Android Google Playstore Analysis

May 12, 2022

0.1 1.Loading Dataset and Libraries

[2]:

```
# Read in dataset
import pandas as pd
apps_with_duplicates = pd.read_csv('datasets/apps.csv')

# Drop duplicates from apps_with_duplicates
apps = apps = apps_with_duplicates.drop_duplicates()

# Print the total number of apps
print('Total number of apps in the dataset = ', len(apps))

# Have a look at a random sample of 5 rows
n = 5
apps.sample(n)
```

```
Total number of apps in the dataset =  9659
```

[2]:

```
      Unnamed: 0                         App             Category  Rating  \
5245        6244              B y H Niños ES  BOOKS_AND_REFERENCE     4.6
8659        9802  File Ex - ES File Explorer                TOOLS     4.2
1544        1952                      TEKKEN                 GAME     4.2
6222        7267      Special Forces Group 2                 GAME     4.6
4126        5077                     AppLock                TOOLS     4.4

      Reviews  Size      Installs  Type Price Content Rating  \
5245       53  16.0        5,000+  Free     0       Everyone
8659       24   5.0        1,000+  Free     0       Everyone
1544   147791  38.0    5,000,000+  Free     0           Teen
6222  1432809  29.0   10,000,000+  Free     0     Mature 17+
4126  4931562   NaN  100,000,000+  Free     0       Everyone

                 Genres        Last Updated         Current Ver  \
5245  Books & Reference  September 22, 2015               1.0.2
8659              Tools   December 27, 2017               1.1.6
1544             Action       July 26, 2018                 1.3
6222             Action       July 29, 2018                 3.3
4126              Tools       June 11, 2018  Varies with device
```

```
             Android Ver
5245          2.3 and up
8659          4.2 and up
1544          5.0 and up
6222          4.0 and up
4126  Varies with device
```

0.2 2. Data cleaning

By looking at a random sample of the dataset rows (from the above task), we observe that some entries in the columns like Installs and Price have a few special characters (+ , $) due to the way the numbers have been represented.

[3]:

```python
chars_to_remove = [',','$','+']
cols_to_clean = ['Installs','Price']

# Loop for each column in cols_to_clean
for col in cols_to_clean:
    # Loop for each char in chars_to_remove
    for char in chars_to_remove:
        # Replace the character with an empty string
        apps[col] = apps[col].apply(lambda x: x.replace(char, ''))


print(apps.info())
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 9659 entries, 0 to 9658
Data columns (total 14 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   Unnamed: 0      9659 non-null   int64
 1   App             9659 non-null   object
 2   Category        9659 non-null   object
 3   Rating          8196 non-null   float64
 4   Reviews         9659 non-null   int64
 5   Size            8432 non-null   float64
 6   Installs        9659 non-null   object
 7   Type            9659 non-null   object
 8   Price           9659 non-null   object
 9   Content Rating  9659 non-null   object
 10  Genres          9659 non-null   object
 11  Last Updated    9659 non-null   object
 12  Current Ver     9651 non-null   object
 13  Android Ver     9657 non-null   object
dtypes: float64(2), int64(2), object(10)
memory usage: 1.1+ MB
```

```
None
```

0.3 3. Correcting data types

From the previous task we noticed that Installs and Price were categorized as object data type (and not int or float) as we would like. We need to work on Installs and Price to make them numeric.

[4]:

```python
import numpy as np

# Convert Installs to float data type
apps['Installs'] = apps['Installs'].astype(float)

# Convert Price to float data type
apps['Price'] = apps['Price'].astype(float)

# Checking dtypes of the apps dataframe
print(apps.info())
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 9659 entries, 0 to 9658
Data columns (total 14 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   Unnamed: 0      9659 non-null   int64
 1   App             9659 non-null   object
 2   Category        9659 non-null   object
 3   Rating          8196 non-null   float64
 4   Reviews         9659 non-null   int64
 5   Size            8432 non-null   float64
 6   Installs        9659 non-null   float64
 7   Type            9659 non-null   object
 8   Price           9659 non-null   float64
 9   Content Rating  9659 non-null   object
 10  Genres          9659 non-null   object
 11  Last Updated    9659 non-null   object
 12  Current Ver     9651 non-null   object
 13  Android Ver     9657 non-null   object
dtypes: float64(4), int64(2), object(8)
memory usage: 1.1+ MB
None
```

0.4 4. Exploring app categories

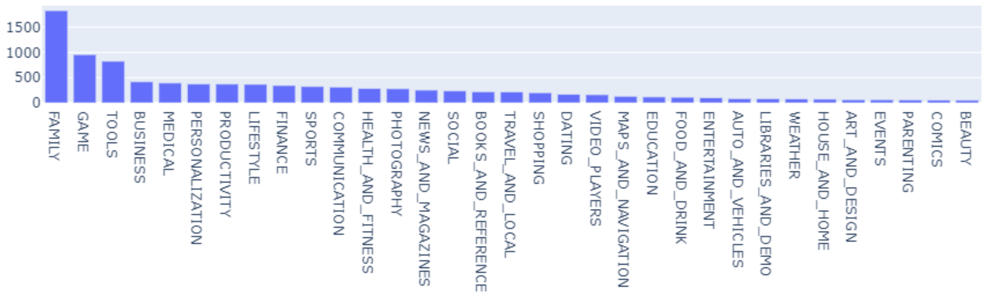
Which category has the highest share of (active) apps in the market?

Is any specific category dominating the market?

Which categories have the fewest number of apps?

```
[5]: import plotly
plotly.offline.init_notebook_mode(connected=True)
import plotly.graph_objs as go

# Print the total number of unique categories
num_categories = len(apps['Category'].unique())
print('Number of categories = ', num_categories)

# Count the number of apps in each 'Category'.
num_apps_in_category =  apps['Category'].value_counts()

# Sort num_apps_in_category in descending order based on the count of apps in␣
 ↪each category
sorted_num_apps_in_category = num_apps_in_category.sort_values(ascending =␣
 ↪False)

data = [go.Bar(
        x = num_apps_in_category.index, # index = category name
        y = num_apps_in_category.values, # value = count
)]

plotly.offline.iplot(data)
```

```
Number of categories =  33
```



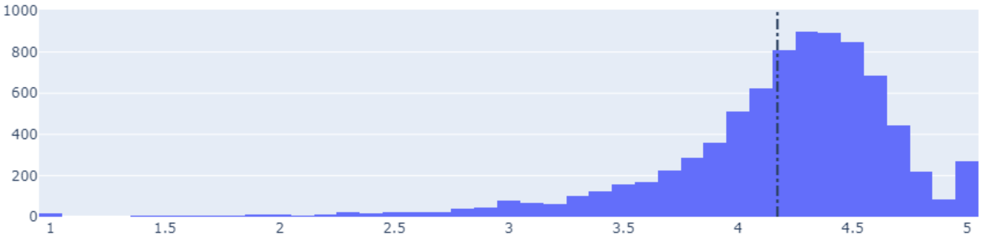
0.5 5. Distribution of app ratings

The average volume of ratings across all app categories is 4.17. The histogram plot is skewed to the left indicating that the majority of the apps are highly rated with only a few exceptions in the low-rated apps.

[6]:

```python
# Average rating of apps
avg_app_rating = apps['Rating'].mean()
print('Average app rating = ', avg_app_rating)

# Distribution of apps according to their ratings
data = [go.Histogram(
        x = apps['Rating']
)]

# Vertical dashed line to indicate the average app rating
layout = {'shapes': [{
              'type' :'line',
              'x0': avg_app_rating,
              'y0': 0,
              'x1': avg_app_rating,
              'y1': 1000,
              'line': { 'dash': 'dashdot'}
          }]
          }

plotly.offline.iplot({'data': data, 'layout': layout})
```

```
Average app rating =  4.173243045387994
```



0.6 6. Size and price of an app

How can we effectively come up with strategies to size and price our app?

Does the size of an app affect its rating?

Do users really care about system-heavy apps or do they prefer light-weighted apps?

Does the price of an app affect its rating?

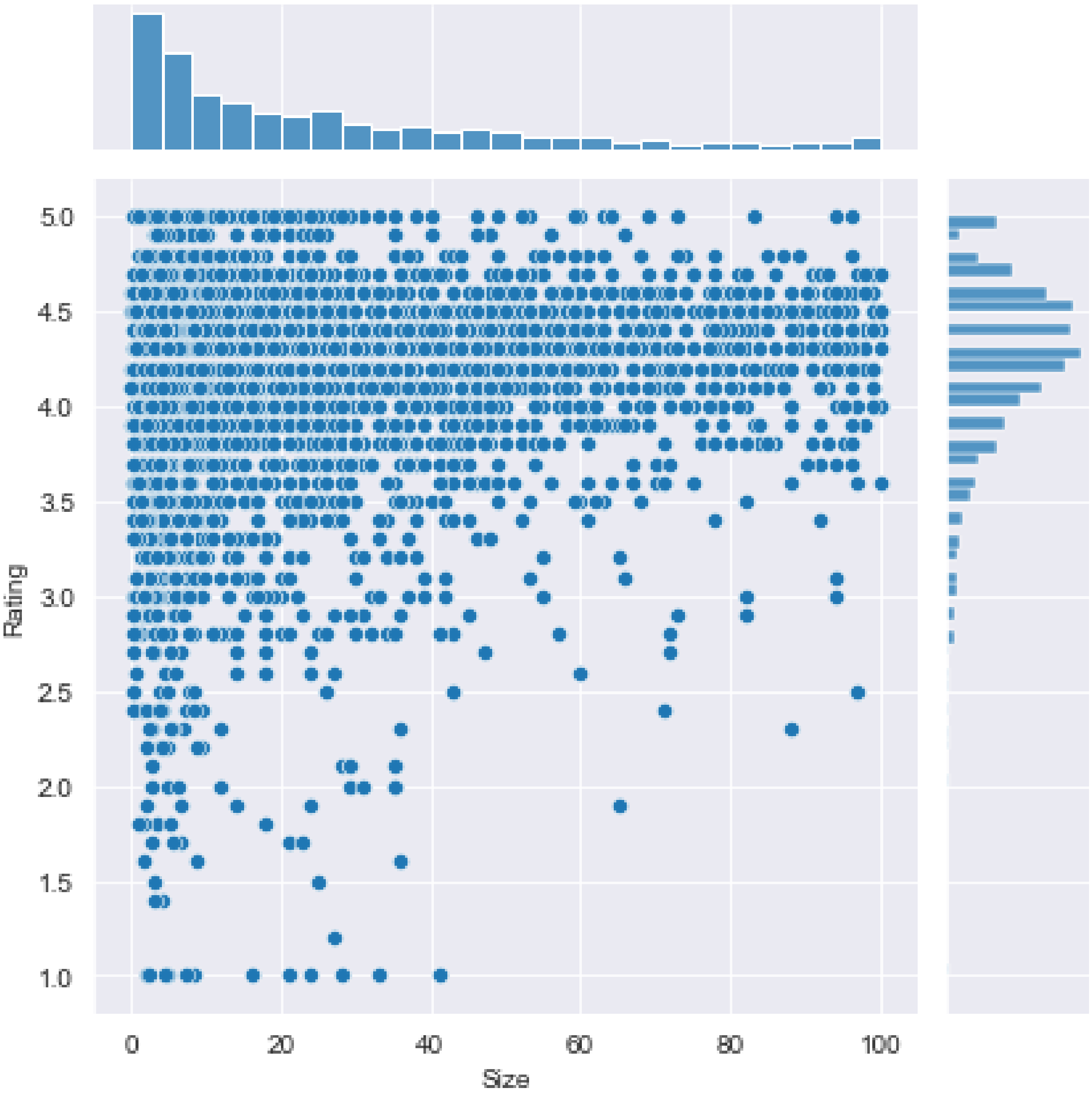
Do users always prefer free apps over paid apps?

We find that the majority of top rated apps (rating over 4) range from 2 MB to 20 MB. We also find that the vast majority of apps price themselves under $10.

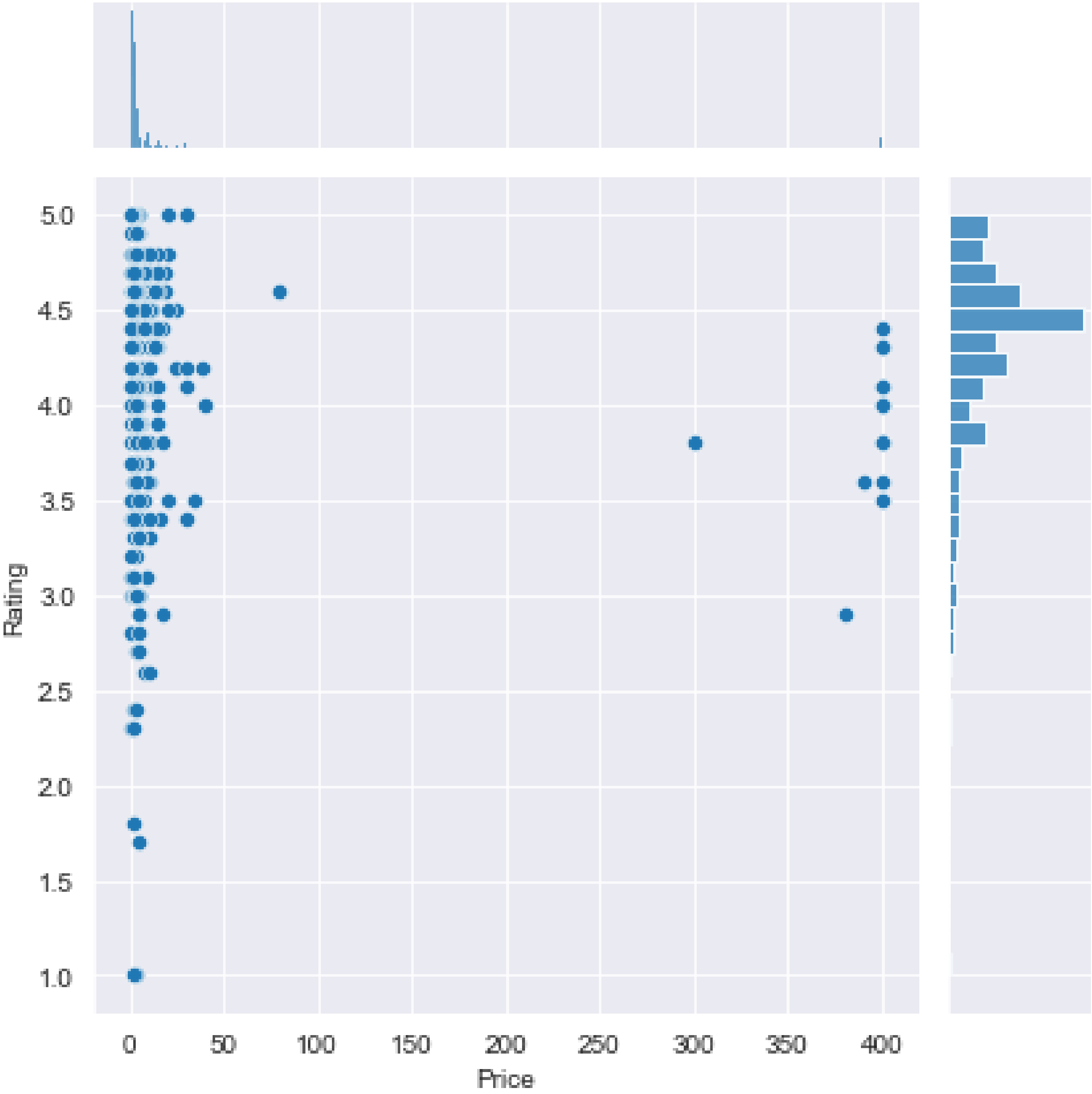
[7]:

```
%matplotlib inline
import seaborn as sns
sns.set_style("darkgrid")
import warnings
warnings.filterwarnings("ignore")

# Select rows where both 'Rating' and 'Size' values are present (ie. the two␣
 ↪values are not null)
apps_with_size_and_rating_present = apps[(~apps['Rating'].isnull()) &␣
 ↪(~apps['Size'].isnull())]

dfa=apps_with_size_and_rating_present['Category'].value_counts().to_frame(name␣
 ↪= 'a')
print(dfa[dfa['a']>=250].reset_index())

# Subset for categories with at least 250 apps
large_categories = apps_with_size_and_rating_present.groupby('Category').
 ↪filter(lambda x: len(x) >= 250)

# Plot size vs. rating
plt1 = sns.jointplot(x = large_categories['Size'], y =␣
 ↪large_categories['Rating'])

# Select apps whose 'Type' is 'Paid'
paid_apps = ␣
 ↪apps_with_size_and_rating_present[apps_with_size_and_rating_present['Type']␣
 ↪== 'Paid']

# Plot price vs. rating
plt2 = sns.jointplot(x = paid_apps['Price'], y = paid_apps['Rating'])
```

```
            index     a
0          FAMILY  1512
1            GAME   832
2           TOOLS   626
3 PERSONALIZATION   276
4       LIFESTYLE   269
5         MEDICAL   266
6         FINANCE   258
```



5.0
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0
Rating
0
20
40
60
80
100
Size



0.7 7. Relation between app category and app price

Different categories demand different price ranges. Some apps that are simple and used daily, like the calculator app, should probably be kept free. However, it would make sense to charge for a highly-specialized medical app that diagnoses diabetic patients. Below, we see that Medical and Family apps are the most expensive. Some medical apps extend even up to $80! All game apps are reasonably priced below $20.

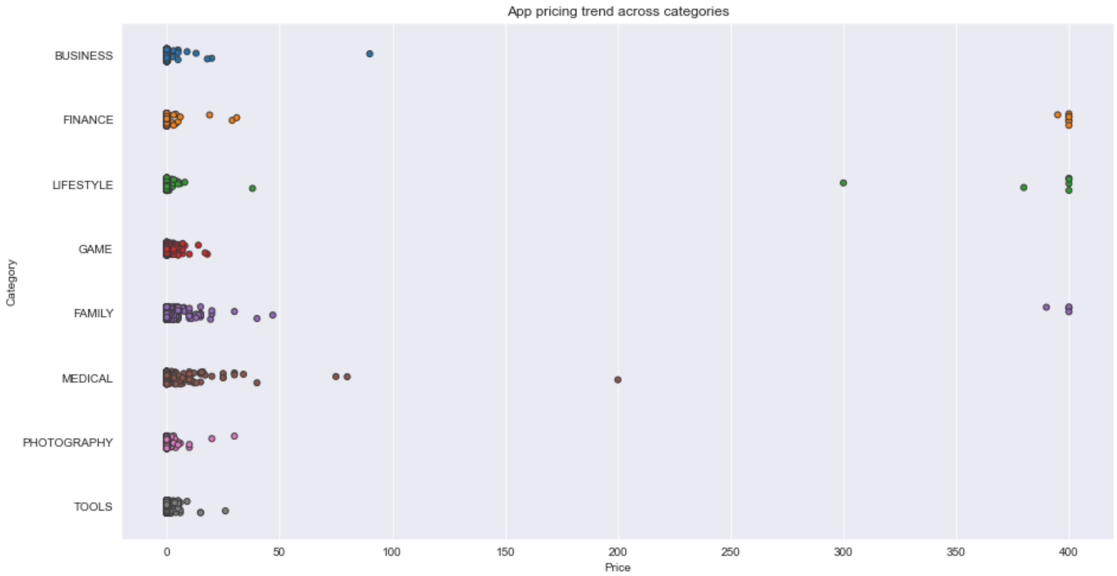
[8]:

```
import matplotlib.pyplot as plt
fig, ax = plt.subplots()
fig.set_size_inches(15, 8)

# Select a few popular app categories
```

```
popular_app_cats = apps[apps.Category.isin(['GAME', 'FAMILY', 'PHOTOGRAPHY',
                                            'MEDICAL', 'TOOLS', 'FINANCE',
                                            'LIFESTYLE','BUSINESS'])]

# Examine the price trend by plotting Price vs Category
ax = sns.stripplot(x = popular_app_cats['Price'], y =␣
 ↪popular_app_cats['Category'], jitter=True, linewidth=1)
ax.set_title('App pricing trend across categories')

# Apps whose Price is greater than 200
apps_above_200 =  popular_app_cats[['Category', 'App',␣
 ↪'Price']][popular_app_cats['Price'] > 200]
apps_above_200[['Category', 'App', 'Price']]
```

[8]:

```
       Category                                App   Price
3327     FAMILY             most expensive app (H)  399.99
3465  LIFESTYLE                           I'm rich  399.99
3469  LIFESTYLE           I'm Rich - Trump Edition  400.00
4396  LIFESTYLE                          I am rich  399.99
4398     FAMILY                     I am Rich Plus  399.99
4399  LIFESTYLE                      I am rich VIP  299.99
4400    FINANCE                  I Am Rich Premium  399.99
4401  LIFESTYLE                I am extremely Rich  379.99
4402    FINANCE                         I am Rich!  399.99
4403    FINANCE                 I am rich(premium)  399.99
4406     FAMILY                      I Am Rich Pro  399.99
4408    FINANCE     I am rich (Most expensive app)  399.99
4410     FAMILY                          I Am Rich  389.99
4413    FINANCE                          I am Rich  399.99
4417    FINANCE                 I AM RICH PRO PLUS  399.99
8763    FINANCE                        Eu Sou Rico  394.99
8780  LIFESTYLE  I'm Rich/Eu sou Rico/    /    399.99
```



0.8 8. Filter out "junk" apps

It looks like a bunch of the really expensive apps are "junk" apps.

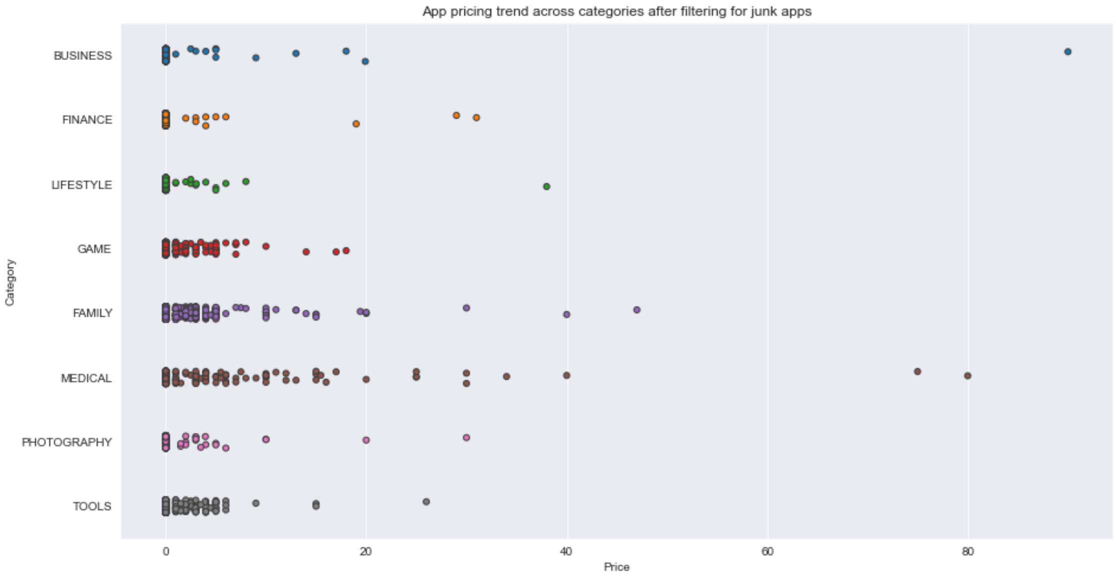
Let's filter out these junk apps and re-do our visualization.

[9]:

```
# Select apps priced below $100
apps_under_100 = popular_app_cats[popular_app_cats['Price']<100]

fig, ax = plt.subplots()
fig.set_size_inches(15, 8)

# Examine price vs category with the authentic apps (apps_under_100)
ax = sns.stripplot(x = 'Price', y = 'Category', data = apps_under_100, jitter =␣
 ↪True, linewidth = 1)
ax.set_title('App pricing trend across categories after filtering for junk␣
 ↪apps')
```

[9]:

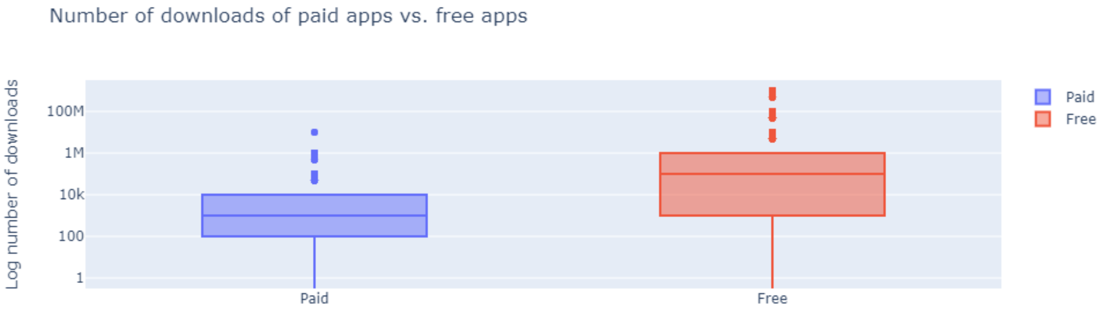
```
Text(0.5, 1.0, 'App pricing trend across categories after filtering for junk
apps')
```



0.9 9. Popularity of paid apps vs free apps

[10]:

```
trace0 = go.Box(
    # Data for paid apps
    y = apps[apps['Type'] == 'Paid']['Installs'],
    name = 'Paid'
)

trace1 = go.Box(
    # Data for free apps
    y = apps[apps['Type'] == 'Free']['Installs'],
    name = 'Free'
)

layout = go.Layout(
    title = "Number of downloads of paid apps vs. free apps",
    yaxis = dict(title = "Log number of downloads",
                type = 'log',
                autorange = True)
)

# Add trace0 and trace1 to a list for plotting
data = [trace0, trace1]
plotly.offline.iplot({'data': data, 'layout': layout})
```

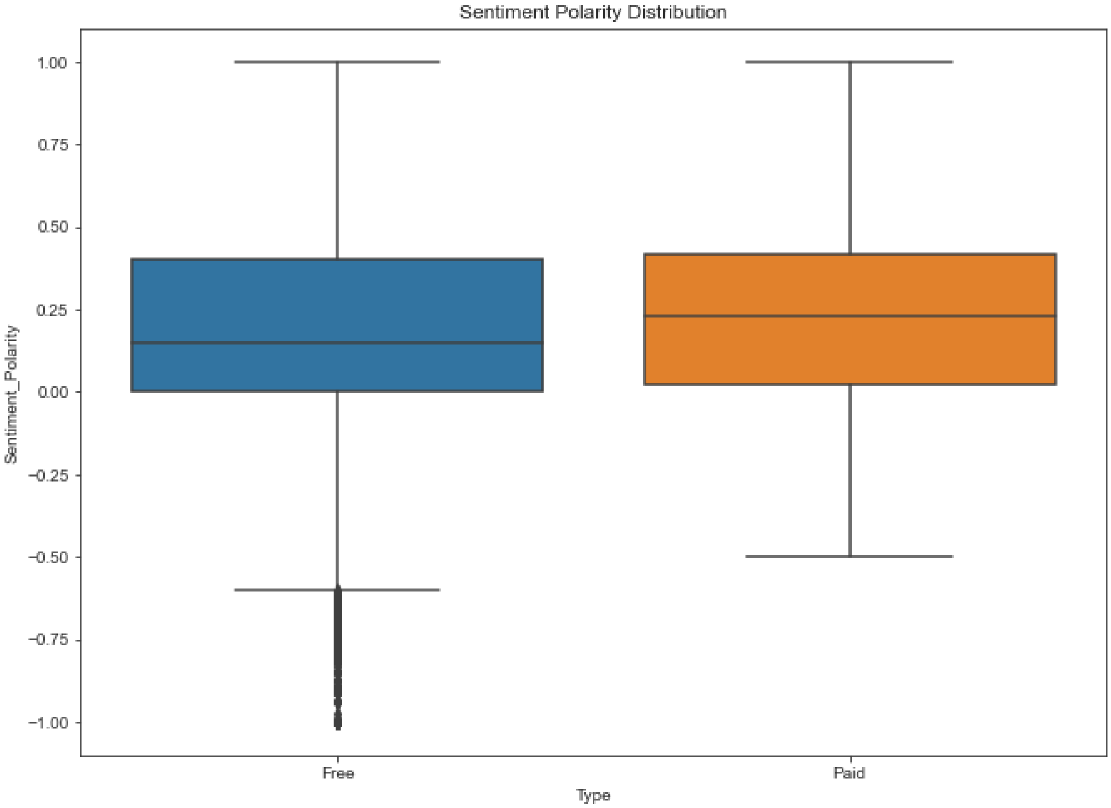


0.10 10. Sentiment analysis of user reviews

By plotting sentiment polarity scores of user reviews for paid and free apps, we observe that free apps receive a lot of harsh comments, as indicated by the outliers on the negative y-axis. Reviews for paid apps appear never to be extremely negative. This may indicate something about app quality, i.e., paid apps being of higher quality than free apps on average. The median polarity score for paid apps is a little higher than free apps, thereby syncing with our previous observation.

[11]:
```
# Load user_reviews.csv
reviews_df = pd.read_csv('datasets/user_reviews.csv')
# Join the two dataframes
merged_df = pd.merge(apps, reviews_df, on = 'App', how = 'inner')

# Drop NA values from Sentiment and Review columns
merged_df = merged_df.dropna(subset = ['Sentiment', 'Review'])

sns.set_style('ticks')
fig, ax = plt.subplots()
fig.set_size_inches(11, 8)

# User review sentiment polarity for paid vs. free apps
ax = sns.boxplot(x = 'Type', y = 'Sentiment_Polarity', data = merged_df )
ax.set_title('Sentiment Polarity Distribution')
```

[11]:
```
Text(0.5, 1.0, 'Sentiment Polarity Distribution')
```


Sentiment Polarity Distribution
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
Sentiment_Polarity
Free
Paid
Type