Final Project: foundations of Computer Science

CdLM Data Science - University of Milano-Bicocca

Davide Lagano

In [1]:

```
# Import libraries

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

In [2]:

```
# Import datasets

gPlay=pd.read_csv("C:/Users/davla/Desktop/Computer_Science/exam/googleplaystore.csv", sep="
gPlay_review=pd.read_csv("C:/Users/davla/Desktop/Computer_Science/exam/googleplaystore_user
```

In [3]:

```
# Overview of gPlay dataset

gPlay.head()
```

Out[3]:

	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	967	14M	500,000+	Free	0	Everyone
2	U Launcher Lite – FREE Live Cool Themes, Hide ...	ART_AND_DESIGN	4.7	87510	8.7M	5,000,000+	Free	0	Everyone
3	Sketch - Draw & Paint	ART_AND_DESIGN	4.5	215644	25M	50,000,000+	Free	0	Teen
4	Pixel Draw - Number Art Coloring Book	ART_AND_DESIGN	4.3	967	2.8M	100,000+	Free	0	Everyone

1. Convert the app size to a number

In [4]:

```
# Visualize unique values of app size

gPlay['Size'].unique()
```

Out[4]:

```
array(['19M', '14M', '8.7M', '25M', '2.8M', '5.6M', '29M', '33M', '3.1M',
       '28M', '12M', '20M', '21M', '37M', '2.7M', '5.5M', '17M', '39M',
       '31M', '4.2M', '7.0M', '23M', '6.0M', '6.1M', '4.6M', '9.2M',
       '5.2M', '11M', '24M', 'Varies with device', '9.4M', '15M', '10M',
       '1.2M', '26M', '8.0M', '7.9M', '56M', '57M', '35M', '54M', '201k',
       '3.6M', '5.7M', '8.6M', '2.4M', '27M', '2.5M', '16M', '3.4M',
       '8.9M', '3.9M', '2.9M', '38M', '32M', '5.4M', '18M', '1.1M',
       '2.2M', '4.5M', '9.8M', '52M', '9.0M', '6.7M', '30M', '2.6M',
       '7.1M', '3.7M', '22M', '7.4M', '6.4M', '3.2M', '8.2M', '9.9M',
       '4.9M', '9.5M', '5.0M', '5.9M', '13M', '73M', '6.8M', '3.5M',
       '4.0M', '2.3M', '7.2M', '2.1M', '42M', '7.3M', '9.1M', '55M',
       '23k', '6.5M', '1.5M', '7.5M', '51M', '41M', '48M', '8.5M', '46M',
       '8.3M', '4.3M', '4.7M', '3.3M', '40M', '7.8M', '8.8M', '6.6M',
       '5.1M', '61M', '66M', '79k', '8.4M', '118k', '44M', '695k', '1.6
M',
       '6.2M', '18k', '53M', '1.4M', '3.0M', '5.8M', '3.8M', '9.6M',
       '45M', '63M', '49M', '77M', '4.4M', '4.8M', '70M', '6.9M', '9.3M',
```

In [5]:

```
# compile(r""): The purpose of it is to compile the regex pattern which will be used later
# (?P<name>regex) Captures the text matched by “regex” into the group “name”.

app_size = re.compile(r'(?P<first_value>\d+(\.\d+)?)(?P<second_value>[kKmM])')

# function that convert the <second_value> to numeric value.

def second_value_to_num(nume):
    if nume.upper()=='K':
        return 1000
    if nume.upper()=='M':
        return 1000000

# this function initially groups 'second_value' (returns a tuple containing all the subgrou
# then usign 'second_value_to_num' transform 'second_value' and transform 'first_value' to
# thus moltiply f (first) and s (second) to obtain the app size as a number.
# all this throw the 'search' function.

def conversion(final):
    act=app_size.search(final)
    if act:
        dim = act.group('second_value')
        s= second_value_to_num(dim)
        f= float(act.group('first_value'))
        return int(f*s)
    else:
        np.nan

gPlay['New_Size']=gPlay['Size'].apply(conversion)
gPlay[['Size','New_Size']].head()
```

Out[5]:

	Size	New_Size
0	19M	19000000.0
1	14M	14000000.0
2	8.7M	8700000.0
3	25M	25000000.0
4	2.8M	2800000.0

In [6]:

```
gPlay[gPlay['Size']=='1,000+']
```

Out[6]:

	App	Category	Rating	Reviews	Size	Installs	Type	Price	Content Rating	Ge
10472	Life Made WI-Fi Touchscreen Photo Frame	1.9	19.0	3.0M	1,000+	Free	0	Everyone	NaN	Febı 11, :

In [7]:

```
# I decided to delete the row 10472 of the dataset because different from the other.
gPlay=gPlay.drop(10472)
```

2. Convert the number of installs to a number

In [8]:

```
gPlay['Installs'].unique()
```

Out[8]:

```
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)
```

In [9]:

```
# I used the replace function to delete the dot and the plus. Then I transformed the comlum

gPlay['New_Installs'] = gPlay['Installs'].str.replace(",","").str.replace("+","").astype(ir
```

In [10]:

```
gPlay.head()
```

Out[10]:

	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	967	14M	500,000+	Free	0	Everyone
2	U Launcher Lite – FREE Live Cool Themes, Hide ...	ART_AND_DESIGN	4.7	87510	8.7M	5,000,000+	Free	0	Everyone
3	Sketch - Draw & Paint	ART_AND_DESIGN	4.5	215644	25M	50,000,000+	Free	0	Teen
4	Pixel Draw - Number Art Coloring Book	ART_AND_DESIGN	4.3	967	2.8M	100,000+	Free	0	Everyone

3. Transform "Varies with device" into a missing value

In [11]:

```
#check where and how "Varies with device" there are

gPlay[gPlay.isin(['Varies with device'])].dropna(how='all').count()
```

Out[11]:

```
App                  0
Category             0
Rating               0
Reviews              0
Size              1695
Installs             0
Type                 0
Price                0
Content Rating       0
Genres               0
Last Updated         0
Current Ver       1459
Android Ver       1362
New_Size             0
New_Installs         0
dtype: int64
```

In [12]:

```
# sum them
gPlay[gPlay.isin(['Varies with device'])].dropna(how='all').count().sum()
```

Out[12]:

```
4516
```

In [13]:

```
# count the missing values
gPlay.isnull().sum().sum()
```

Out[13]:

```
3180
```

In [14]:

```
# replace 'Varies with device' into missing values

gPlay.replace('Varies with device', np.nan, inplace = True)
```

In [15]:

```
# check if there are any 'Varies with device'

gPlay[gPlay.isin(['Varies with device'])].dropna(how='all').count().sum()
```

Out[15]:

```
0
```

In [16]:

```
# checking throw counting the missing values. 4516 + 3180 =7696

gPlay.isnull().sum().sum()
```

Out[16]:

```
7696
```

4. Convert Current Ver and Android Ver into a dotted number (e.g. 4.0.3 of 4.2)

In [17]:

```
gPlay[['Current Ver','Android Ver']].head()
```

Out[17]:

	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

In [18]:

```
pattern=re.compile(r'(?P<ver>\d+(\.\d+)*)')

def extracted_ver(ver):
    occ=pattern.search(ver)
    if occ:
        return occ.group('ver')
    else:
        return np.nan

gPlay['Current Ver_converted']=gPlay['Current Ver'].astype(str).apply(extracted_ver)
gPlay[['Current Ver_converted','Current Ver']].head()
```

Out[18]:

	Current Ver_converted	Current Ver
0	1.0.0	1.0.0
1	2.0.0	2.0.0
2	1.2.4	1.2.4
3	NaN	NaN
4	1.1	1.1

In [19]:

```
gPlay['Android Ver_converted']=gPlay['Android Ver'].astype(str).apply(extracted_ver)
gPlay[['Android Ver','Android Ver_converted']].head()
```

Out[19]:

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

5.Remove the duplicates

In [20]:

```
# remove all equal rows

gPlay.drop_duplicates(keep=False, inplace=True)
```

In [21]:

```
# Check is some Apps belong to different categories.

pd.DataFrame(gPlay.groupby('App')['Category'].count().sort_values(ascending=False)).head()
```

Out[21]:

	Category
App	
ROBLOX	9
8 Ball Pool	7
Helix Jump	6
Bubble Shooter	6
Zombie Catchers	6

In [22]:

```
# Some Apps belong to different categories. I keep the Apps with different categories,
# to keep information that might be important.

gPlay[gPlay['App']=='ROBLOX'].sort_values(['App','Reviews'],ascending=False)
```

Out[22]:

	App	Category	Rating	Reviews	Size	Installs	Type	Price	Content Rating	
2206	ROBLOX	FAMILY	4.5	4450890	67M	100,000,000+	Free	0	Everyone 10+	Adventu & A
2088	ROBLOX	FAMILY	4.5	4450855	67M	100,000,000+	Free	0	Everyone 10+	Adventu & A
1870	ROBLOX	GAME	4.5	4449910	67M	100,000,000+	Free	0	Everyone 10+	Adventu & A
2016	ROBLOX	FAMILY	4.5	4449910	67M	100,000,000+	Free	0	Everyone 10+	Adventu & A
1841	ROBLOX	GAME	4.5	4449882	67M	100,000,000+	Free	0	Everyone 10+	Adventu & A
1748	ROBLOX	GAME	4.5	4448791	67M	100,000,000+	Free	0	Everyone 10+	Adventu & A
1653	ROBLOX	GAME	4.5	4447388	67M	100,000,000+	Free	0	Everyone 10+	Adventu & A
1701	ROBLOX	GAME	4.5	4447346	67M	100,000,000+	Free	0	Everyone 10+	Adventu & A
4527	ROBLOX	FAMILY	4.5	4443407	67M	100,000,000+	Free	0	Everyone 10+	Adventu & A

In [23]:

```
# Convert the 'Review' column from string to num.
gPlay['Reviews']=gPlay['Reviews'].apply(pd.to_numeric)
```

In [24]:

```
# Delete duplicates maintaining the differents categories and the row with the highest numb

gPlay=gPlay.sort_values(['App','Reviews'],ascending=False).drop_duplicates(subset=['App','C
gPlay[gPlay['App']=='ROBLOX'].sort_values(['App','Reviews'],ascending=False)
```

Out[24]:

	App	Category	Rating	Reviews	Size	Installs	Type	Price	Content Rating	
2206	ROBLOX	FAMILY	4.5	4450890	67M	100,000,000+	Free	0	Everyone 10+	Adventu & A
1870	ROBLOX	GAME	4.5	4449910	67M	100,000,000+	Free	0	Everyone 10+	Adventu & A

6. For each category, compute the number of apps

In [25]:

```
pd.DataFrame(gPlay.groupby('Category')['App'].count().sort_values(ascending=False)).head()
```

Out[25]:

	App
Category	
FAMILY	1889
GAME	957
TOOLS	829
BUSINESS	394
PERSONALIZATION	374

7. For each category, compute the average rating

In [26]:

```
pd.DataFrame(gPlay.groupby('Category')['Rating'].mean()).head()
```

Out[26]:

	Rating
Category	
ART_AND_DESIGN	4.357377
AUTO_AND_VEHICLES	4.190411
BEAUTY	4.278571
BOOKS_AND_REFERENCE	4.346429
BUSINESS	4.078481

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

In [27]:

```
# use 'split' to divide the tupla inside the column 'Genres' in the different Genres.

split_genres = gPlay['Genres'].str.split(pat = ';', expand=True)

# Use concat in order to concatenate the 2 columns previusly founded.
df_conc = pd.concat([split_genres[0],split_genres[1]]).drop_duplicates().sort_values().drop
df_genres = pd.DataFrame(df_conc, columns = ['Genres'])
df_genres.reset_index().drop(columns=['index']).head(7)
```

Out[27]:

	Genres
0	Action
1	Action & Adventure
2	Adventure
3	Arcade
4	Art & Design
5	Auto & Vehicles
6	Beauty

In [28]:

```
df_genres.to_csv(r"C:/Users/davla/Desktop/Computer_Science/df_genres.csv",sep=';')
```

In [29]:

```
# Create 2 different dataframes in which I associate every apps with his genre/s.
# Use axis=1 in the function concat in order to obtain a concatenation based on columns.
# Finally, use the function append in order to merge the dataframes in one single dataframe

app = pd.DataFrame(gPlay['App'])
col1 = pd.concat([app,split_genres[0]],axis=1)
col1.columns = ['App','Genres']
col2 = pd.concat([app,split_genres[1]],axis=1)
col2.columns = ['App','Genres']
app_genres = col1.append(col2).dropna()

#Example: "Pixel Draw" belongs to 2 Genres.

app_genres[app_genres['App'] == 'Pixel Draw - Number Art Coloring Book']
```

Out[29]:

	App	Genres
4	Pixel Draw - Number Art Coloring Book	Art & Design
4	Pixel Draw - Number Art Coloring Book	Creativity

In [30]:

```
app_genres.to_csv(r"C:/Users/davla/Desktop/Computer_Science/app_genres.csv",sep=';')
```

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)

In [31]:

```
gPlay['G_0']=split_genres[0]
gPlay['G_1']=split_genres[1]
```

In [32]:

```
# Use get_dummies in order to crate 2 dataframes. In input: 'G_0' and 'G_1', the columns of
# We obtain a boolean values throw dtype=bool (True e False as output). Then use concat in
# a single dataframe 'df_dum'. Finally, merge it with the original dataset.

df_dum_0=pd.get_dummies(gPlay['G_0'],dtype=bool)
df_dum_1=pd.get_dummies(gPlay['G_1'],dtype=bool)
df_dum=pd.concat([df_dum_0,df_dum_1],axis=1)
gPlay=pd.concat([gPlay,df_dum],axis=1)
gPlay.head(4)
```

Out[32]:

	App	Category	Rating	Reviews	Size	Installs	Type	Price	Conte Rati
882	🔥 Football Wallpapers 4K | Full HD Backgrounds 😍	ENTERTAINMENT	4.7	11661	4.0M	1,000,000+	Free	0	Everyo
7559	📏 Smart Ruler ↔ cm/inch measuring for homework!	TOOLS	4.0	19	3.2M	10,000+	Free	0	Everyo
2575	💘 WhatsLov: Smileys of love, stickers and GIF	SOCIAL	4.6	22098	18M	1,000,000+	Free	0	Everyo
4362	💎 I'm rich	LIFESTYLE	3.8	718	26M	10,000+	Paid	$399.99	Everyo

4 rows × 73 columns

In [33]:

```
# Example of 'ROBLOX' application

ex=gPlay[['App','Genres', 'Action & Adventure', 'Adventure']]
ex[ex['App']=='ROBLOX']
```

Out[33]:

	App	Genres	Action & Adventure	Adventure
2206	ROBLOX	Adventure;Action & Adventure	True	True
1870	ROBLOX	Adventure;Action & Adventure	True	True

In [34]:

```
# example

gPlay[['App','Genres','Entertainment','Tools','Social','Lifestyle']].head(4)
```

Out[34]:

	App	Genres	Entertainment	Tools	Social	Lifestyle
882	🔥 Football Wallpapers 4K | Full HD Backgrounds 😍	Entertainment	True	False	False	False
7559	📏 Smart Ruler ↔ cm/inch measuring for homework!	Tools	False	True	False	False
2575	💘 WhatsLov: Smileys of love, stickers and GIF	Social	False	False	True	False
4362	💎 I'm rich	Lifestyle	False	False	False	True

10. For each genre, compute the average rating. What is the genre with highest average?

In [35]:

```
# Same procedure of the exercise 8.

rating = pd.DataFrame(gPlay['Rating'])
col1 = pd.concat([rating,split_genres[0]],axis=1)
col1.columns = ['Rating','Genres']
col2 = pd.concat([rating,split_genres[1]],axis=1)
col2.columns = ['Rating','Genres']
rating_genres = col1.append(col2).dropna()

# Sort the values and select the first value.

pd.DataFrame(rating_genres.groupby('Genres')['Rating'].mean().sort_values(ascending = False
```

Out[35]:

	Rating
Genres	
Events	4.435556

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

In [36]:

```
gPlay['Price'].unique()
```

Out[36]:

```
array(['0', '$399.99', '$1.49', '$2.99', '$1.99', '$1.20', '$0.99',
       '$4.29', '$4.99', '$4.49', '$3.99', '$10.00', '$154.99', '$1.96',
       '$5.49', '$19.40', '$5.99', '$12.99', '$2.50', '$19.99', '$2.56',
       '$9.99', '$1.04', '$2.90', '$1.00', '$2.49', '$8.99', '$16.99',
       '$1.97', '$3.49', '$17.99', '$6.99', '$6.49', '$2.95', '$4.59',
       '$4.85', '$7.49', '$10.99', '$4.84', '$1.76', '$7.99', '$4.60',
       '$3.02', '$14.99', '$39.99', '$1.70', '$15.99', '$1.59', '$8.49',
       '$1.61', '$89.99', '$74.99', '$15.46', '$1.26', '$400.00',
       '$299.99', '$379.99', '$18.99', '$37.99', '$389.99', '$24.99',
       '$29.99', '$25.99', '$3.88', '$13.99', '$2.00', '$30.99',
       '$394.99', '$4.77', '$3.61', '$200.00', '$28.99', '$46.99',
       '$3.28', '$3.95', '$14.00', '$2.59', '$11.99', '$4.80', '$109.99',
       '$1.75', '$3.90', '$1.50', '$1.29', '$2.60', '$19.90', '$5.00',
       '$3.04', '$3.08'], dtype=object)
```

In [37]:

```
gPlay['New_Price']=gPlay['Price'].str.replace('$','',regex=True).apply(pd.to_numeric)
gPlay['Approx_Income'] = gPlay['New_Installs']*gPlay['New_Price']
gPlay[['App','New_Installs','New_Price','Approx_Income']].sort_values('Approx_Income',ascer
```

Out[37]:

	App	New_Installs	New_Price	Approx_Income
2241	Minecraft	10000000	6.99	69900000.0
5351	I am rich	100000	399.99	39999000.0
5356	I Am Rich Premium	50000	399.99	19999500.0
4034	Hitman Sniper	10000000	0.99	9900000.0
7417	Grand Theft Auto: San Andreas	1000000	6.99	6990000.0

12. For each app, compute its minimum and maximum Sentiment_polarity

In [38]:

```
gPlay_review.head()
```

Out[38]:

	App	Translated_Review	Sentiment	Sentiment_Polarity	Sentiment_Subjectivity
0	10 Best Foods for You	I like eat delicious food. That's I'm cooking ...	Positive	1.00	0.533333
1	10 Best Foods for You	This help eating healthy exercise regular basis	Positive	0.25	0.288462
2	10 Best Foods for You	NaN	NaN	NaN	NaN
3	10 Best Foods for You	Works great especially going grocery store	Positive	0.40	0.875000
4	10 Best Foods for You	Best idea us	Positive	1.00	0.300000

In [41]:

```
# Group by the apps, then calculate max and min of 'Sentiment Polarity' and merge everythin

max_sentiment_polarity = gPlay_review.groupby('App')['Sentiment_Polarity'].max()
min_sentiment_polarity = gPlay_review.groupby('App')['Sentiment_Polarity'].min()

max_sent_pol=pd.DataFrame(max_sentiment_polarity)
min_sent_pol=pd.DataFrame(min_sentiment_polarity)

pd.DataFrame(min_sent_pol.merge(max_sent_pol,left_on='App', right_on='App',suffixes=('_min'
```

Out[41]:

	Sentiment_Polarity_min	Sentiment_Polarity_max
App		
10 Best Foods for You	-0.800000	1.000000
104 找工作 - 找工作 找打工 找兼職 履歷健檢 履歷診療室	-0.112500	0.910000
11st	-1.000000	1.000000
1800 Contacts - Lens Store	-0.300000	0.838542
1LINE – One Line with One Touch	-0.825000	1.000000
2018Emoji Keyboard 😂 Emoticons Lite -sticker&gif	-0.800000	1.000000
21-Day Meditation Experience	-0.265625	0.587500
2Date Dating App, Love and matching	-0.645833	1.000000
2GIS: directory & navigator	-0.375000	1.000000
2RedBeans	-0.800000	1.000000