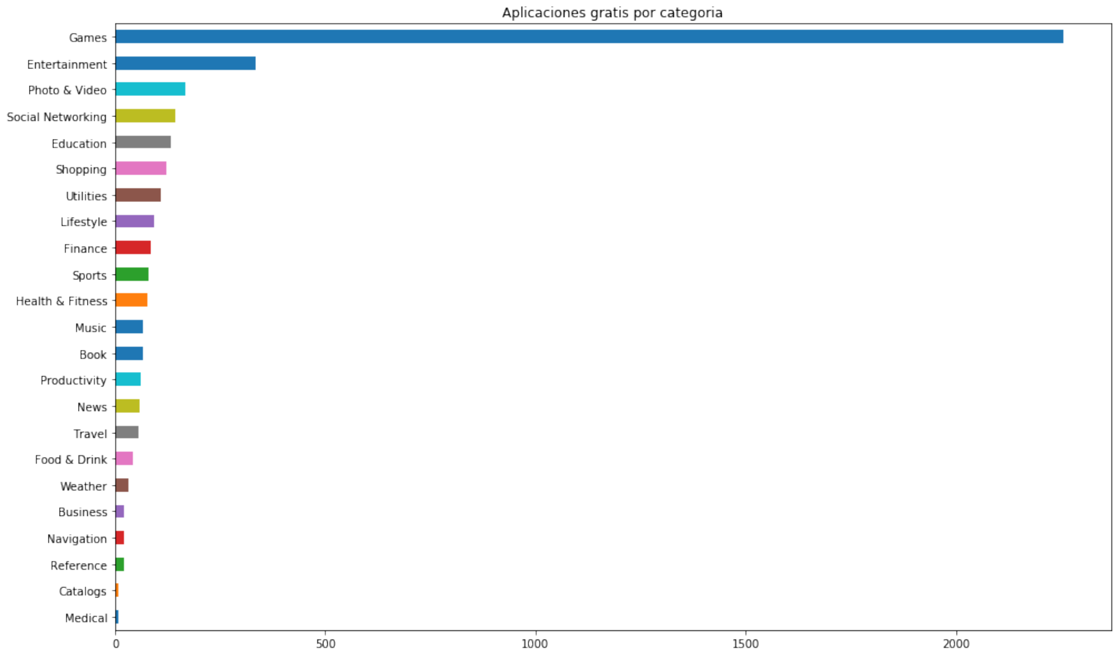
```
Out[14]: Text(0.5, 1.0, 'Aplicaciones pagas por categoria')
```



Cantidad de aplicaciones pagas por categoría

```
In [15]: # plot generos de apps gratis
         apps[apps['price'] == 0].prime_genre.value_counts(ascending=True).plot(kind='barh', fig
```

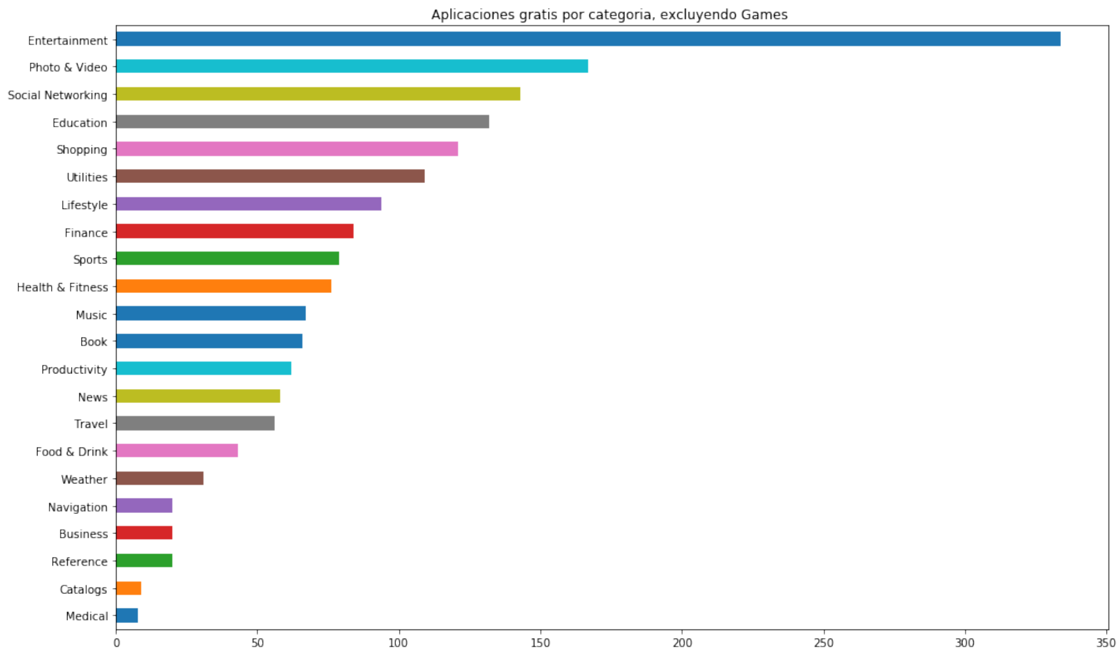
```
Out[15]: Text(0.5, 1.0, 'Aplicaciones gratis por categoria')
```



Cantidad de aplicaciones gratis por categoría

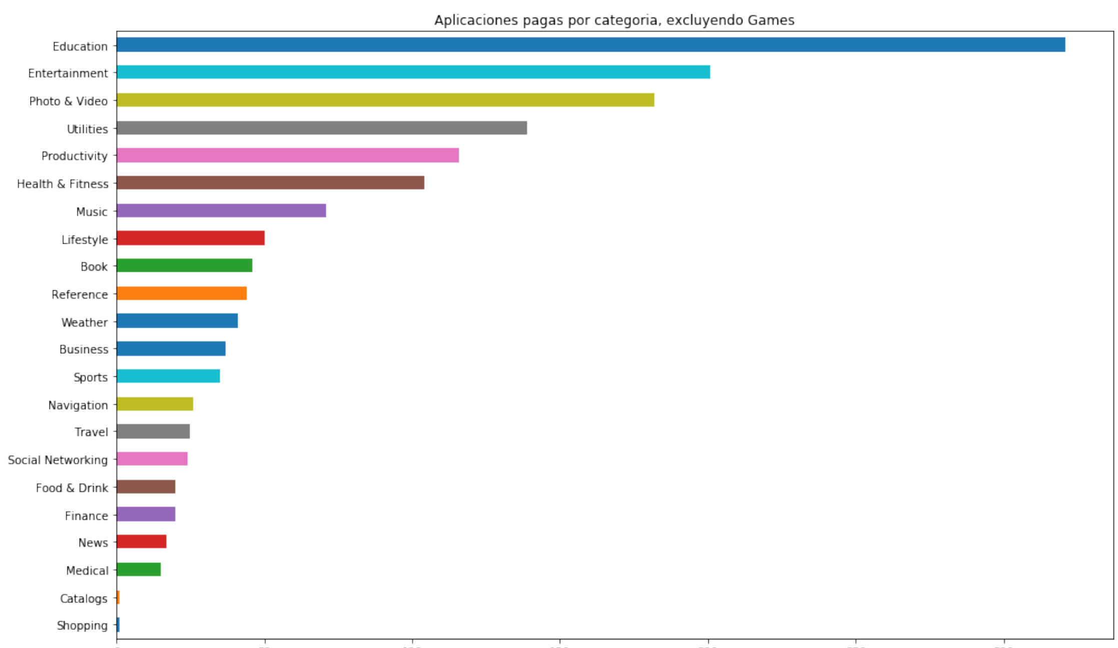
```
In [16]: # mismo plot filtrando games
         apps[(apps['price'] == 0) & (apps['prime_genre'] != 'Games')].prime_genre.value_counts(
```

```
Out[16]: Text(0.5, 1.0, 'Aplicaciones gratis por categoria, excluyendo Games ')
```



Cantidad de aplicaciones gratis por categoría filtrando games

```
In [17]: # mismo plot filtrando games
         apps[(apps['price'] > 0) & (apps['prime_genre'] != 'Games')].prime_genre.value_counts(a

Out[17]: Text(0.5, 1.0, 'Aplicaciones pagas por categoria, excluyendo Games')
```



Cantidad de aplicaciones pagas por categoria filtrando games

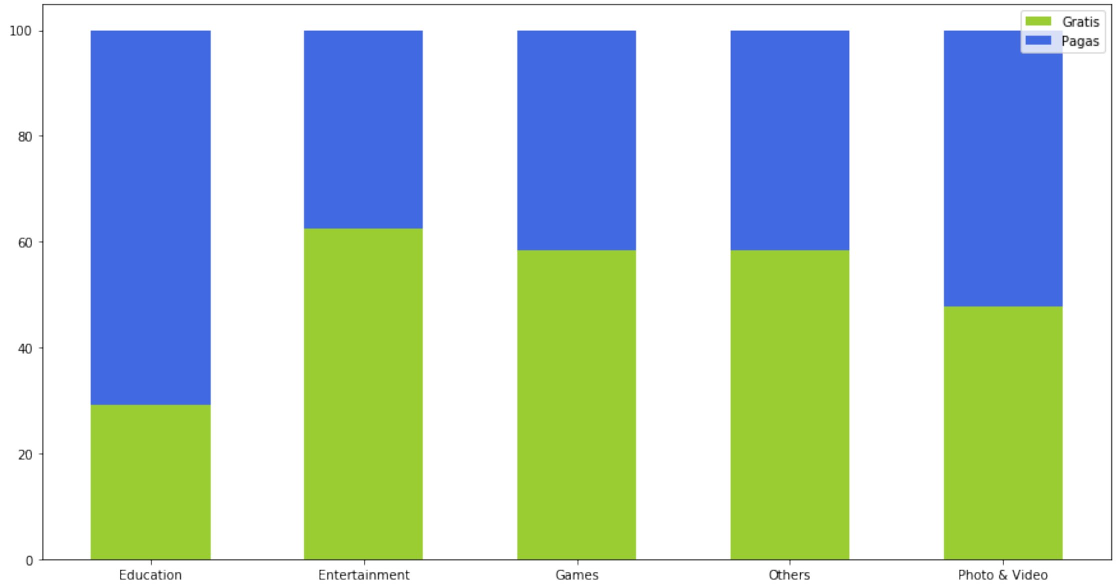
In [18]:
```
s = ['Games', 'Entertainment', 'Education', 'Photo & Video']

def categ(x):
    if x in s:
        return x
    else :
        return "Others"

apps['broad_genre']= apps.prime_genre.apply(lambda x : categ(x))
```

In [19]:
```
free = apps[apps.price==0].broad_genre.value_counts().sort_index().to_frame()
paid = apps[apps.price>0].broad_genre.value_counts().sort_index().to_frame()
total = apps.broad_genre.value_counts().sort_index().to_frame()
free.columns=['Gratis']
paid.columns=['Pagas']
total.columns=['Total']
dist = free.join(paid).join(total)
dist ['Porcentaje Pagas %'] = dist.Pagas*100/dist.Total
dist ['Porcentaje Gratis %'] = dist.Gratis*100/dist.Total
dist
```

Out[19]:

	Gratis	Pagas	Total	Porcentaje Pagas %	Porcentaje Gratis %
Education	132	321	453	70.860927	29.139073
Entertainment	334	201	535	37.570093	62.429907
Games	2257	1605	3862	41.558778	58.441222
Others	1166	832	1998	41.641642	58.358358
Photo & Video	167	182	349	52.148997	47.851003

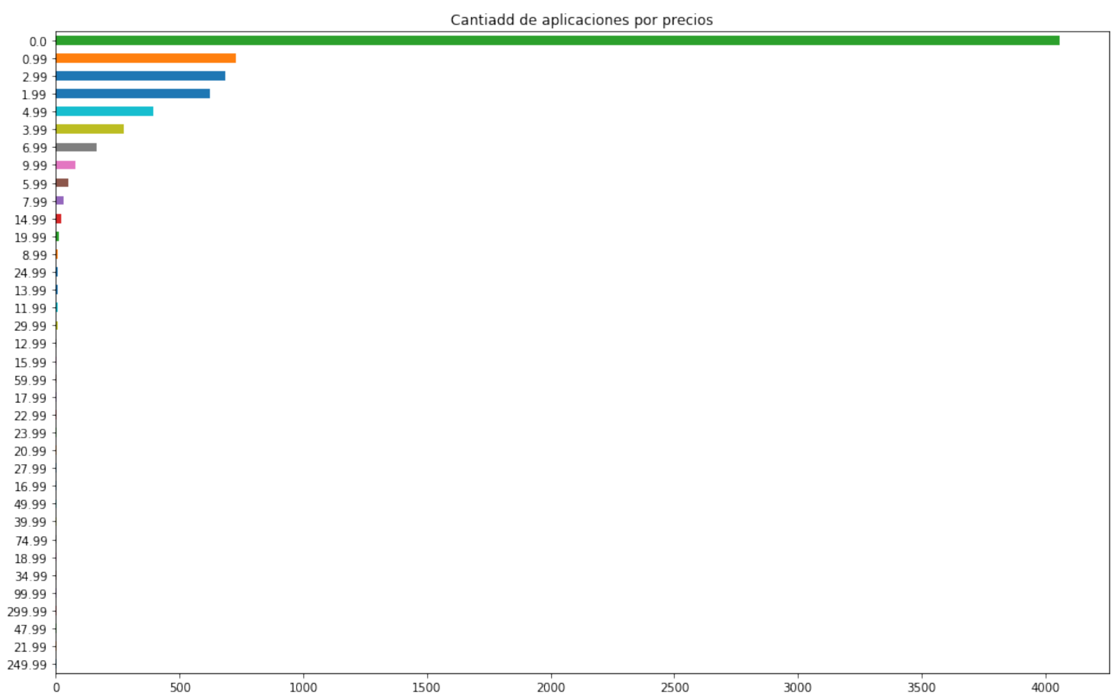
In [20]:
```
list_free= dist['Porcentaje Gratis %'].tolist()
tuple_free = tuple(list_free)
tuple_paidapps = tuple(dist['Porcentaje Pagas %'].tolist())
plt.figure(figsize=(15,8))
N=5
ind = np.arange(N)
width =0.56
p1 = plt.bar(ind, tuple_free, width, color='#9ACD32')
p2 = plt.bar(ind, tuple_paidapps, width,bottom=tuple_free,color='#4169E1')
plt.xticks(ind,tuple(dist.index.tolist() ))
plt.legend((p1[0], p2[0]), ('Gratis', 'Pagas'))
plt.show()

pies = dist[['Porcentaje Gratis %','Porcentaje Pagas %']]
pies.columns=['free %','paid %']
plt.show()
```



Vemos que para la única categoría que la cantidad de aplicaciones pagas supera a la cantidad de aplicaciones no pagas es Eduación

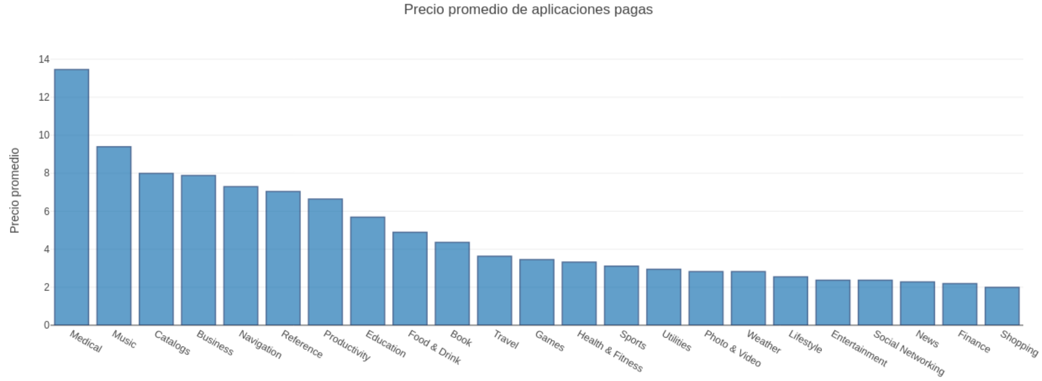
```
In [21]: apps.price.value_counts(ascending=True).plot(kind='barh', figsize=(16,10)).set_title('C
```

```
Out[21]: Text(0.5, 1.0, 'Cantiadd de aplicaciones por precios')
```



Como se puede ver, la gran mayoría de apps son gratis; de las pagas los precios van hasta los USD 10, a partir de ahí ya hay muy pocos casos como para generalizar.

```
In [22]: apps['isNotFree'] = apps['price'].apply(lambda x: 1 if x > 0 else 0)
         df_app_notfree = apps[apps['isNotFree'] == 1]
         df_app_free = apps[apps['isNotFree'] == 0]
         cnt_srs = df_app_notfree[['prime_genre', 'price']].groupby('prime_genre').mean()['price
         text = ['{:.2f}%'.format(100 * (value / cnt_srs.sum())) for value in cnt_srs.values]

         trace = go.Bar(
             x = cnt_srs.index,
             y = cnt_srs.values,
             text = text,
             marker = dict(
                 line = dict(color='rgb(8, 48, 107)',
                             width = 1.5)
             ),
             opacity = 0.7
         )
         data = [trace]

         layout = go.Layout(
             title = 'Precio promedio de aplicaciones pagas',
             margin = dict(
                 l = 100
             ),
             yaxis = dict(
                 title = 'Precio promedio'
             ),
             width = 800,
             height = 500
         )

         fig = go.Figure(data=data, layout=layout)
         py.iplot(fig)
```



Dentro de las aplicaciones pagas, las aplicaciones Medicas son las mas caras, con un promedio de casi 14 dolares. Y si analizamos aquellas categorías con mayor cantidad de aplicaciones en el store mantienen precios menores a 6 dólares en promedio, como lo son Games, Entretainment, Education y Photo & Video

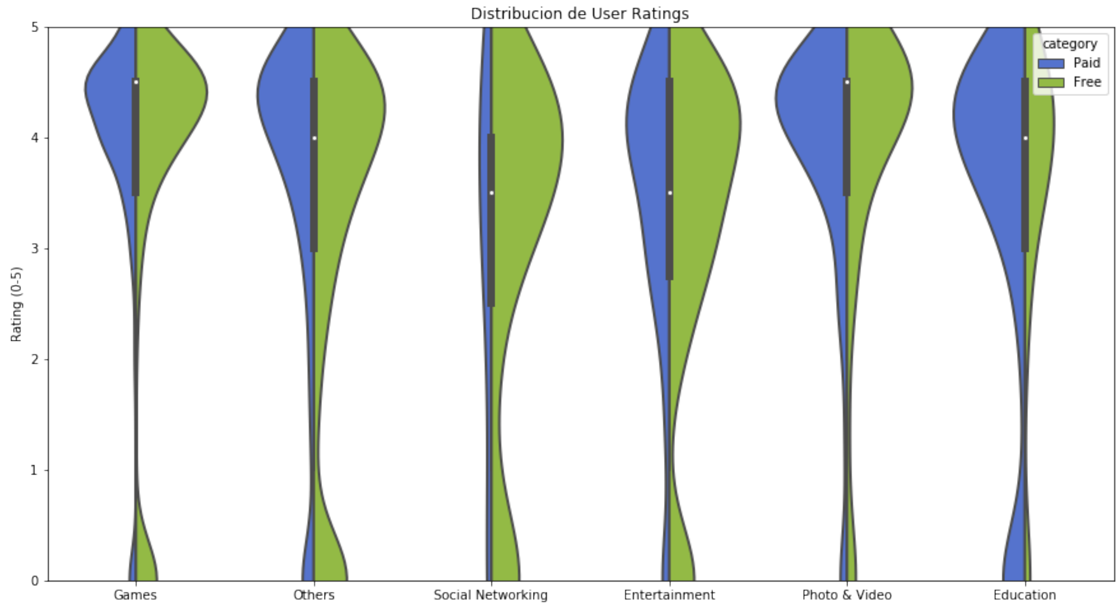
0.4 ## User ratings

Una vez analizadas la cantidad de aplicaciones pagas y gratis por categoría nos preguntamos si las aplicaciones pagas son realmente buenas. Esto lo vamos a analizar siguiendo la opinión de los usuarios.

El campo que define la opinion de los usuarios es USER RATING, que tiene valores entre 0 y 5, siendo 5 la mejor puntuacion.

```
In [23]: s = ['Games', 'Entertainment', 'Education', 'Photo & Video', 'Social Networking']

         def categ(x):
             if x in s:
                 return x
             else :
                 return "Others"

         apps['broad_genre']= apps.prime_genre.apply(lambda x : categ(x))
```

```
In [24]: plt.figure(figsize=(15,8))
         plt.style.use('fast')
         plt.ylim([0,5])
         plt.title("Distribucion de User Ratings")
         sns.violinplot(data=apps, y ='user_rating',x='broad_genre',hue='category',
                        vertical=True,kde=False,split=True ,linewidth=2,
                        scale ='count', palette=['#4169E1','#9ACD32'])
         plt.xlabel(" ")
         plt.ylabel("Rating (0-5)")

         plt.show()
```



Los resultados varían parecido en el caso de las apps pagas y de las gratuitas. En general vemos que la distribución de las puntuaciones de usuarios son parecidas en todas las categorías.

Se puede destacar que para el caso de `entretenimiento` y `otros` sí se aprecian puntuaciones entre 1 y 2, mientras que en el resto de las categorías las puntuaciones más presentes son 0 y luego entre 3 y 5, generalmente los valores entre 1 y 2 son poco usados. De todos modos las distribuciones son parecidas tanto para pagas como no pagas.

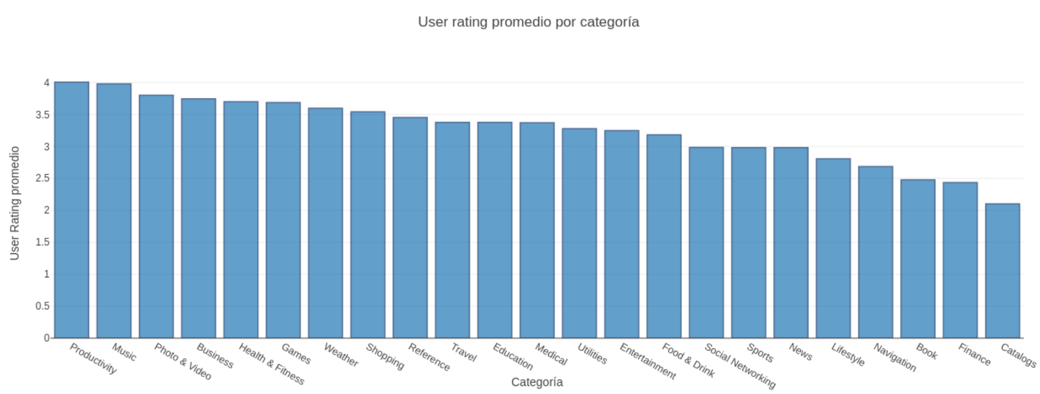
Se puede notar que en las apps pagas, en el género *Education* hay mayor concentración en la puntuación 4 para las pagas, mientras que para las no pagas, entre los valors 3 y 5, la distribuión es mas pareja.

La mayoría de apps relacionadas a Redes sociales por amplia diferencia es gratis. No sería popular tener que pagar por una red social; ademas las redes sociales generan ingresos mediante publicidades, por lo que conviene tener una base amplia de usuarios, por más que eso signifique que ellos estén 'gratis'.

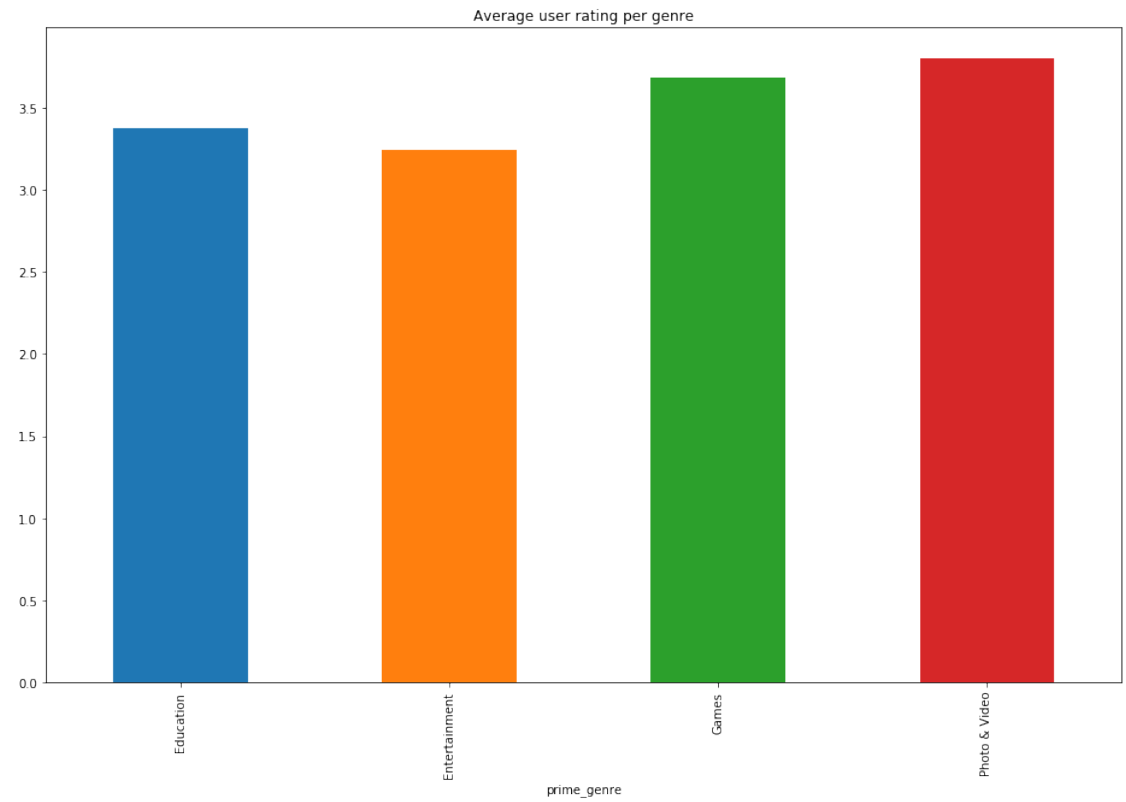
No perder de vista que todas las apps también pueden generar ingresos mediante *in app purchases*.

```
In [25]: cnt_srs = apps[['prime_genre', 'user_rating']].groupby('prime_genre').mean()['user_rati

         trace = go.Bar(
             x = cnt_srs.index,
             y = cnt_srs.values,
             marker = dict(
                 line = dict(color='rgb(8, 48, 107)',
                             width = 1.5)
             ),
             opacity = 0.7
         )
         data = [trace]
```

```
layout = go.Layout(
    title = 'User rating promedio por categoría',
    margin = dict(
        l = 100
    ),
    xaxis = dict(
        title = 'Categoría'
    ),
    yaxis = dict(
        title = 'User Rating promedio'
    ),
    width = 800,
    height = 500
)

fig = go.Figure(data=data, layout=layout)
py.iplot(fig)
```

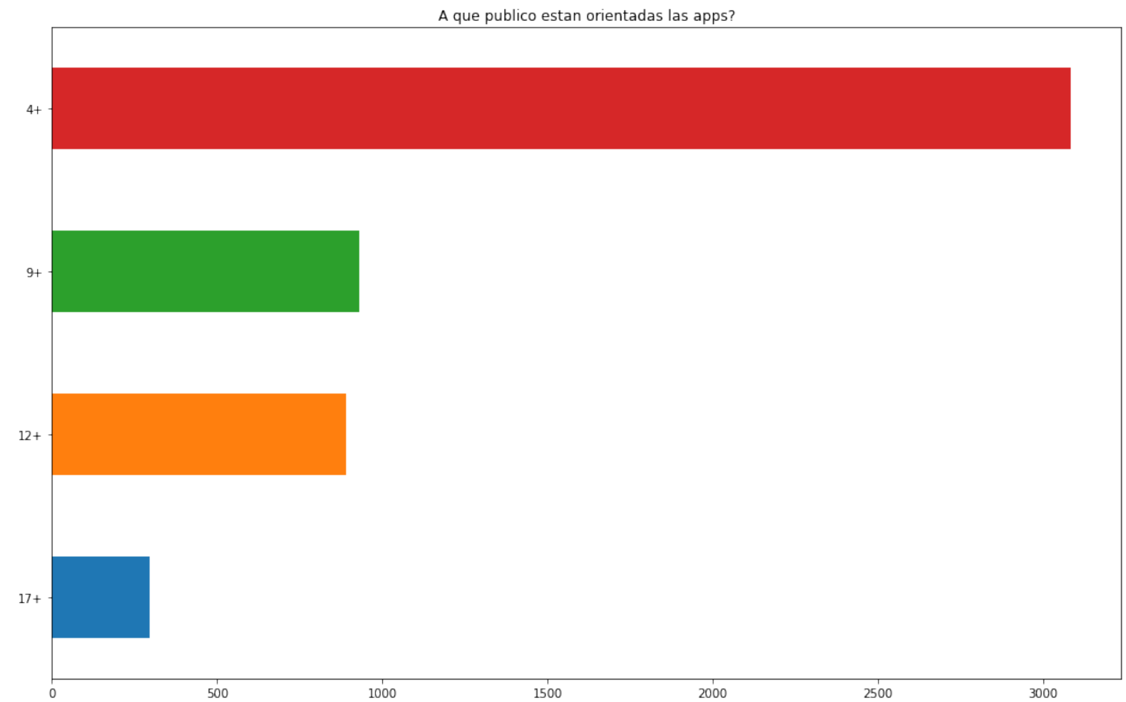



```
In [26]: interest_areas =  ['Games', 'Entertainment', 'Education', 'Photo & Video']
         apps[apps.prime_genre.isin(interest_areas)].groupby('prime_genre')['user_rating'].mean(

Out[26]: Text(0.5, 1.0, 'Average user rating per genre')
```



0.4.1 cont_rating

```
In [27]: # veamos los ratings de contenido de las areas de interes
         apps[apps['prime_genre'].isin(interest_areas)].cont_rating.value_counts(ascending=True)

Out[27]: Text(0.5, 1.0, 'A que publico estan orientadas las apps?')
```



La columna de rating edad se transforma en feature numérico, ya que no tiene sentido que sea del tipo *object*.

```
In [28]: apps.cont_rating = apps.cont_rating.apply(lambda x: int(x.split('+')[0]))
```

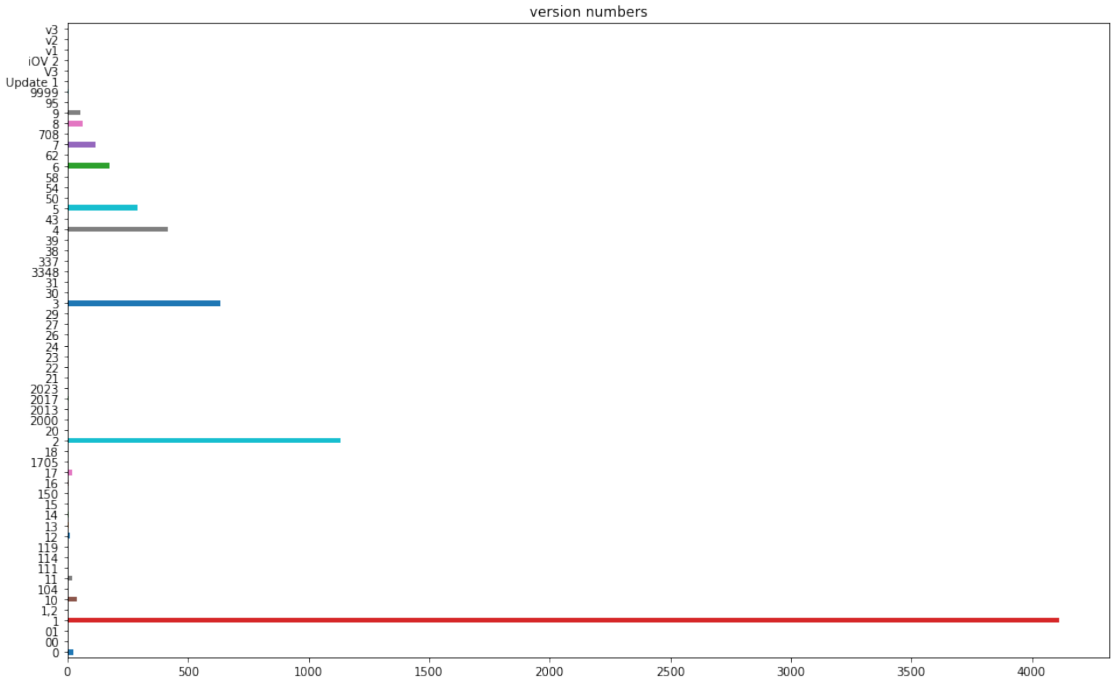
0.4.2 track_name

```
In [29]: apps.track_name = apps.track_name.apply(lambda x: x.lower())
```

0.4.3 ver

Nos quedamos con la *major version* de las apps.

```
In [30]: # for Apple versioning criteria, see: https://en.wikipedia.org/wiki/Software_versioning
         apps['major_version'] = apps.ver.apply(lambda x: x.split('.')[0])

In [31]: apps.major_version.value_counts(ascending=True).sort_index().plot(kind='barh', figsize=

Out[31]: Text(0.5, 1.0, 'version numbers')
```



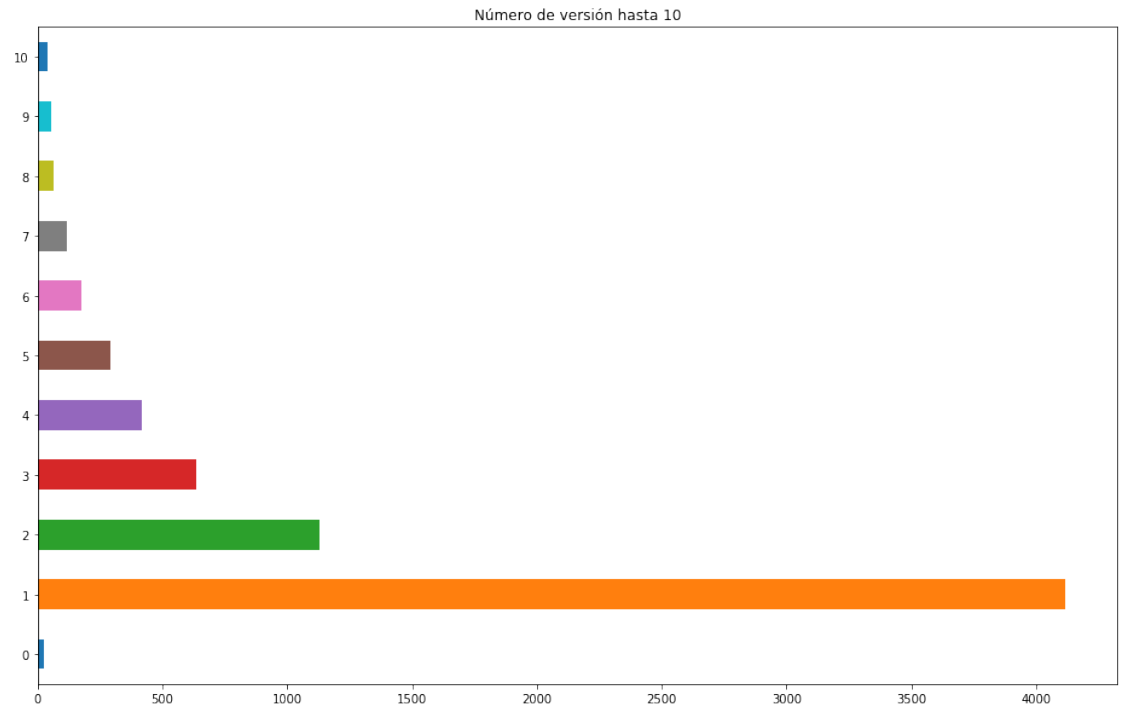
Se observan versiones un poco raras.... como `1,2` o `iOV2`, `9999`, `2000`.... no es posible que esto sean numeros de versiones reales.

Mas alla de eso, la gran mayoria esta en su primera version, llegando hasta la 9. Mas alla de la 9

```
In [32]: # pasaje a mayor version a numerico descartando los registros que no son validos.
         # Consideramos que los registros que no tienen nro de version valido, no se pueden cons
         apps.loc[:,'major_version'] = pd.to_numeric(apps['major_version'], errors='coerce')
         apps = apps.dropna(subset=['major_version'])
         apps.loc[:,'major_version'] = apps['major_version'].astype('int')
```

```
In [33]: apps[apps['major_version'].isin(range(11))].major_version.value_counts(ascending=True).
```

```
Out[33]: Text(0.5, 1.0, 'Número de versión hasta 10')
```



1 Merge datasets

In [34]:
```
# se hace left join porque importan solamente los registros que estan en apps df
m = apps.merge(desc, how='left')
```

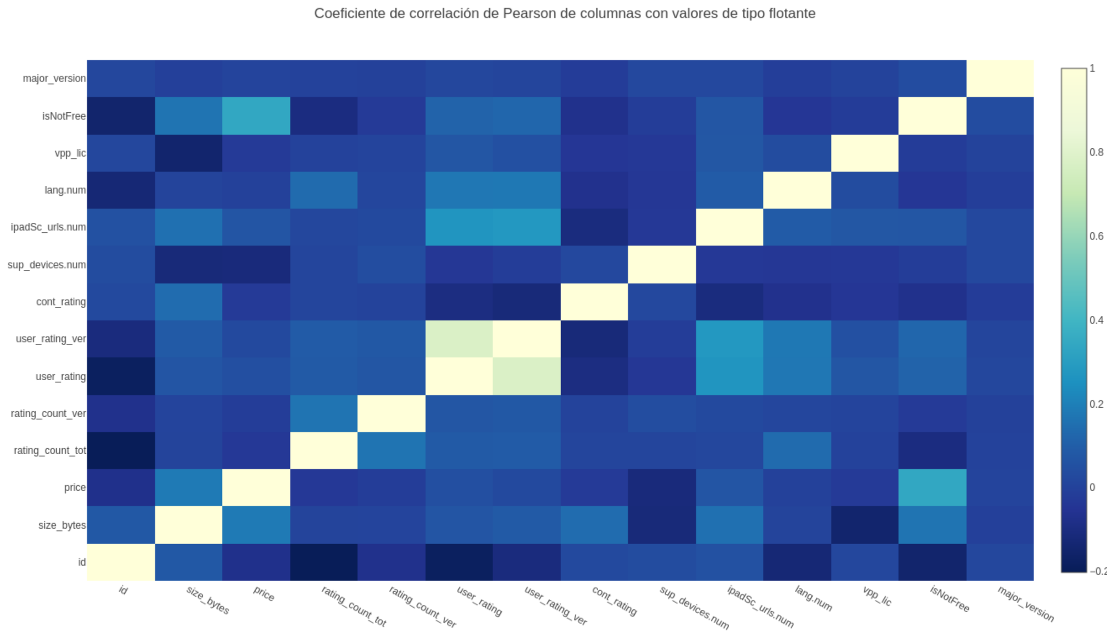
In [35]:
```
m.head()
```

Out[35]:
```
          id                                     track_name  size_bytes  \
0  281656475                                pac-man premium   100788224
1  281796108                      evernote - stay organized   158578688
2  281940292    weatherbug - local weather, radar, maps, alerts   100524032
3  282614216  ebay: best app to buy, sell, save! online shop...   128512000
4  282935706                                          bible    92774400

   price  rating_count_tot  rating_count_ver  user_rating  user_rating_ver  \
0   3.99             21292                26          4.0              4.5
1   0.00            161065                26          4.0              3.5
2   0.00            188583              2822          3.5              4.5
3   0.00            262241               649          4.0              4.5
4   0.00            985920              5320          4.5              5.0

     ver  cont_rating   prime_genre  sup_devices.num  ipadSc_urls.num  \
0  6.3.5            4         Games               38                5
1  8.2.2            4  Productivity               37                5
```

```
2   5.0.0            4       Weather               37                5
3  5.10.0           12      Shopping               37                5
4   7.5.1            4     Reference               37                5

   lang.num  vpp_lic category broad_genre  isNotFree  major_version  \
0        10        1     Paid       Games          1              6
1        23        1     Free      Others          0              8
2         3        1     Free      Others          0              5
3         9        1     Free      Others          0              5
4        45        1     Free      Others          0              7

                                            app_desc
0  SAVE 20%, now only $3.99 for a limited time!\n...
1  Let Evernote change the way you organize your ...
2  Download the most popular free weather app pow...
3  The eBay app is the best way to find anything ...
4  On more than 250 million devices around the wo...
```

Con esto se obtienen datos adicionales: track name, tamano de la app en bytes y la descripcion.

In [36]:
```
data = [
    go.Heatmap(
        z = m.corr().values,
        x = m.corr().columns.values,
        y = m.corr().columns.values,
        colorscale='YlGnBu',
        reversescale=False,
    )
]

layout = go.Layout(
    title='Coeficiente de correlación de Pearson de columnas con valores de tipo flotan
    xaxis = dict(ticks=''),
    yaxis = dict(ticks='' ),
    width = 800, height = 800,
    margin = dict(
        l = 100
    )
)

fig = go.Figure(data=data, layout=layout)
py.iplot(fig, filename='labelled-heatmap')
```



El coeficiente de correlación de Pearson es la estadística de prueba que mide la relación estadística, o asociación, entre dos variables continuas. Es conocido como el mejor método para medir la asociación entre variables de interés porque se basa en el método de covarianza. Da información sobre la magnitud de la asociación, o correlación, así como la dirección de la relación

1.0.1 Análisis de la descripción de las apps

Se agregan tres columnas con las tres palabras mas frecuentes en la descripción de las apps, omitiendo stopwords.

```
In [37]: desc = m['app_desc'][0].split(' ')
```

```
In [38]: import nltk
         from nltk.corpus import stopwords
         nltk.download('punkt')
         nltk.download('stopwords')
```

```
[nltk_data] Downloading package punkt to /home/rozanecm/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package stopwords to
[nltk_data]     /home/rozanecm/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

```
Out[38]: True
```

```
In [39]: %%time
         for i in range(m.shape[0]):
```

```
        temp_desc = m['app_desc'][i]
        temp_word_list = nltk.word_tokenize(temp_desc)
        temp_word_list = [word.lower() for word in temp_word_list if word not in stopwords.
        for char in " {}()#&[]^`ť-_ů@|£?ą!'+*\"?.!/;:<>ż%,":
            for ele in temp_word_list:
                if char in ele:
                    temp_word_list.remove(ele)
        fdist = nltk.FreqDist(temp_word_list)
        temp_srs = pd.Series(fdist).sort_values(ascending=False)
        try:
            m.loc[i, 'most_freq_word_1'] = temp_srs.index[0]
            m.loc[i, 'most_freq_word_2'] = temp_srs.index[1]
            m.loc[i, 'most_freq_word_3'] = temp_srs.index[2]
        except:
            m.loc[i, 'most_freq_word_1'] = temp_srs.index[0]
```

```
CPU times: user 4min 52s, sys: 17.7 s, total: 5min 10s
Wall time: 5min 10s
```

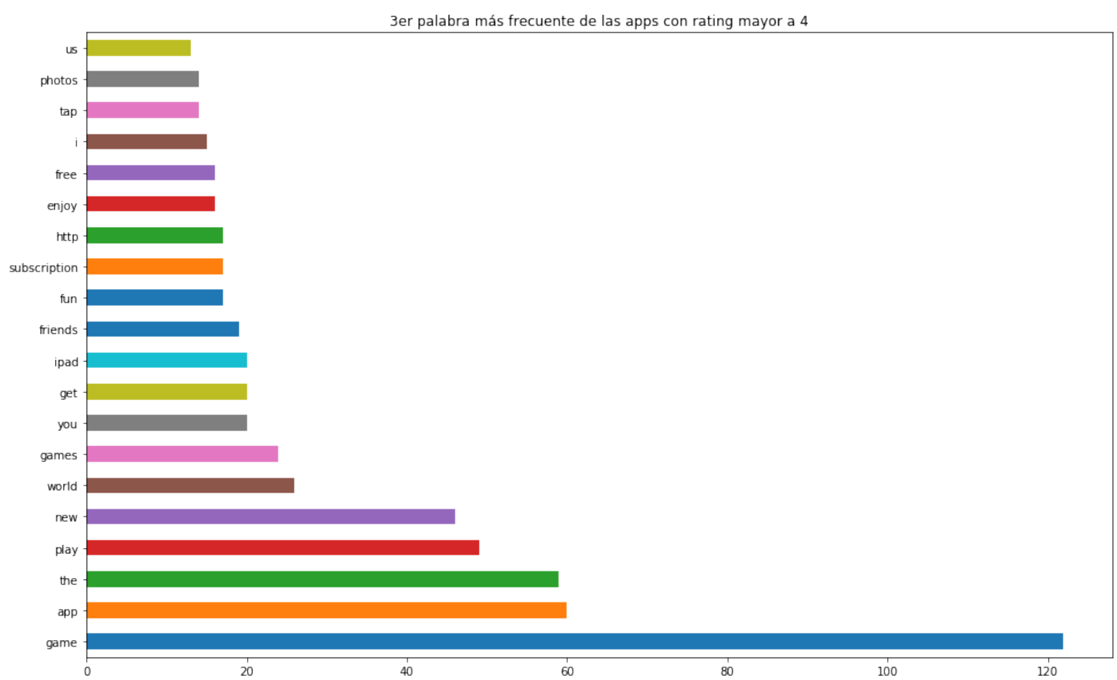
In [40]:
```
m.tail()
```

Out[40]:
```
              id                               track_name  size_bytes  \
7184  1187617475                                    kubik   126644224
7185  1187682390                        vr roller-coaster   120760320
7186  1187779532       bret michaels emojis + lyric keyboard   111322112
7187  1187838770  vr roller coaster world - virtual reality    97235968
7188  1188375727                escape the sweet shop series    90898432

      price  rating_count_tot  rating_count_ver  user_rating  user_rating_ver  \
7184   0.00               142                75          4.5              4.5
7185   0.00                30                30          4.5              4.5
7186   1.99                15                 0          4.5              0.0
7187   0.00                85                32          4.5              4.5
7188   0.00                 3                 3          5.0              5.0

         ver  cont_rating       ...        lang.num  vpp_lic  category  \
7184     1.3            4       ...               1        1      Free
7185     0.9            4       ...               1        1      Free
7186   1.0.2            9       ...               1        1      Paid
7187  1.0.15           12       ...               2        1      Free
7188     1.0            4       ...               2        1      Free

      broad_genre  isNotFree major_version  \
7184        Games          0             1
7185        Games          0             0
7186       Others          1             1
7187        Games          0             1
```

```
7188       Games         0            1

                                              app_desc  most_freq_word_1  \
7184  Place the falling blocks correctly in order to...            blocks
7185  A thrilling virtual reality roller coaster exp...               vox
7186  Rock star Bret Michaels, winner of Celebrity A...              bret
7187  VR Roller Coaster World is an app for Google C...                vr
7188  5 previous escape games plus 1 new game in one...              game

     most_freq_word_2 most_freq_word_3
7184          falling           layers
7185          virtual               3d
7186             rock           lyrics
7187            world          coaster
7188           escape             shop

[5 rows x 23 columns]
```

In [41]:
```
m.loc[m['user_rating'] > 4, 'most_freq_word_3'].value_counts().head(20).plot(kind='barh
```

Out[41]:
```
Text(0.5, 1.0, '3er palabra más frecuente de las apps con rating mayor a 4')
```



1.1 Preparación de datos

In [42]:
```
from sklearn.metrics import confusion_matrix
from sklearn.preprocessing import OneHotEncoder
from sklearn.model_selection import train_test_split
```

Vamos a definir una columna que clasifique como **aplicaciones exitosas aquellas con user raiting mayor o igual a 4**.

```
In [43]: m['succesful'] = apps['user_rating'].apply(lambda x: 1 if x > 3 else 0)
         m.tail(10)
```

```
Out[43]:            id                                track_name  size_bytes  \
         7179  1186126548                 escape game: illumination    52342784
         7180  1186384912  demolition derby virtual reality (vr) racing 168774656
         7181  1187128255                       -pk   537462784
         7182  1187279979                 add-ons studio for minecraft    22999040
         7183  1187282363                   plead the fifth - the game    27853824
         7184  1187617475                                         kubik   126644224
         7185  1187682390                              vr roller-coaster  120760320
         7186  1187779532        bret michaels emojis + lyric keyboard   111322112
         7187  1187838770      vr roller coaster world - virtual reality    97235968
         7188  1188375727                 escape the sweet shop series     90898432

               price  rating_count_tot  rating_count_ver  user_rating  user_rating_ver  \
         7179   0.00                23                23          4.5              4.5
         7180   0.00                18                18          4.0              4.0
         7181   0.99                 0                 0          0.0              0.0
         7182   2.99                97                97          3.0              3.0
         7183   2.99                11                 0          4.0              0.0
         7184   0.00               142                75          4.5              4.5
         7185   0.00                30                30          4.5              4.5
         7186   1.99                15                 0          4.5              0.0
         7187   0.00                85                32          4.5              4.5
         7188   0.00                 3                 3          5.0              5.0

                  ver  cont_rating    ...    vpp_lic  category  broad_genre  isNotFree  \
         7179     1.0            4    ...          1      Free        Games          0
         7180   1.0.0           12    ...          1      Free        Games          0
         7181   2.1.0            9    ...          1      Paid        Games          1
         7182     1.0            4    ...          1      Paid        Games          1
         7183   1.1.1           17    ...          1      Paid        Games          1
         7184     1.3            4    ...          1      Free        Games          0
         7185     0.9            4    ...          1      Free        Games          0
         7186   1.0.2            9    ...          1      Paid       Others          1
         7187  1.0.15           12    ...          1      Free        Games          0
         7188     1.0            4    ...          1      Free        Games          0

               major_version                                       app_desc  \
         7179              1  Escape from here! \n\nTap on the objects in th...
         7180              1  Multiplayer Demolition Derby goes Virtual Real...
         7181              2  ARPGPK...
         7182              1  The makers of the official Minecraft Skin Stud...
         7183              1  It's the cheeky, provocative, late-night telev...
```

```
7184              1  Place the falling blocks correctly in order to...
7185              0  A thrilling virtual reality roller coaster exp...
7186              1  Rock star Bret Michaels, winner of Celebrity A...
7187              1  VR Roller Coaster World is an app for Google C...
7188              1  5 previous escape games plus 1 new game in one...

     most_freq_word_1  most_freq_word_2  most_freq_word_3 succesful
7179            items              look            closer       1.0
7180               vr            racing               car       1.0
7181                                                        0.0
7182        minecraft             horse            rabbit       0.0
7183            round              game             scoop       1.0
7184           blocks           falling            layers       0.0
7185              vox           virtual                3d       0.0
7186             bret              rock            lyrics       0.0
7187               vr             world           coaster       1.0
7188             game            escape              shop       1.0

[10 rows x 24 columns]
```

Agregamos la columna que va a definir si es una aplicación exitosa. Como mencionamos antes, son aplicaciones exitosas aquellas con user rating mayor o igual a 4.

Eliminamos las columnas ID y category.

Esta decisión fue tomada porque, para el caso de ID, como se ve en el plot del coeficiente de Perason, la correlación con el resto de los atributos es 0 o menor a 0.

Y para el caso de category, la columna isNotFree representa lo mismo.

```
In [44]: m = m.drop(['id', 'category'], axis=1)
         m.tail()
```

```
Out[44]:                                  track_name  size_bytes  price  \
         7184                                  kubik   126644224   0.00
         7185                      vr roller-coaster   120760320   0.00
         7186      bret michaels emojis + lyric keyboard   111322112   1.99
         7187  vr roller coaster world - virtual reality    97235968   0.00
         7188                escape the sweet shop series    90898432   0.00

               rating_count_tot  rating_count_ver  user_rating  user_rating_ver  \
         7184               142                75          4.5              4.5
         7185                30                30          4.5              4.5
         7186                15                 0          4.5              0.0
         7187                85                32          4.5              4.5
         7188                 3                 3          5.0              5.0

                 ver  cont_rating prime_genre    ...     lang.num  vpp_lic  \
         7184    1.3            4       Games    ...            1        1
```

```
7185     0.9             4       Games    ...             1         1
7186   1.0.2             9   Utilities    ...             1         1
7187  1.0.15            12       Games    ...             2         1
7188     1.0             4       Games    ...             2         1

      broad_genre  isNotFree major_version  \
7184        Games          0             1
7185        Games          0             0
7186       Others          1             1
7187        Games          0             1
7188        Games          0             1

                                               app_desc  most_freq_word_1  \
7184  Place the falling blocks correctly in order to...            blocks
7185  A thrilling virtual reality roller coaster exp...               vox
7186  Rock star Bret Michaels, winner of Celebrity A...              bret
7187  VR Roller Coaster World is an app for Google C...                vr
7188  5 previous escape games plus 1 new game in one...              game

     most_freq_word_2 most_freq_word_3 succesful
7184          falling           layers       0.0
7185          virtual               3d       0.0
7186             rock           lyrics       0.0
7187            world          coaster       1.0
7188           escape             shop       1.0

[5 rows x 22 columns]
```

In [45]:
```
m.isnull().any()
```

Out[45]:
```
track_name          False
size_bytes          False
price               False
rating_count_tot    False
rating_count_ver    False
user_rating         False
user_rating_ver     False
ver                 False
cont_rating         False
prime_genre         False
sup_devices.num     False
ipadSc_urls.num     False
lang.num            False
vpp_lic             False
broad_genre         False
isNotFree           False
major_version       False
app_desc            False
```

```
most_freq_word_1    False
most_freq_word_2     True
most_freq_word_3     True
succesful            True
dtype: bool
```

Vemos que tenemos valores nulos en: * most_freq_word_2 True * most_freq_word_3 True * succesful True

Vamos a analizar cuántos registros de cada uno son los que cuentan con estos valores nulos

In [46]:
```
m.succesful.isnull().sum()
```

Out[46]:
```
8
```

In [47]:
```
nan_rows = m[m['succesful'].isnull()]
nan_rows
```

Out[47]:
```
                                          track_name  size_bytes  price  \
224                            the impossible quiz!    44652544   0.00
1133                      abrsm aural trainer grades 1-5   329866240   7.99
3092  todo number matrix: brain teasers, logic puzzl...    41566208   0.99
3224                                          -    85620736   0.00
3581                               green riding hood   316589056   0.00
3668                            immortal legends - td   273789952   0.00
4121                         jcnews - anime & game culture    25467904   0.00
4176                                 292899840   0.00

      rating_count_tot  rating_count_ver  user_rating  user_rating_ver    ver  \
224              18884               451          4.0              4.5   1.62
1133                 0                 0          0.0              0.0    2.5
3092                15                 5          4.5              4.0  1.1.1
3224                 0                 0          0.0              0.0  4.7.0
3581               392                 2          4.0              5.0  1.2.1
3668                26                26          3.0              3.0  2.0.0
4121                 0                 0          0.0              0.0  2.0.5
4176                 0                 0          0.0              0.0  2.1.6

      cont_rating    prime_genre    ...      lang.num  vpp_lic    broad_genre  \
224             9  Entertainment    ...             1        1  Entertainment
1133            4      Education    ...             1        1      Education
3092            4      Education    ...             8        1      Education
3224            4       Shopping    ...             2        1         Others
3581            4           Book    ...            12        1         Others
3668            9          Games    ...             3        1          Games
4121           17           News    ...            31        1         Others
4176           17          Games    ...             1        1          Games

      isNotFree major_version  \
224           0             1
```

```
1133          1             2
3092          1             1
3224          0             4
3581          0             1
3668          0             2
4121          0             2
4176          0             2

                                              app_desc  most_freq_word_1  \
224   Its not the "difficult quiz". Its not the "r...             quiz
1133  The OFFICIAL ABRSM Aural Trainer contains inte...            music
3092  4.5 stars! "Todo Number Matrix is a unique mat...             todo
3224  \n20153...
3581  * "Free App of the Week" in 2017\n* iPad App o...              app
3668  Defend your spirit master in epic battles, evo...           heroes
4121  We'll update the latest news of Japanese anima...             news
4176  200\n...                  --

          most_freq_word_2 most_freq_word_3 succesful
224             impossible               it       NaN
1133                 aural            abrsm       NaN
3092                number         children       NaN
3224                        NaN
3581                riding            green       NaN
3668               journey           across       NaN
4121                                    you       NaN
4176                                               NaN

[8 rows x 22 columns]
```

Para el caso de ***succesful*** son solo 8 registros. Al ser tan pocos determinamos directamente eliminar dichos registros

```
In [48]: m.most_freq_word_2.isnull().sum()

Out[48]: 7

In [49]: nan_rows = m[m['most_freq_word_2'].isnull()]
         nan_rows

Out[49]:                 track_name  size_bytes  price  rating_count_tot  \
         4899                     97892352   0.00                 0
         5890                    267673600   0.99                 0
         5982                     67259392   0.00                 0
         6440                  27525120   0.00                 0
         6765           83985408   0.00                 0
         7044  -   106102784   0.00                 5
         7112              js     1841152   0.00                 0
```

```
     rating_count_ver  user_rating  user_rating_ver     ver  cont_rating  \
4899                0          0.0              0.0   2.2.1           17
5890                0          0.0              0.0   10.99            4
5982                0          0.0              0.0     1.7           17
6440                0          0.0              0.0   1.0.6            4
6765                0          0.0              0.0     1.6           12
7044                0          3.5              0.0  0.16.3            4
7112                0          0.0              0.0     1.0           17

        prime_genre    ...     lang.num  vpp_lic    broad_genre  isNotFree  \
4899      Lifestyle    ...            2        1         Others          0
5890          Games    ...            1        1          Games          1
5982          Games    ...            1        1          Games          0
6440  Entertainment    ...            1        1  Entertainment          0
6765          Games    ...            1        1          Games          0
7044          Games    ...            2        1          Games          0
7112  Entertainment    ...            1        1  Entertainment          0

     major_version                                          app_desc  \
4899             2  ...
5890            10  MMORPGQMMORPG...
5982             1
6440             1  ...
6765             1  ...
7044             0  3D...
7112             1

                                 most_freq_word_1 most_freq_word_2  \
4899  ...              NaN
5890  mmorpgqmmorpg...             NaN
5982                                     NaN
6440  ...              NaN
6765  ...              NaN
7044  3d...              NaN
7112                        NaN

     most_freq_word_3 succesful
4899              NaN       1.0
5890              NaN       0.0
5982              NaN       1.0
6440              NaN       0.0
6765              NaN       1.0
7044              NaN       1.0
7112              NaN       1.0

[7 rows x 22 columns]
```

Para el caso de ***most_freq_word_2*** son solo 7 registros. Al ser tan pocos determinamos directamente eliminar dichos registros

In [50]:
```
m.most_freq_word_3.isnull().sum()
```

Out[50]:
```
14
```

In [51]:
```
nan_rows = m[m['most_freq_word_3'].isnull()]
nan_rows
```

Out[51]:
```
                track_name  size_bytes  price  rating_count_tot  \
560    hd    68668416   0.00             990
1267              61895680   0.00              22
1285          -    98640896   0.00               9
4899                    97892352   0.00                 0
4912              20328448   0.00               0
5503                 44026880   0.00                0
5890                  267673600   0.99                0
5982                 67259392   0.00                0
6440              27525120   0.00               0
6765        83985408   0.00                 0
6962             gogogo     7212032   0.00                 0
7019  777-2017   102120448   0.00                0
7044  -   106102784   0.00                 5
7112              js     1841152   0.00                0

      rating_count_ver  user_rating  user_rating_ver    ver  cont_rating  \
560                  4          3.0              2.0  5.5.2           12
1267                 0          3.0              0.0  4.1.8            4
1285                 0          1.5              0.0  6.1.2           17
4899                 0          0.0              0.0  2.2.1           17
4912                 0          0.0              0.0    2.0            4
5503                 0          0.0              0.0    1.1           12
5890                 0          0.0              0.0  10.99            4
5982                 0          0.0              0.0    1.7           17
6440                 0          0.0              0.0  1.0.6            4
6765                 0          0.0              0.0    1.6           12
6962                 0          0.0              0.0    1.0            4
7019                 0          0.0              0.0  1.1.0           17
7044                 0          3.5              0.0 0.16.3            4
7112                 0          0.0              0.0    1.0           17

        prime_genre    ...     lang.num  vpp_lic   broad_genre  isNotFree  \
560   Entertainment    ...            1        1 Entertainment          0
1267      Lifestyle    ...            2        1        Others          0
1285        Finance    ...            1        1        Others          0
4899      Lifestyle    ...            2        1        Others          0
4912          Games    ...            2        1         Games          0
5503          Games    ...            1        1         Games          0
5890          Games    ...            1        1         Games          1
```

```
5982          Games    ...           1        1         Games          0
6440  Entertainment    ...           1        1  Entertainment         0
6765          Games    ...           1        1         Games          0
6962       Shopping    ...           2        1        Others          0
7019          Games    ...           1        1         Games          0
7044          Games    ...           2        1         Games          0
7112  Entertainment    ...           1        1  Entertainment         0

     major_version                                          app_desc  \
560              5  ...
1267             4  ...
1285             6  ...
4899             2  ...
4912             2                         HEY!HEY!HEY!!\n
5503             1  \n
5890            10  MMORPGQMMORPG...
5982             1
6440             1  ...
6765             1  ...
6962             1                               gogogo APP
7019             1  777\n...
7044             0  3D...
7112             1

                                  most_freq_word_1  \
560   ...
1267
1285
4899  ...
4912                                           hey
5503
5890  mmorpgqmmorpg...
5982
6440  ...
6765  ...
6962                                        gogogo
7019  77...
7044  3d...
7112

                                  most_freq_word_2 most_freq_word_3  \
560                  app              NaN
1267  ...          NaN
1285  ...          NaN
4899                                           NaN              NaN
4912                                                 NaN
5503                                  NaN
5890                                           NaN              NaN
```

```
5982                                                NaN              NaN
6440                                                NaN              NaN
6765                                                NaN              NaN
6962                                       app              NaN
7019                                       777              NaN
7044                                                NaN              NaN
7112                                                NaN              NaN

     succesful
560        1.0
1267       1.0
1285       1.0
4899       1.0
4912       0.0
5503       0.0
5890       0.0
5982       1.0
6440       0.0
6765       1.0
6962       1.0
7019       0.0
7044       1.0
7112       1.0

[14 rows x 22 columns]
```

Para el caso de ***most_freq_word_3*** son solo 14 registros. Al ser tan pocos determinamos directamente eliminar dichos registros

```
In [52]: m.dropna(inplace=True)
         m.isnull().any()
```

```
Out[52]: track_name           False
         size_bytes           False
         price                False
         rating_count_tot     False
         rating_count_ver     False
         user_rating          False
         user_rating_ver      False
         ver                  False
         cont_rating          False
         prime_genre          False
         sup_devices.num      False
         ipadSc_urls.num      False
         lang.num             False
         vpp_lic              False
         broad_genre          False
         isNotFree            False
```

```
major_version       False
app_desc            False
most_freq_word_1    False
most_freq_word_2    False
most_freq_word_3    False
succesful           False
dtype: bool
```

```
In [53]: m['rating_count_previo'] = m['rating_count_tot'] - m['rating_count_ver']

         df_train = m[['size_bytes', 'isNotFree', 'price', 'rating_count_previo', 'sup_devices.n
         target = m['user_rating']

         df_train = pd.get_dummies(df_train)

         target = target.apply(lambda x: 1 if x>3 else 0)

         X_train, X_test, y_train, y_test = train_test_split(df_train.values, target, test_size=

         print('X_train shape:', X_train.shape)
         print('X_test shape:', X_test.shape)
```

```
X_train shape: (5733, 31)
X_test shape: (1434, 31)
```

2 Predicciones

Para el presente caso de estudio se eligió utilizar la métrica *Accuracy score*. De la documentación de `sklearn`:

Accuracy classification score. In multilabel classification, this function computes subset accuracy: the set of labels predicted for a sample must exactly match the corresponding set of labels in y_true.

```
In [54]: from sklearn.metrics import accuracy_score
         from sklearn.model_selection import KFold
         from sklearn.model_selection import cross_val_score, cross_validate

         from sklearn.ensemble import RandomForestClassifier
         from lightgbm import LGBMClassifier
         from xgboost import XGBClassifier
         from keras.models import Sequential
```

```
Using TensorFlow backend.
```

```
In [55]: from keras.layers import Dense
         from keras.wrappers.scikit_learn import KerasClassifier
```

```python
    # Function to create model, required for KerasClassifier
    def create_model():
        # create model
        model = Sequential()
        model.add(Dense(12, activation='relu'))
        model.add(Dense(8, activation='relu'))
        model.add(Dense(1, activation='sigmoid'))

        # Compile model
        model.compile(optimizer='rmsprop',
                      loss='binary_crossentropy',
                      metrics=['accuracy'])
        return model
```

In [56]:
```python
models = [RandomForestClassifier(), LGBMClassifier(), XGBClassifier(), KerasClassifier(

kfold = KFold(n_splits=5, random_state=1234)

model_comparison = pd.DataFrame(columns=['Classfier_name', 'train_score', 'test_score']

for i, model in enumerate(models):
    clf = model
    cv_result = cross_validate(model, X_train, y_train, cv=kfold, scoring='accuracy')
    model_comparison.loc[i, 'Classfier_name'] = model.__class__.__name__
    model_comparison.loc[i, 'train_score'] = cv_result['train_score'].mean()
    model_comparison.loc[i, 'test_score'] = cv_result['test_score'].mean()

model_comparison
```

Out[56]:

	Classfier_name	train_score	test_score
0	RandomForestClassifier	0.989403	0.830283
1	LGBMClassifier	0.917931	0.848944
2	XGBClassifier	0.868176	0.852085
3	KerasClassifier	0.658177	0.65824

3 Pred basadas en descripciones

In [57]:
```python
m.loc[:, 'isGame'] = m['app_desc'].apply(lambda x: 1 if 'game' in x.lower() else 0)
m.loc[:, 'descLen'] = m['app_desc'].apply(lambda x: len(x.lower()))

df_train = m[['size_bytes', 'isNotFree', 'price', 'rating_count_previo', 'sup_devices.n
target = m['user_rating']

df_train = pd.get_dummies(df_train)

target = target.apply(lambda x: 1 if x > 3 else 0)
```

```
X_train, X_test, y_train, y_test = train_test_split(df_train.values, target, test_size=

print('X_train shape:', X_train.shape)
print('X_test shape:', X_test.shape)
```

```
X_train shape: (5733, 33)
X_test shape: (1434, 33)
```

In [58]:
```
model_comparison = pd.DataFrame(columns=['Classfier_name', 'train_score', 'test_score']

for i, model in enumerate(models):
    clf = model
    cv_result = cross_validate(model, X_train, y_train, cv=kfold, scoring='accuracy')
    model_comparison.loc[i, 'Classfier_name'] = model.__class__.__name__
    model_comparison.loc[i, 'train_score'] = cv_result['train_score'].mean()
    model_comparison.loc[i, 'test_score'] = cv_result['test_score'].mean()

model_comparison
```

Out[58]:

	Classfier_name	train_score	test_score
0	RandomForestClassifier	0.991104	0.843886
1	LGBMClassifier	0.934284	0.853304
2	XGBClassifier	0.877333	0.861329
3	KerasClassifier	0.446402	0.450742

In [59]:
```
# play a sound when whole notebook finished executing
import os
duration = 1  # second
freq = 1500  # Hz
os.system('play --no-show-progress --null --channels 1 synth %s sine %f' % (duration, f
```

Out[59]: 0