Project On New Profitable App Profiles for the App Store and Google Play Markets, with Explanation of Code

This project aims to suggest new profitable apps to developers on the Android and iOS mobile app markets. To determine what sort of apps may be profitable, we investigate apps that have been successful in the past. Success is operationalized as the number of users of a given app because the revenue at our company stems from in-app ads. Apps are free to download and to install.

Two free data sets with n = 10000 (Android) and n = 7000 (iOS) respectively are used as samples for the ca. 4 million apps that were available in 2018: 2 million iOS and 2.1 million Android apps.

The links to the data sets are:

https://dq-content.s3.amazonaws.com/350/googleplaystore.csv (https://dq-content.s3.amazonaws.com/350/googleplaystore.csv)

https://dq-content.s3.amazonaws.com/350/AppleStore.csv (https://dq-content.s3.amazonaws.com/350/AppleStore.csv)

Let us open the two data sets in Python for further exploration and analysis.

1. Data Exploration

In [1]:
```
from csv import reader

### The Google Play data set (Android apps) ###
opened_file = open('googleplaystore.csv') # Open the file using the open
() command. Save the output to a variable named 'opened_file'.
read_file = reader(opened_file) # Read in the opened file using the read
er() command. Save the output to a variable named 'read_file'.
android = list(read_file) # Transform the read-in file to a list of list
s using the list() command. Save the list of lists to a variable named
 'android'.
android_header = android[0] # The first row is the header
android = android[1:] # Data begins in row 2, after the header

### The App Store data set (iOS apps) ###
opened_file = open('AppleStore.csv')
read_file = reader(opened_file)
ios = list(read_file)
ios_header = ios[0]
ios = ios[1:]
```

The following creates a function that allows one to explore the data sets.

In [2]:
```
def explore_data(dataset, start, end, rows_and_columns=False): # We call the function 'explore_data'
    dataset_slice = dataset[start:end]
    for row in dataset_slice: # Looping through data set
        print(row)
        print('\n') # Adds a new (empty) line after each row

    if rows_and_columns: # If rows_and_columns is True
        print('Number of rows:', len(dataset)) # Print this for any data set
        print('Number of columns:', len(dataset[0])) # Print this for any data set

print(android_header)
print('\n') # Insert an empty line
explore_data(android, 0, 3, True)
```

```
['App', 'Category', 'Rating', 'Reviews', 'Size', 'Installs', 'Type', 'Price', 'Content Rating', 'Genres', 'Last Updated', 'Current Ver', 'Android Ver']


['Photo Editor & Candy Camera & Grid & ScrapBook', 'ART_AND_DESIGN', '4.1', '159', '19M', '10,000+', 'Free', '0', 'Everyone', 'Art & Design', 'January 7, 2018', '1.0.0', '4.0.3 and up']


['Coloring book moana', 'ART_AND_DESIGN', '3.9', '967', '14M', '500,000+', 'Free', '0', 'Everyone', 'Art & Design;Pretend Play', 'January 15, 2018', '2.0.0', '4.0.3 and up']


['U Launcher Lite – FREE Live Cool Themes, Hide Apps', 'ART_AND_DESIGN', '4.7', '87510', '8.7M', '5,000,000+', 'Free', '0', 'Everyone', 'Art & Design', 'August 1, 2018', '1.2.4', '4.0.3 and up']


Number of rows: 10841
Number of columns: 13
```

We printed the first few rows of the Google Play data set, and found the number of rows and columns of the data set. The data sets should not have a header row as the function explore_data() assumes that there is no header row. This data set contains 10841 rows and 13 columns. Columns useful to investigate are 'App', 'Category', 'Reviews', 'Installs', 'Type', 'Price', and 'Genres'. Next, let us use the new function in [2] on the App store data set.

In [3]:
```
print(ios_header)
print('\n')
explore_data(ios, 0, 3, True)
```

```
['id', 'track_name', 'size_bytes', 'currency', 'price', 'rating_count_t
ot', 'rating_count_ver', 'user_rating', 'user_rating_ver', 'ver', 'cont
_rating', 'prime_genre', 'sup_devices.num', 'ipadSc_urls.num', 'lang.nu
m', 'vpp_lic']


['284882215', 'Facebook', '389879808', 'USD', '0.0', '2974676', '212',
'3.5', '3.5', '95.0', '4+', 'Social Networking', '37', '1', '29', '1']


['389801252', 'Instagram', '113954816', 'USD', '0.0', '2161558', '128
9', '4.5', '4.0', '10.23', '12+', 'Photo & Video', '37', '0', '29',
'1']


['529479190', 'Clash of Clans', '116476928', 'USD', '0.0', '2130805',
'579', '4.5', '4.5', '9.24.12', '9+', 'Games', '38', '5', '18', '1']


Number of rows: 7197
Number of columns: 16
```

This data set has 7197 rows and 16 columns. The columns that might help with this analysis are 'track_name', 'currency', 'price', 'rating_count_tot', 'rating_count_ver', and 'prime_genre'. The documentation of these can be found in https://www.kaggle.com/ramamet4/app-store-apple-data-set-10k-apps/home (https://www.kaggle.com/ramamet4/app-store-apple-data-set-10k-apps/home).

2. Data Cleaning

2.1 Data Cleaning: Removing Incorrect Data

The discussion section for the Google Play data set at https://www.kaggle.com/lava18/google-play-store-apps/discussion/66015 (https://www.kaggle.com/lava18/google-play-store-apps/discussion/66015) reveals that line 10472 contains an error. We print that line in the following as well as the header and a correct row, in order to be able to compare them to find the error.

In [4]:
```
print(android[10472])  # incorrect row
print('\n')
print(android_header)  # header
print('\n')
print(android[0])      # correct row
```

```
['Life Made WI-Fi Touchscreen Photo Frame', '1.9', '19', '3.0M', '1,000
+', 'Free', '0', 'Everyone', '', 'February 11, 2018', '1.0.19', '4.0 an
d up']


['App', 'Category', 'Rating', 'Reviews', 'Size', 'Installs', 'Type', 'P
rice', 'Content Rating', 'Genres', 'Last Updated', 'Current Ver', 'Andr
oid Ver']


['Photo Editor & Candy Camera & Grid & ScrapBook', 'ART_AND_DESIGN',
'4.1', '159', '19M', '10,000+', 'Free', '0', 'Everyone', 'Art & Desig
n', 'January 7, 2018', '1.0.0', '4.0.3 and up']
```

As noted in the link cited in the text immediately preceding [4], the 'Category' column is missing a value in the row with an error, which is the app 'Life Made WI-Fi Touchscreen Photo Frame'. This leads to the rating to be 19, whereas the maximum rating for the Android apps is actually 5. This row could be deleted, which is what is done in the following.

In [5]:
```
print(len(android)) # Number of elements, here, rows, in the data set
del android[10472]  # Don't run this more than once
print(len(android))
```

```
10841
10840
```

We went from 10841 to 10840, which indicates that we succeeded in eliminating one row.

2.2 Data Cleaning Continued: Removing Duplicates

The Google Play data set contains some duplicate rows:

In [6]:
```
for app in android:
    name = app[0] # 'Name' is the first column
    if name == 'Instagram':
        print(app)
```

```
['Instagram', 'SOCIAL', '4.5', '66577313', 'Varies with device', '1,00
0,000,000+', 'Free', '0', 'Teen', 'Social', 'July 31, 2018', 'Varies wi
th device', 'Varies with device']
['Instagram', 'SOCIAL', '4.5', '66577446', 'Varies with device', '1,00
0,000,000+', 'Free', '0', 'Teen', 'Social', 'July 31, 2018', 'Varies wi
th device', 'Varies with device']
['Instagram', 'SOCIAL', '4.5', '66577313', 'Varies with device', '1,00
0,000,000+', 'Free', '0', 'Teen', 'Social', 'July 31, 2018', 'Varies wi
th device', 'Varies with device']
['Instagram', 'SOCIAL', '4.5', '66509917', 'Varies with device', '1,00
0,000,000+', 'Free', '0', 'Teen', 'Social', 'July 31, 2018', 'Varies wi
th device', 'Varies with device']
```

The example above is for the app 'Instagram'. What about the other apps in this data set? Let us explore this question in the following.

In [7]:
```
duplicate_apps = [] # Creating new empty list
unique_apps = [] # Creating new empty list

for app in android: # Looping through the 'Android' data set
    name = app[0]
    if name in unique_apps: # For each repetition, if column 1 is in 'un
ique_apps'
        duplicate_apps.append(name) # Add name to list 'duplicate_apps'
    else: # If not yet in 'unique_apps'
        unique_apps.append(name) # Add name to list 'unique_apps'

print('Number of duplicate apps:', len(duplicate_apps))
print('\n')
print('Examples of duplicate apps:', duplicate_apps[:15]) # Print 15 ent
ries
```

```
Number of duplicate apps: 1181


Examples of duplicate apps: ['Quick PDF Scanner + OCR FREE', 'Box', 'Go
ogle My Business', 'ZOOM Cloud Meetings', 'join.me - Simple Meetings',
'Box', 'Zenefits', 'Google Ads', 'Google My Business', 'Slack', 'FreshB
ooks Classic', 'Insightly CRM', 'QuickBooks Accounting: Invoicing & Exp
enses', 'HipChat - Chat Built for Teams', 'Xero Accounting Software']
```

This data set thus contains 1181 duplicate data sets. To remove them and to only keep one row per app, one way is to only keep that row per app that has the highest number of ratings. Rating numbers vary across the duplicates.

A dictionary is created below in which each app corresponds to a separate key and its value is the highest number of reviews of that app. This creates a data set with only one app entry each, the highest one.

In [8]:
```
reviews_max = {} # Create new dictionary, empty for now

for app in android: # If app exists as dictionary key in android list
    name = app[0] # Name is column 1
    n_reviews = float(app[3]) # Number of reviews appears in column 4 an
d we convert it to floats

    if name in reviews_max and reviews_max[name] < n_reviews: # Getting
 rid of all below the max
        reviews_max[name] = n_reviews

    elif name not in reviews_max:
        reviews_max[name] = n_reviews
```

The length of this new dictionary should be the length of the entire data set minus 1,181, which corresponds to the number of entries that have multiple ratings (see above). Printing the result of this difference and comparing it to the number of rows in the new dictionary supports this assumption.

In [9]:
```
print('Expected length:', len(android) - 1181)
print('Actual length:', len(reviews_max))
```

```
Expected length: 9659
Actual length: 9659
```

Next, it would be good to remove the duplicates using the new dictionary 'max_reviews'.

In [10]:
```
android_clean = [] # Creating empty list
already_added = [] # Creating empty list

for app in android: # Loop started
    name = app[0]
    n_reviews = float(app[3])

    if (reviews_max[name] == n_reviews) and (name not in
                                             already_added): # If number
 of reviews = as in 'reviews_max' dictionary and if name of app not yet
 in 'already_added' list
        android_clean.append(app) # Add row to 'android_clean' list
        already_added.append(name) # Add name to 'already_added' list
```

In this step, we added the current row (app) to a list called 'android_clean' and the app name (name) to another list ('already_added') if the number of reviews of the current app is equal to the number of reviews as specified in the new dictionary ('reviews_max'), and if the name of the app is not already in the 'already_added' list. The latter condition refers to cases in which there are multiple identical max review numbers. The android_clean list should have 9659 rows. Let us test this.

In [11]:
```
explore_data(android_clean, 0, 3, True) # Using our function from [2] to explore the new data set/list
```

```
['Photo Editor & Candy Camera & Grid & ScrapBook', 'ART_AND_DESIGN',
'4.1', '159', '19M', '10,000+', 'Free', '0', 'Everyone', 'Art & Desig
n', 'January 7, 2018', '1.0.0', '4.0.3 and up']


['U Launcher Lite – FREE Live Cool Themes, Hide Apps', 'ART_AND_DESIG
N', '4.7', '87510', '8.7M', '5,000,000+', 'Free', '0', 'Everyone', 'Art
& Design', 'August 1, 2018', '1.2.4', '4.0.3 and up']


['Sketch - Draw & Paint', 'ART_AND_DESIGN', '4.5', '215644', '25M', '5
0,000,000+', 'Free', '0', 'Teen', 'Art & Design', 'June 8, 2018', 'Vari
es with device', '4.2 and up']


Number of rows: 9659
Number of columns: 13
```

The number looks correct.

2.3 Data Cleaning Continued: Removing Non-English Apps

In the following, let us clean the data further by removing apps that have a non-English name, for the sake of an easy preliminary analysis and because the present project tries to work with apps designed for an English audience. We can accomplish this for example by removing apps containing non-English symbols, that is, symbols that are not encoded using the ASCII standard (numbers 0 - 127 that encode a symbol each).

Let us build a function that tells us whether an app name contains ASCII characters. First, let us look at examples with non-English names.

In [12]:
```
print(ios[813][1])
print(ios[6731][1])

print(android_clean[4412][0])
print(android_clean[7940][0])
```

```
爱奇艺PPS -《欢乐颂2》电视剧热播
【脱出ゲーム】絶対に最後までプレイしないで ～謎解き＆ブロックパズル～
中国語 AQリスニング
لعبة تقدر تربح DZ
```

Now, to a function that isolates English names.

In [13]:
```
def is_english(string): # Defining new function

    for character in string:
        if ord(character) > 127: # Built-in function: find encoding numb
ers
            return False

    return True # Else return 'true'

print(is_english('Instagram'))
print(is_english('爱奇艺PPS -《欢乐颂2》电视剧热播'))
```

```
True
False
```

The function 'is_english' shows correctly that the non-English app tested in the second function run is not ASCII encoded. However, note that the current function will also dub as false those apps that contain symbols other than non-ASCII symbols, for example, smileys (example [14] below). This can cause data loss. The function thus requires some tweaking. One way to keep app names that are English but contain say a smiley symbol is to specify we only get rid of an app if its name has > 3 non-ASCII characters. The code in [15] below does exactly that.

In [14]:
```
print(is_english('Docs To Go™ Free Office Suite'))
print(is_english('Instachat 😜'))

print(ord('™'))
print(ord('😜'))
```

```
False
False
8482
128540
```

In [15]:
```
def is_english(string):
    non_ascii = 0

    for character in string:
        if ord(character) > 127:
            non_ascii += 1

    if non_ascii > 3:
        return False
    else:
        return True

print(is_english('Docs To Go™ Free Office Suite'))
print(is_english('Instachat 😜'))
```

```
True
True
```

We leave optimization of this function for a later exercise. Let us in the following test the function 'is_english(string)' on both data sets under investigation in the present project.

In [16]:
```
android_english = [] # Initiating an empty list
ios_english = [] # Initiating another empty list

for app in android_clean:
    name = app[0] # First column
    if is_english(name):
        android_english.append(app) # Append row if name is English

for app in ios:
    name = app[1] # Second column
    if is_english(name):
        ios_english.append(app)

explore_data(android_english, 0, 3, True)
print('\n')
explore_data(ios_english, 0, 3, True)
```

```
['Photo Editor & Candy Camera & Grid & ScrapBook', 'ART_AND_DESIGN',
'4.1', '159', '19M', '10,000+', 'Free', '0', 'Everyone', 'Art & Desig
n', 'January 7, 2018', '1.0.0', '4.0.3 and up']


['U Launcher Lite – FREE Live Cool Themes, Hide Apps', 'ART_AND_DESIG
N', '4.7', '87510', '8.7M', '5,000,000+', 'Free', '0', 'Everyone', 'Art
& Design', 'August 1, 2018', '1.2.4', '4.0.3 and up']


['Sketch - Draw & Paint', 'ART_AND_DESIGN', '4.5', '215644', '25M', '5
0,000,000+', 'Free', '0', 'Teen', 'Art & Design', 'June 8, 2018', 'Vari
es with device', '4.2 and up']


Number of rows: 9614
Number of columns: 13


['284882215', 'Facebook', '389879808', 'USD', '0.0', '2974676', '212',
'3.5', '3.5', '95.0', '4+', 'Social Networking', '37', '1', '29', '1']


['389801252', 'Instagram', '113954816', 'USD', '0.0', '2161558', '128
9', '4.5', '4.0', '10.23', '12+', 'Photo & Video', '37', '0', '29',
'1']


['529479190', 'Clash of Clans', '116476928', 'USD', '0.0', '2130805',
'579', '4.5', '4.5', '9.24.12', '9+', 'Games', '38', '5', '18', '1']


Number of rows: 6183
Number of columns: 16
```