FOCS_FedericaFiorentini

November 26, 2019

1 Final project of *Foundations of Computer Science*

1.1 Federica Fiorentini

Study course: Data Science

Roll number: 807124

E-mail: f.fiorentini1@campus.unimib.it

[1]:
```
#libraries import

import numpy as np
import pandas as pd
import re
import time
```

[2]:
```
start_time=time.time()
```

[3]:
```
#data import

df = pd.read_csv(r'C:
 ↪\Users\FedericaFiorentini\Desktop\Università\1Anno\Foundations of␣
 ↪CS\googleplaystore.csv', sep=',')
df_review = pd.read_csv(r'C:
 ↪\Users\FedericaFiorentini\Desktop\Università\1Anno\Foundations of␣
 ↪CS\googleplaystore_user_reviews.csv')
```

[4]:
```
df.head()
```

[4]:

	App	Category	Rating	Reviews	Size	Installs	Type	Price	Content Rating
0	Photo Editor & Candy Camera & Grid & ScrapBook	ART_AND_DESIGN	4.1	159	19M	10,000+	Free	0	Everyone
1	Coloring book moana	ART_AND_DESIGN	3.9						
2	U Launcher Lite - FREE Live Cool Themes, Hide …	ART_AND_DESIGN	4.7						
3	Sketch - Draw & Paint	ART_AND_DESIGN	4.5						
4	Pixel Draw - Number Art Coloring Book	ART_AND_DESIGN	4.3						

```
1     967   14M     500,000+  Free     0       Everyone
2   87510  8.7M   5,000,000+  Free     0       Everyone
3  215644   25M  50,000,000+  Free     0           Teen
4     967  2.8M     100,000+  Free     0       Everyone

                      Genres      Last Updated         Current Ver  \
0               Art & Design   January 7, 2018               1.0.0
1  Art & Design;Pretend Play  January 15, 2018               2.0.0
2               Art & Design    August 1, 2018               1.2.4
3               Art & Design      June 8, 2018  Varies with device
4    Art & Design;Creativity     June 20, 2018                 1.1

    Android Ver
0  4.0.3 and up
1  4.0.3 and up
2  4.0.3 and up
3    4.2 and up
4    4.4 and up
```

[5]:
```
df_review.head()
```

[5]:
```
                     App                                Translated_Review  \
0  10 Best Foods for You  I like eat delicious food. That's I'm cooking …
1  10 Best Foods for You    This help eating healthy exercise regular basis
2  10 Best Foods for You                                                NaN
3  10 Best Foods for You         Works great especially going grocery store
4  10 Best Foods for You                                       Best idea us

  Sentiment  Sentiment_Polarity  Sentiment_Subjectivity
0  Positive                1.00                0.533333
1  Positive                0.25                0.288462
2       NaN                 NaN                     NaN
3  Positive                0.40                0.875000
4  Positive                1.00                0.300000
```

[6]:
```
df.loc[[10472]]
```

[6]:
```
                                        App Category  Rating Reviews  \
10472  Life Made WI-Fi Touchscreen Photo Frame      1.9    19.0    3.0M

         Size Installs Type     Price Content Rating             Genres  \
10472  1,000+     Free    0  Everyone            NaN  February 11, 2018

      Last Updated Current Ver Android Ver
10472       1.0.19  4.0 and up         NaN
```

Come si può notare, l'osservazione precedentemente visualizzata potrebbe creare problemi durante

l'utilizzo del dataset poiché la variabile *Category* non è coerente con le altre osservazioni.

Si procede, quindi, con l'eliminazione dell'osservazione stessa.

[7]:
```
df = df.drop(10472)
```

Successivamente viene effettuata un'analisi delle statistiche descrittive principali relative a tutte le variabili.

[8]:
```
df.describe(include='all')
```

[8]:

	App	Category	Rating	Reviews	Size	Installs
count	10840	10840	9366.000000	10840	10840	10840
unique	9659	33	NaN	6001	461	21
top	ROBLOX	FAMILY	NaN	0	Varies with device	1,000,000+
freq	9	1972	NaN	596	1695	1579
mean	NaN	NaN	4.191757	NaN	NaN	NaN
std	NaN	NaN	0.515219	NaN	NaN	NaN
min	NaN	NaN	1.000000	NaN	NaN	NaN
25%	NaN	NaN	4.000000	NaN	NaN	NaN
50%	NaN	NaN	4.300000	NaN	NaN	NaN
75%	NaN	NaN	4.500000	NaN	NaN	NaN
max	NaN	NaN	5.000000	NaN	NaN	NaN

	Type	Price	Content Rating	Genres	Last Updated
count	10839	10840	10840	10840	10840
unique	2	92	6	119	1377
top	Free	0	Everyone	Tools	August 3, 2018
freq	10039	10040	8714	842	326
mean	NaN	NaN	NaN	NaN	NaN
std	NaN	NaN	NaN	NaN	NaN
min	NaN	NaN	NaN	NaN	NaN
25%	NaN	NaN	NaN	NaN	NaN
50%	NaN	NaN	NaN	NaN	NaN
75%	NaN	NaN	NaN	NaN	NaN
max	NaN	NaN	NaN	NaN	NaN

	Current Ver	Android Ver
count	10832	10838
unique	2831	33
top	Varies with device	4.1 and up
freq	1459	2451
mean	NaN	NaN
std	NaN	NaN
min	NaN	NaN
25%	NaN	NaN
50%	NaN	NaN
75%	NaN	NaN

```
max                    NaN         NaN
```

[9]:
```
df_review.describe(include='all')
```

[9]:
```
                        App Translated_Review Sentiment  Sentiment_Polarity  \
count                 64295             37427     37432        37432.000000
unique                 1074             27994         3                 NaN
top     Angry Birds Classic              Good  Positive                 NaN
freq                    320               247     23998                 NaN
mean                    NaN               NaN       NaN            0.182146
std                     NaN               NaN       NaN            0.351301
min                     NaN               NaN       NaN           -1.000000
25%                     NaN               NaN       NaN            0.000000
50%                     NaN               NaN       NaN            0.150000
75%                     NaN               NaN       NaN            0.400000
max                     NaN               NaN       NaN            1.000000

        Sentiment_Subjectivity
count             37432.000000
unique                     NaN
top                        NaN
freq                       NaN
mean                  0.492704
std                   0.259949
min                   0.000000
25%                   0.357143
50%                   0.514286
75%                   0.650000
max                   1.000000
```

[10]:
```
df.dtypes
```

[10]:
```
App               object
Category          object
Rating           float64
Reviews           object
Size              object
Installs          object
Type              object
Price             object
Content Rating    object
Genres            object
Last Updated      object
Current Ver       object
Android Ver       object
dtype: object
```

[11]:
```
df_review.dtypes
```

[11]:
```
App                        object
Translated_Review          object
Sentiment                  object
Sentiment_Polarity        float64
Sentiment_Subjectivity    float64
dtype: object
```

1.1.1 Exercise 1

Convert the app sizes to a number Dopo aver verificato l'assenza di missing value nella variabile 'Size' si procede nel seguente modo:

- si crea una nuova variabile NewSize. In questo modo, non modificando la variabile originaria 'Size' sarà possibile confrontarle.
- per tutte le app in cui 'Size' = "Varies with device" la nuova variabile 'NewSize' sarà missing.
- vengono rimosse le lettere 'k' e 'M' e i valori vengono trasformati da object a float.
- i valori ottenuti vengono moltiplicati per 1000 o 1000000, in base alla presenza rispettivamente di 'k' o 'M' nella variabile originaria 'Size'

Per verificare che l'operazione sia andata a buon fine, si effetuno alcuni check.

[12]:
```
df.Size.head()
```

[12]:
```
0     19M
1     14M
2    8.7M
3     25M
4    2.8M
Name: Size, dtype: object
```

[13]:
```
df['Size'].isnull().sum() #NA check
```

[13]:
```
0
```

[14]:
```
df['Size_New'] = (df[df.Size != 'Varies with device'].Size.replace(r'[kM]+$',␣
↪'', regex=True).astype(float) * \
   df[df.Size != 'Varies with device'].Size.str.extract(r'[\d\.]+([kM]+)',␣
↪expand=False).fillna(1).replace(['k','M'], [10**3, 10**6]).astype(int))

df[['Size', 'Size_New']]
```

[14]:

	Size	Size_New
0	19M	19000000.0
1	14M	14000000.0
2	8.7M	8700000.0
3	25M	25000000.0

```
4                    2.8M   2800000.0
…                     …         …
10836                 53M  53000000.0
10837                3.6M   3600000.0
10838                9.5M   9500000.0
10839  Varies with device         NaN
10840                 19M  19000000.0

[10840 rows x 2 columns]
```

[15]:
```
print('Osservazioni con la variabile originale Size uguale a "varies with␣
 ↪device":',len(df.Size.loc[df.Size == 'Varies with device']))

print('Missing value della variabile Size new creata:', df['Size_New'].isnull().
 ↪sum())
```

```
Osservazioni con la variabile originale Size uguale a "varies with device": 1695
Missing value della variabile Size new creata: 1695
```

Si nota che il numero di osservazioni aventi nel dataset originale la variabile Size pari a "Varies with device" sono 1695, pari al numero di missing value corrispondenti alla variabile create Size_New. Questo significa che la modalità varies with device è stata sostituita con successo da un NA, come indicato inizialmente.

1.1.2 Exercise 2

Convert the number of installs to a number Inizialmente si osserva la struttura della varibile 'Installs'. Successivamente si crea una nuova variabile 'NewInstalls' in cui sono stati rimossi tutti i caratteri non numerici tramite il comando replace di pandas e la funzione regex ' D' ed, in seguito, ne è stato modificato il type da object a numerico.

[16]:
```
df[['Installs']].head()
```

[16]:
```
       Installs
0       10,000+
1      500,000+
2    5,000,000+
3   50,000,000+
4      100,000+
```

[17]:
```
df.Installs.unique() #modalità della variabile categorica
```

[17]:
```
array(['10,000+', '500,000+', '5,000,000+', '50,000,000+', '100,000+',
       '50,000+', '1,000,000+', '10,000,000+', '5,000+', '100,000,000+',
       '1,000,000,000+', '1,000+', '500,000,000+', '50+', '100+', '500+',
       '10+', '1+', '5+', '0+', '0'], dtype=object)
```

[18]:
```
df['Installs_New'] = (df.Installs.replace('\\D', '', regex=True).astype(float))

df[['Installs', 'Installs_New']].head()
```

[18]:
```
       Installs  Installs_New
0       10,000+       10000.0
1      500,000+      500000.0
2    5,000,000+     5000000.0
3   50,000,000+    50000000.0
4      100,000+      100000.0
```

[19]:
```
df.Installs_New.unique()
```

[19]:
```
array([1.e+04, 5.e+05, 5.e+06, 5.e+07, 1.e+05, 5.e+04, 1.e+06, 1.e+07,
       5.e+03, 1.e+08, 1.e+09, 1.e+03, 5.e+08, 5.e+01, 1.e+02, 5.e+02,
       1.e+01, 1.e+00, 5.e+00, 0.e+00])
```

1.1.3 Exercise 3

Transform "Varies with device" into a missing value Dopo aver trasformato in missing value le osservazioni per cui la variabile 'Size' era pari a 'Varies with device', si effettua un check sfruttando la nuova variabile creata al punto 1. Nella creazione della variabile 'NewSize', infatti, le osservazioni per cui 'Size' era pari a 'Varies with device' erano già state etichettate come NaN. Si verifica, quindi, di avere lo stesso numero di missing value in 'Size' e 'NewSize'.

[20]:
```
df.Size.loc[df.Size == 'Varies with device']
#per vedere gli elementi che hanno df.Size=='varies with device'
```

[20]:
```
37       Varies with device
42       Varies with device
52       Varies with device
67       Varies with device
68       Varies with device
                ...
10713    Varies with device
10725    Varies with device
10765    Varies with device
10826    Varies with device
10839    Varies with device
Name: Size, Length: 1695, dtype: object
```

[21]:
```
df.loc[[37]]
```

[21]:
```
                    App        Category  Rating Reviews                Size  \
37  Floor Plan Creator  ART_AND_DESIGN     4.1   36639  Varies with device

      Installs  Type Price Content Rating        Genres   Last Updated  \
```

```
37  5,000,000+  Free     0       Everyone  Art & Design  July 14, 2018

           Current Ver   Android Ver  Size_New  Installs_New
37  Varies with device  2.3.3 and up       NaN     5000000.0
```

[22]:
```
df = df.replace(to_replace = 'Varies with device', value = np.nan)
```

[23]:
```
df.loc[[37]]
```

[23]:
```
                    App        Category  Rating Reviews Size    Installs  Type  \
37  Floor Plan Creator  ART_AND_DESIGN     4.1   36639  NaN  5,000,000+  Free

   Price Content Rating        Genres   Last Updated Current Ver  \
37     0       Everyone  Art & Design  July 14, 2018         NaN

     Android Ver  Size_New  Installs_New
37  2.3.3 and up       NaN     5000000.0
```

Per quanto riguarda le 1695 variabili hanno la variabile Size pari a Varies with device (cambia da dispositivo a dispositivo), è stata sostituita questa modalità con nan. In particolare, si può vedere sopra che la 37 osservazione inizialmente aveva varies with device e dopo è diventato nan.

1.1.4 Exercise 4

Convert Current Ver and Android Ver into a dotted number Si utilizzano i comandi di regex per sostituire i caratteri non numerici all'interno delle variabili prese in considerazione.

[24]:
```
df[['Current Ver', 'Android Ver']].head()
```

[24]:
```
  Current Ver   Android Ver
0       1.0.0  4.0.3 and up
1       2.0.0  4.0.3 and up
2       1.2.4  4.0.3 and up
3         NaN    4.2 and up
4         1.1    4.4 and up
```

Android ver

[25]:
```
df['Android Ver'].unique()
```

[25]:
```
array(['4.0.3 and up', '4.2 and up', '4.4 and up', '2.3 and up',
       '3.0 and up', '4.1 and up', '4.0 and up', '2.3.3 and up', nan,
       '2.2 and up', '5.0 and up', '6.0 and up', '1.6 and up',
       '1.5 and up', '2.1 and up', '7.0 and up', '5.1 and up',
       '4.3 and up', '4.0.3 - 7.1.1', '2.0 and up', '3.2 and up',
       '4.4W and up', '7.1 and up', '7.0 - 7.1.1', '8.0 and up',
       '5.0 - 8.0', '3.1 and up', '2.0.1 and up', '4.1 - 7.1.1',
       '5.0 - 6.0', '1.0 and up', '2.2 - 7.1.1', '5.0 - 7.1.1'],
```

```
      dtype=object)
```

[26]:
```
df['Android Ver'].replace('[a-z]+|[A-z]+', value = '', regex = True, inplace =␣
↪True)

df[['Android Ver']].head()
```

[26]:
```
  Android Ver
0     4.0.3
1     4.0.3
2     4.0.3
3       4.2
4       4.4
```

[27]:
```
#verifichiamo le modalità della variabile Android Ver
df['Android Ver'].unique()
```

[27]:
```
array(['4.0.3  ', '4.2  ', '4.4  ', '2.3  ', '3.0  ', '4.1  ', '4.0  ',
       '2.3.3  ', nan, '2.2  ', '5.0  ', '6.0  ', '1.6  ', '1.5  ',
       '2.1  ', '7.0  ', '5.1  ', '4.3  ', '4.0.3 - 7.1.1', '2.0  ',
       '3.2  ', '7.1  ', '7.0 - 7.1.1', '8.0  ', '5.0 - 8.0', '3.1  ',
       '2.0.1  ', '4.1 - 7.1.1', '5.0 - 6.0', '1.0  ', '2.2 - 7.1.1',
       '5.0 - 7.1.1'], dtype=object)
```

Current Ver

[28]:
```
df['Current Ver'].replace('[a-z]+|[A-z]+', value = '', regex = True, inplace =␣
↪True)

df[['Current Ver']].head()
```

[28]:
```
  Current Ver
0       1.0.0
1       2.0.0
2       1.2.4
3         NaN
4         1.1
```

[29]:
```
#verifichiamo le modalità della variabile Current Ver
df['Current Ver'].unique()
```

[29]:
```
array(['1.0.0', '2.0.0', '1.2.4', …, '1.0.612928', '0.3.4', '2.0.148.0'],
      dtype=object)
```

1.1.5 Exercise 5

Remove the duplicates Si nota che nel dataset le righe esattamente duplicate sono 483.

In questo caso, però, due righe sono duplicate se si riferiscono alla stessa App. Si verifica che nel dataset in questione sono presenti 1181 righe duplicate secondo questa definizione.

Come criterio per eliminare i duplicati è stata considerato il numero di review e, in particolare, si conserva la riga avente il numero di review più alto in quanto più recente.

[30]:
```
#identifichiamo le righe esattamente duplicate

print('Number of duplicated rows: ', df[df.duplicated()].shape[0])
```

```
Number of duplicated rows:  483
```

[31]:
```
#app duplicate

print('Number of duplicated Apps: ', df[df.duplicated('App')].shape[0])
```

```
Number of duplicated Apps:  1181
```

Si procede ordinando il dataset secondo il numero di Review (colonna *Reviews*) e eliminando tutte le righe della stessa app eccetto la prima (quella con più review).

Inizialmente, però, viene trasformata la variabile Reviews in numerica.

[32]:
```
df['Reviews'] = df['Reviews'].astype(int)

df = df.sort_values('Reviews', ascending = False).drop_duplicates('App', keep =␣
↪'first').reset_index(drop = True)
```

[33]:
```
#verifica che sono stati eliminati

print('Number of duplicated Apps: ', df[df.duplicated('App')].shape[0])
```

```
Number of duplicated Apps:  0
```

1.1.6 Exercise 6

For each category, compute the number of apps Per calcolare il numero di App appartenenti ad ogni categoria presente nel dataset si procede utilizzando Pandas e raggruppando le osservazioni per la variabile **Category** e ordinando i valori in ordine decrescente.

[34]:
```
pd.DataFrame(df.groupby('Category').size().sort_values(ascending = False))
```

[34]:

Category	0
FAMILY	1875
GAME	945
TOOLS	829
BUSINESS	420
MEDICAL	395
PERSONALIZATION	376

```
PRODUCTIVITY          374
LIFESTYLE             369
FINANCE               345
SPORTS                325
COMMUNICATION         315
HEALTH_AND_FITNESS    288
PHOTOGRAPHY           281
NEWS_AND_MAGAZINES    254
SOCIAL                239
BOOKS_AND_REFERENCE   222
TRAVEL_AND_LOCAL      219
SHOPPING              202
DATING                170
VIDEO_PLAYERS         164
MAPS_AND_NAVIGATION   131
FOOD_AND_DRINK        112
EDUCATION             107
ENTERTAINMENT          87
AUTO_AND_VEHICLES      85
LIBRARIES_AND_DEMO     84
WEATHER                79
HOUSE_AND_HOME         73
EVENTS                 64
ART_AND_DESIGN         61
PARENTING              60
COMICS                 56
BEAUTY                 53
```

1.1.7 Exercise 7

For each category, compute the average rating Anche in questo caso si procede con Pandas, in particolare raggruppando le osservazioni per la variabile **Category**, per poi calcolare la media della variabile **Rating** per ogni categoria.

[35]:
```
pd.DataFrame(df.groupby('Category')['Rating'].mean())
```

[35]:

Category	Rating
ART_AND_DESIGN	4.359322
AUTO_AND_VEHICLES	4.190411
BEAUTY	4.278571
BOOKS_AND_REFERENCE	4.344970
BUSINESS	4.098479
COMICS	4.181481
COMMUNICATION	4.121484
DATING	3.980451
EDUCATION	4.354717

```
ENTERTAINMENT        4.129885
EVENTS               4.435556
FAMILY               4.183576
FINANCE              4.115563
FOOD_AND_DRINK       4.171277
GAME                 4.244432
HEALTH_AND_FITNESS   4.243033
HOUSE_AND_HOME       4.140984
LIBRARIES_AND_DEMO   4.178125
LIFESTYLE            4.093355
MAPS_AND_NAVIGATION  4.036441
MEDICAL              4.165862
NEWS_AND_MAGAZINES   4.121569
PARENTING            4.300000
PERSONALIZATION      4.332215
PHOTOGRAPHY          4.155894
PRODUCTIVITY         4.183389
SHOPPING             4.230556
SOCIAL               4.247291
SPORTS               4.216154
TOOLS                4.040278
TRAVEL_AND_LOCAL     4.069519
VIDEO_PLAYERS        4.044966
WEATHER              4.243056
```

1.1.8 Exercise 8

Create two dataframes: one for the genres and one bridging apps and genres. So that, for instance, the app Pixel Draw - Number Art Coloring Book appears twice in the bridging table, once for Art & Design, once for Creativity Innanzitutto, come viene visualizzato successivamente, si nota che alcune App presentano due generi, divisi da un ;. In alcuni casi potrebbero essere uguali mentre in altri due generi completamente diversi.

Si identificano, quindi, le App aventi due generi e si splittano in due colonne diverse, Genre e Genre_2, creando così una nuova tabella Genre_Split.

[36]:
```
df[['Genres']]
```

[36]:
```
                Genres
0               Social
1        Communication
2               Social
3        Communication
4             Strategy
…                  …
9654  Auto & Vehicles
9655  Personalization
9656            Arcade
```

```
9657  Personalization
9658           Sports

[9659 rows x 1 columns]
```

Ci sono app che hanno più di un genere, separati da ; e in alcuni casi hanno il genere stesso oppure due generi diversi.

[37]:
```
df.Genres.loc[df.Genres == 'Role Playing;Education']
```

[37]:
```
9051    Role Playing;Education
Name: Genres, dtype: object
```

[38]:
```
pd.DataFrame(df.loc[8956]) #due generi diversi
```

[38]:
```
                         8956
App                    CI Crew
Category              FAMILY
Rating                   NaN
Reviews                    1
Size                     30M
Installs                100+
Type                    Free
Price                      0
Content Rating       Everyone
Genres          Entertainment
Last Updated    March 8, 2018
Current Ver               0.7
Android Ver           2.3.3
Size_New                3e+07
Installs_New              100
```

[39]:
```
df.Genres.loc[df.Genres == 'Education;Education'] #due generi uguali
```

[39]:
```
56      Education;Education
1107    Education;Education
1202    Education;Education
1271    Education;Education
1290    Education;Education
1935    Education;Education
2470    Education;Education
2510    Education;Education
2761    Education;Education
3024    Education;Education
3946    Education;Education
4111    Education;Education
4356    Education;Education
```

```
4462    Education;Education
4511    Education;Education
4598    Education;Education
4648    Education;Education
4793    Education;Education
4990    Education;Education
5193    Education;Education
5250    Education;Education
5280    Education;Education
5429    Education;Education
5749    Education;Education
5770    Education;Education
6018    Education;Education
6030    Education;Education
6039    Education;Education
6307    Education;Education
6397    Education;Education
7108    Education;Education
7169    Education;Education
7339    Education;Education
8673    Education;Education
8676    Education;Education
9037    Education;Education
Name: Genres, dtype: object
```

[40]:
```
pd.DataFrame(df.loc[396])
```

[40]:

	396
App	Google Now Launcher
Category	TOOLS
Rating	4.2
Reviews	857215
Size	7.9M
Installs	100,000,000+
Type	Free
Price	0
Content Rating	Everyone
Genres	Tools
Last Updated	December 7, 2017
Current Ver	1.4.
Android Ver	4.1 - 7.1.1
Size_New	7.9e+06
Installs_New	1e+08

Quindi come prima cosa vanno splittati se hanno due generi.

[41]:
```
Genres_Split = pd.DataFrame([Genres.split(';') for Genres in df.Genres]).
↪rename(columns = {0: 'Genre', 1:'Genre_2'})
```

[42]:
```
Genres_Split.head(20)
```

[42]:
```
                     Genre Genre_2
0                   Social    None
1            Communication    None
2                   Social    None
3            Communication    None
4                 Strategy    None
5                    Tools    None
6                   Arcade    None
7   Video Players & Editors    None
8                    Tools    None
9                 Strategy    None
10                  Casual    None
11           Communication    None
12                  Social    None
13                   Tools    None
14                  Casual    None
15                  Sports    None
16                   Tools    None
17           Communication    None
18                   Tools    None
19        News & Magazines    None
```

[43]:
```
pd.DataFrame(Genres_Split.loc[396])
```

[43]:
```
            396
Genre     Tools
Genre_2    None
```

[44]:
```
pd.DataFrame(Genres_Split.loc[8956])
```

[44]:
```
                  8956
Genre    Entertainment
Genre_2           None
```

Per creare una connessione, facciamo un subset del dataframe iniziale considerando solamente la colonna Apps.

[45]:
```
App_DF = pd.DataFrame(df.App)

App_DF.head()
```

[45]:
```
                                   App
0                             Facebook
1                   WhatsApp Messenger
2                            Instagram
3  Messenger – Text and Video Chat for Free
4                       Clash of Clans
```

A questo punto si effettua un merge tra la tabella App_DF e la tabella Genres_Split creata in precedenza.

[46]:
```
App_Genre = pd.merge(App_DF, Genres_Split, right_index = True, left_index =␣
↪True).melt(id_vars = ['App'], value_name = 'Genre').drop('variable', axis =␣
↪1).dropna()

App_Genre.head()
```

[46]:
```
                                   App          Genre
0                             Facebook         Social
1                   WhatsApp Messenger  Communication
2                            Instagram         Social
3  Messenger – Text and Video Chat for Free  Communication
4                       Clash of Clans       Strategy
```

Possiamo verificare che l'app *Pixel Draw - Number Art Coloring Book* che prima aveva due generi (Art & Design, Creativity) compaia due volte nel dataframe creato, una volta per ogni genere.

[47]:
```
App_Genre.loc[App_Genre.App == 'Pixel Draw - Number Art Coloring Book']
```

[47]:
```
                                    App         Genre
4831   Pixel Draw - Number Art Coloring Book  Art & Design
14490  Pixel Draw - Number Art Coloring Book    Creativity
```

1.1.9 Exercise 9

For each genre, create a new column of the original dataframe. The new columns must have boolean values (True if the app has a given genre) Per svolgere questo task vengono utilizzate due funzioni di Pandas, *concat* e *get_dummies* che, rispettivamente, permettono di concatenare due dataset e trasformare una variabile categorica in variabile binaria.

[48]:
```
df.shape
```

[48]:
```
(9659, 15)
```

[49]:
```
df = pd.concat([df, pd.get_dummies(Genres_Split['Genre'], dtype = 'bool')],␣
↪axis = 1)
df = pd.concat([df, pd.get_dummies(Genres_Split['Genre_2'], dtype = 'bool')],␣
↪axis = 1)
```

```
df.head()
```

[49]:

```
                                     App        Category  Rating   Reviews  \
0                               Facebook          SOCIAL     4.1  78158306
1                       WhatsApp Messenger   COMMUNICATION     4.4  69119316
2                              Instagram          SOCIAL     4.5  66577446
3  Messenger – Text and Video Chat for Free   COMMUNICATION     4.0  56646578
4                           Clash of Clans            GAME     4.6  44893888

  Size        Installs  Type Price Content Rating         Genres  … Trivia  \
0  NaN  1,000,000,000+  Free     0           Teen         Social  …  False
1  NaN  1,000,000,000+  Free     0       Everyone  Communication  …  False
2  NaN  1,000,000,000+  Free     0           Teen         Social  …  False
3  NaN  1,000,000,000+  Free     0       Everyone  Communication  …  False
4  98M    100,000,000+  Free     0   Everyone 10+       Strategy  …  False

  Video Players & Editors Weather   Word  Action & Adventure  Brain Games  \
0                   False   False  False               False        False
1                   False   False  False               False        False
2                   False   False  False               False        False
3                   False   False  False               False        False
4                   False   False  False               False        False

   Creativity  Education  Music & Video  Pretend Play
0       False      False          False         False
1       False      False          False         False
2       False      False          False         False
3       False      False          False         False
4       False      False          False         False

[5 rows x 69 columns]
```

[50]:

```
df.shape
```

[50]:

```
(9659, 69)
```

Il dataframe è diventato di 69 colonne rispetto alle 15 iniziali.

Facciamo il solito check con l'app di prima.

[51]:

```
pd.DataFrame(df.loc[df.App == 'Pixel Draw - Number Art Coloring Book']).
↪transpose()[17:71]
```

[51]:

```
                        4831
Arcade                 False
Art & Design            True
Auto & Vehicles        False
Beauty                 False
```

```
Board                    False
Books & Reference        False
Business                 False
Card                     False
Casino                   False
Casual                   False
Comics                   False
Communication            False
Dating                   False
Education                False
Educational              False
Entertainment            False
Events                   False
Finance                  False
Food & Drink             False
Health & Fitness         False
House & Home             False
Libraries & Demo         False
Lifestyle                False
Maps & Navigation        False
Medical                  False
Music                    False
Music & Audio            False
News & Magazines         False
Parenting                False
Personalization          False
Photography              False
Productivity             False
Puzzle                   False
Racing                   False
Role Playing             False
Shopping                 False
Simulation               False
Social                   False
Sports                   False
Strategy                 False
Tools                    False
Travel & Local           False
Trivia                   False
Video Players & Editors  False
Weather                  False
Word                     False
Action & Adventure       False
Brain Games              False
Creativity                True
Education                False
Music & Video            False
```

```
Pretend Play                False
```

Si vede che i due generi che ha l'app sono uguali a True mentre gli altri sono tutti uguali a False.

1.1.10 Exercise 10

For each genre, compute the average rating. What is the genre with highest average? Per calcolare il punteggio medio per ogni genere, innanzitutto si mergiano il dataset iniziale con l'app creata in precedenze App_Genre per evitare le problematiche riguardo il doppio genere. Successivamente, tramite la funzione groupby si calcola la media della variabile Rating raggruppando per la variabile Genre.

Inifine, tramite la funzione idxmax si estrae il genere con il rating medio più alto che risulta essere **Events**.

[52]:
```
#uniamo la tabella principale (df) con App_Genre

df_rating = App_Genre.merge(df, on = 'App')
df_rating[['App', 'Rating', 'Genre']].head()
```

[52]:

	App	Rating	Genre
0	Facebook	4.1	Social
1	WhatsApp Messenger	4.4	Communication
2	Instagram	4.5	Social
3	Messenger – Text and Video Chat for Free	4.0	Communication
4	Clash of Clans	4.6	Strategy

[53]:
```
mean_rating = pd.DataFrame(df_rating.groupby(['Genre'])['Rating'].mean().
↪sort_values(ascending = False))
```

[54]:
```
#rating più alto

print('The genre with highest average is', mean_rating.idxmax()[0], ' with a␣
↪rating equal to', round(mean_rating.max()[0],2))
```

```
The genre with highest average is Events  with a rating equal to 4.44
```

1.1.11 Exercise 11

For each app, compute the approximate income, obtain as a product of number of installs and price.

[55]:
```
df.Price.unique()
```

[55]:
```
array(['0', '$6.99', '$0.99', '$2.99', '$1.99', '$2.49', '$4.99', '$5.99',
       '$4.49', '$9.99', '$3.99', '$1.49', '$3.95', '$7.99', '$3.49',
       '$8.99', '$13.99', '$19.99', '$11.99', '$12.99', '$2.90', '$17.99',
       '$399.99', '$29.99', '$14.99', '$2.95', '$4.77', '$24.99', '$3.90',
       '$2.50', '$3.28', '$1.20', '$2.59', '$9.00', '$1.59', '$1.00',
       '$5.49', '$18.99', '$299.99', '$1.97', '$400.00', '$16.99',
```

```
'$389.99', '$33.99', '$10.00', '$10.99', '$4.84', '$37.99',
'$1.61', '$8.49', '$4.60', '$79.99', '$4.29', '$1.70', '$1.50',
'$1.29', '$19.40', '$3.08', '$379.99', '$2.56', '$15.46', '$7.49',
'$4.59', '$2.00', '$14.00', '$6.49', '$15.99', '$74.99', '$3.88',
'$3.02', '$39.99', '$89.99', '$5.00', '$1.75', '$1.26', '$2.60',
'$19.90', '$4.80', '$1.76', '$46.99', '$3.04', '$4.85', '$30.99',
'$3.61', '$154.99', '$394.99', '$109.99', '$1.96', '$1.04',
'$28.99', '$25.99', '$200.00'], dtype=object)
```

E' presente il simbolo del dollaro che va sostituito perchè nel calcolo potrebbe dare fastidio. Dopo aver rimosso il simbolo e convertito la variabile *Price* in numerica, si calcola il guadagno moltiplicando il numero di Install per il prezzo dell'App. Si mostrano le prime 10 app che hanno ottenuto un guadagno maggiore.

[56]:
```
df['Price_New'] = df['Price'].str.replace('$', '', regex = True).astype(float)

df[['Price', 'Price_New']]
```

[56]:

	Price	Price_New
0	0	0.00
1	0	0.00
2	0	0.00
3	0	0.00
4	0	0.00
…	…	…
9654	0	0.00
9655	0	0.00
9656	$0.99	0.99
9657	$0.99	0.99
9658	0	0.00

[9659 rows x 2 columns]

[57]:
```
df['Income'] = df['Installs_New'] * df['Price_New']

df[['App', 'Income']].sort_values(by = 'Income', ascending = False).head(10)
```

[57]:

	App	Income
172	Minecraft	69900000.0
3986	I am rich	39999000.0
4441	I Am Rich Premium	19999500.0
660	Hitman Sniper	9900000.0
730	Grand Theft Auto: San Andreas	6990000.0
2593	Sleep as Android Unlock	5990000.0
1954	Facetune - For Free	5990000.0
1531	DraStic DS Emulator	4990000.0
5648	I'm Rich - Trump Edition	4000000.0

```
5031                    I'm rich   3999900.0
```

1.1.12 Exercise 12

For each app, compute its minimum and maximum Sentiment_polarity

[58]:
```
df_review.head(10)
```

[58]:
```
                      App                                 Translated_Review  \
0  10 Best Foods for You  I like eat delicious food. That's I'm cooking …
1  10 Best Foods for You    This help eating healthy exercise regular basis
2  10 Best Foods for You                                               NaN
3  10 Best Foods for You         Works great especially going grocery store
4  10 Best Foods for You                                    Best idea us
5  10 Best Foods for You                                        Best way
6  10 Best Foods for You                                         Amazing
7  10 Best Foods for You                                             NaN
8  10 Best Foods for You                           Looking forward app,
9  10 Best Foods for You              It helpful site ! It help foods get !

  Sentiment  Sentiment_Polarity  Sentiment_Subjectivity
0  Positive                1.00                0.533333
1  Positive                0.25                0.288462
2       NaN                 NaN                     NaN
3  Positive                0.40                0.875000
4  Positive                1.00                0.300000
5  Positive                1.00                0.300000
6  Positive                0.60                0.900000
7       NaN                 NaN                     NaN
8   Neutral                0.00                0.000000
9   Neutral                0.00                0.000000
```

Si estrae dal dataframe df_review inizialmente il valore minimo della variabile *Sentiment_Polarity* e successivamente il valore massimo.

Infine, tramite la funzione merge vengono unite le due tabelle create con l'obiettivo di avere, per ogni App il valore mimimo e massimo della variabile Sentiment_Polarity.

[59]:
```
sentiment_min = pd.DataFrame(df_review.groupby('App')['Sentiment_Polarity'].
↪min())
sentiment_max = pd.DataFrame(df_review.groupby('App')['Sentiment_Polarity'].
↪max())
```

[60]:
```
sentiment_min.merge(sentiment_max, on = 'App', suffixes = [' (min)', ' (max)']).
↪head(10)
```

[60]:
```
                                        Sentiment_Polarity (min)  \
App
```

```
10 Best Foods for You                                          -0.800000
104    -                                                -0.112500
11st                                                           -1.000000
1800 Contacts - Lens Store                                     -0.300000
1LINE - One Line with One Touch                                -0.825000
2018Emoji Keyboard  Emoticons Lite -sticker&gif                -0.800000
21-Day Meditation Experience                                   -0.265625
2Date Dating App, Love and matching                            -0.645833
2GIS: directory & navigator                                    -0.375000
2RedBeans                                                      -0.800000

                                                 Sentiment_Polarity (max)
App
10 Best Foods for You                                           1.000000
104    -                                                 0.910000
11st                                                            1.000000
1800 Contacts - Lens Store                                      0.838542
1LINE - One Line with One Touch                                 1.000000
2018Emoji Keyboard  Emoticons Lite -sticker&gif                 1.000000
21-Day Meditation Experience                                    0.587500
2Date Dating App, Love and matching                             1.000000
2GIS: directory & navigator                                     1.000000
2RedBeans                                                       1.000000
```

[61]:
```
end_time = time.time()
```

[62]:
```
print("Program executed in %s seconds" % (time.time()-start_time))
```

```
Program executed in 2.3119993209838867 seconds
```