**[1. Google Play Store apps and reviews](https://app.datacamp.com/workspace/w/7513e937-d0c1-4603-979f-af0cdcc63bed/edit" \l "1-google-play-store-apps-and-reviews)**

# Read in dataset

**import** pandas **as** pd

apps\_with\_duplicates = pd.read\_csv("datasets/apps.csv")

​

# Drop duplicates from apps\_with\_duplicates

apps = apps\_with\_duplicates.drop\_duplicates()

​

# Print the total number of apps

**print**('Total number of apps in the dataset = ', **len**(apps))

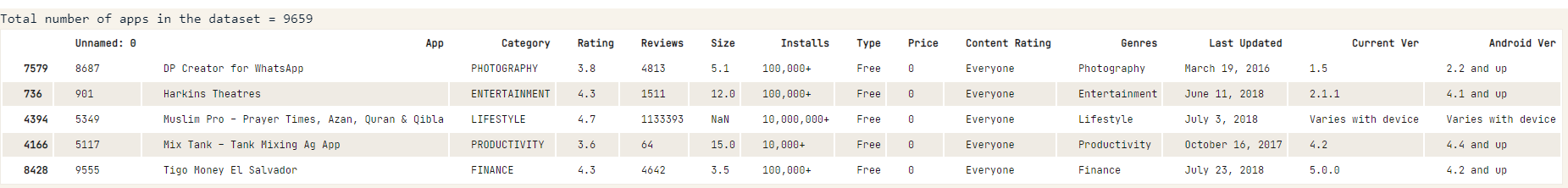
​

# Have a look at a random sample of 5 rows

n = 5

apps.sample(n)

​



[**2. Data cleaning**](https://app.datacamp.com/workspace/w/7513e937-d0c1-4603-979f-af0cdcc63bed/edit#2-data-cleaning)

# List of characters to remove

chars\_to\_remove = ['+', ',', '$']

# List of column names to clean

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 a summary of the apps dataframe

**print**(apps.info())

[**3. Correcting data types**](https://app.datacamp.com/workspace/w/7513e937-d0c1-4603-979f-af0cdcc63bed/edit#3-correcting-data-types)

**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.dtypes)

[**4. Exploring app categories**](https://app.datacamp.com/workspace/w/7513e937-d0c1-4603-979f-af0cdcc63bed/edit#4-exploring-app-categories)

**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(

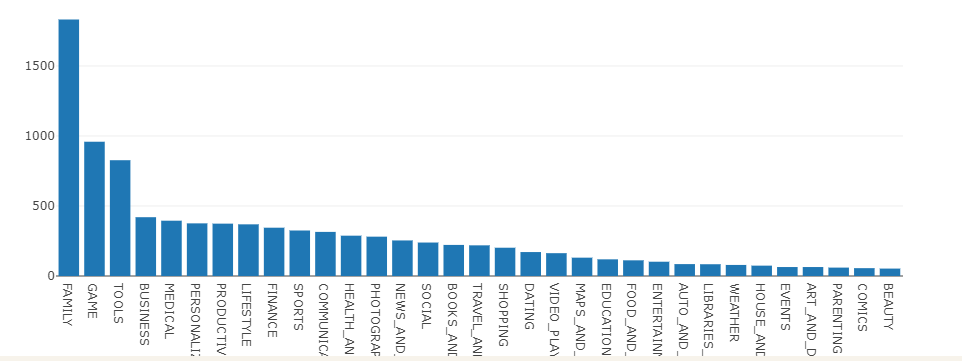
       x = num\_apps\_in\_category.index, # index = category name

       y = num\_apps\_in\_category.values, # value = count

)]

​

plotly.offline.iplot(data)



[**5. Distribution of app ratings**](https://app.datacamp.com/workspace/w/7513e937-d0c1-4603-979f-af0cdcc63bed/edit#4-exploring-app-categories)

# 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,

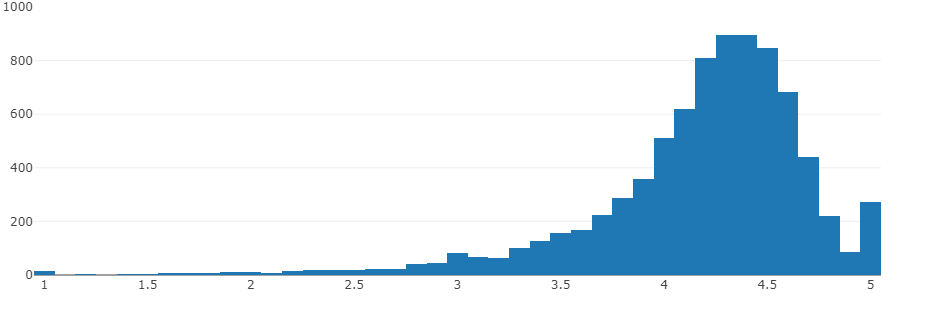
             'line': { 'dash': 'dashdot'}

        }]

        }

​

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



[**5. Size & price of an app**](https://app.datacamp.com/workspace/w/7513e937-d0c1-4603-979f-af0cdcc63bed/edit#4-exploring-app-categories)

%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())]

​

# 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'])

​

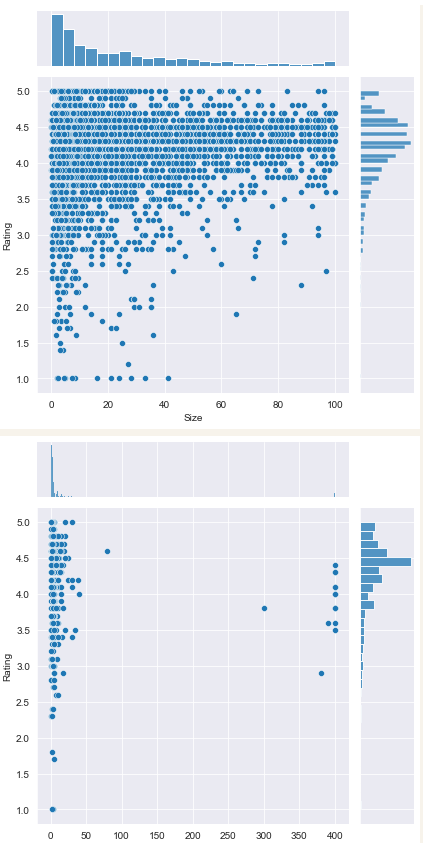
# Select apps whose 'Type' is 'Paid'

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'])



[**7. Relation between app category and app price**](https://app.datacamp.com/workspace/w/7513e937-d0c1-4603-979f-af0cdcc63bed/edit#7-relation-between-app-category-and-app-price)

**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')

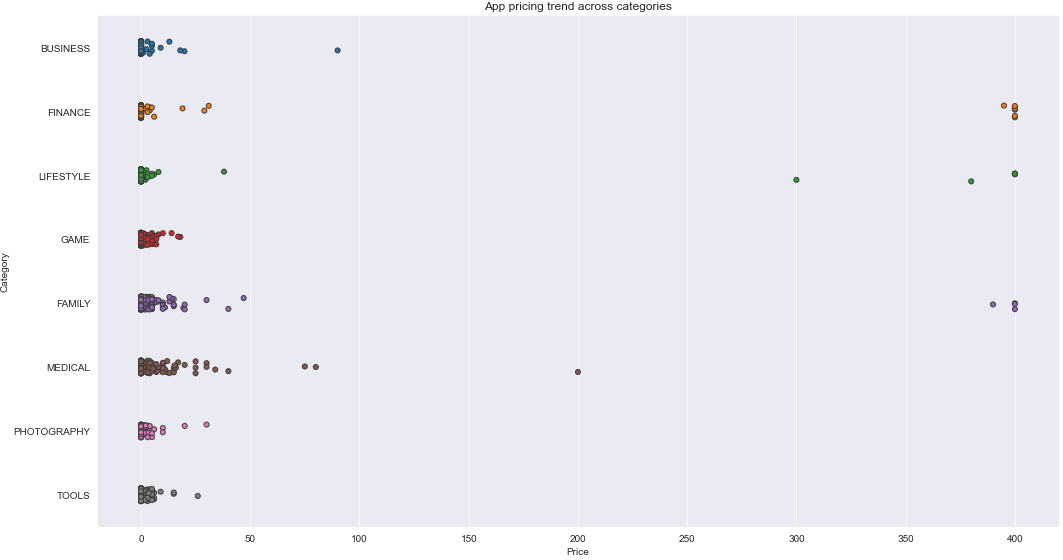
​

# Apps whose Price is greater than 200

apps\_above\_200 = popular\_app\_cats[popular\_app\_cats['Price'] > 200]

apps\_above\_200[['Category', 'App', 'Price']]





[**8. Filter out "junk" apps**](https://app.datacamp.com/workspace/w/7513e937-d0c1-4603-979f-af0cdcc63bed/edit#8-filter-out-junk-apps)

# 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')

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



[**9. Popularity of paid apps vs free apps**](https://app.datacamp.com/workspace/w/7513e937-d0c1-4603-979f-af0cdcc63bed/edit#9-popularity-of-paid-apps-vs-free-apps)

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**)

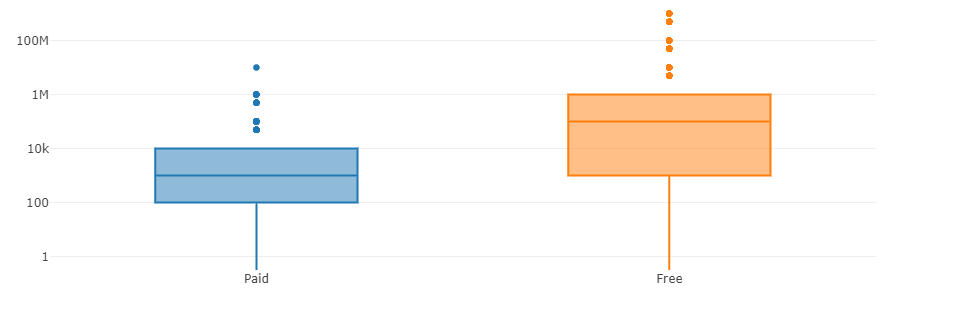
)

​

# Add trace0 and trace1 to a list

data = [trace0, trace1]

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



[**9. Sentiment analysis of user reviews**](https://app.datacamp.com/workspace/w/7513e937-d0c1-4603-979f-af0cdcc63bed/edit#9-popularity-of-paid-apps-vs-free-apps)

# 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")

​

# 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')

