Springboard

The App Project

By
M Tariq Aziz
Data Science (Sep, 22 cohort)

Agenda

- Business Problem
- The Data
- Data Science Pipeline - DSP

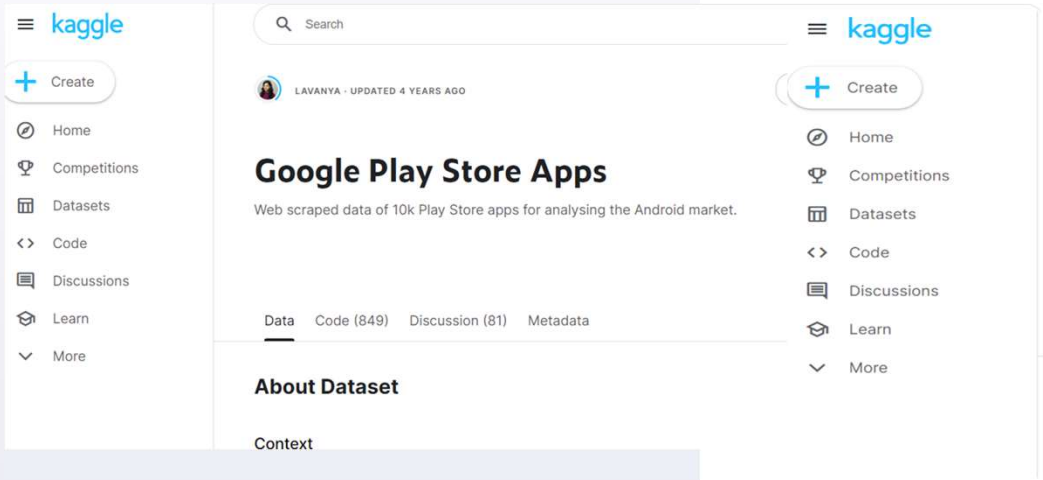
Business Problem

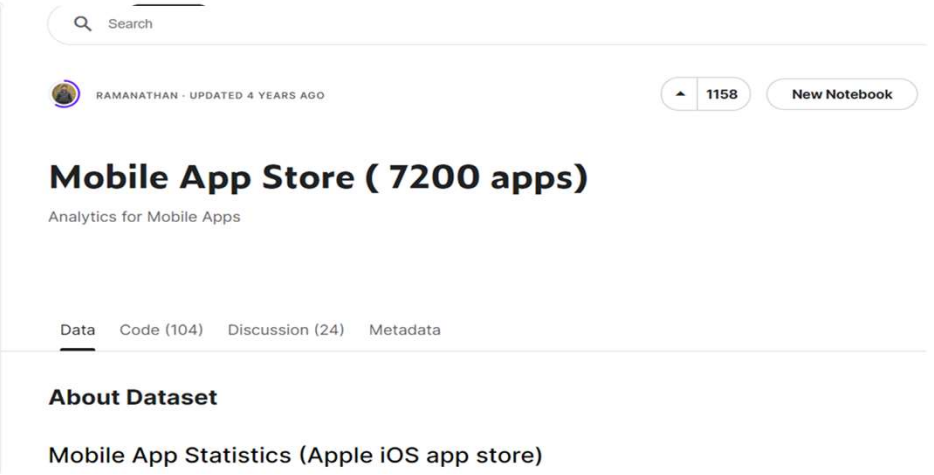
- Marketing consultancy wants to build App store based on higher reviews,

  → Google Play store VS Apple App store

The Data

- Structured datasets from Data Scientists favourite choice ⇒ KAGGLE





Sources:
https://www.kaggle.com/datasets/ramamet4/app-store-apple-data-set-10k-apps
https://www.kaggle.com/datasets/lava18/google-play-store-apps

Data Science Pipeline DSP

- Sourcing & loading data
- Cleaning, transforming & visualizations
- Modeling
- Evaluation & Conclusion

Sourcing & Loading Data

In [1]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
# scipi is a library for statistical tests and visualizations
from scipy import stats
# random enables us to generate random numbers
import random
```

In [67]:

```
# Now that the files are saved, we want to load them into Python using read_csv and pandas.

# Create a variable called google, and store in it the path of the csv file that contains your google dataset.
# If your dataset is in the same folder as this notebook, the path will simply be the name of the file.
google = '/content/sample_data/googleplaystore.csv'

# Read the csv file into a data frame called Google using the read_csv() pandas method.
Google = pd.read_csv(google)

# Using the head() pandas method, observe the first three entries.
Google.head(3)
```

Out[67]:

	App	Category	Rating	Reviews	Size	Installs	Type	Price	Content Rating	Genres	Last Updated	Current Ver	Android Ver
0	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
1	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
2	U Launcher Lite – FREE Live Cool Themes, Hide ...	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

In [68]:

```
# Create a variable called apple, and store in it the path of the csv file that contains your apple dataset.
apple = '/content/sample_data/AppleStore.csv'

# Read the csv file into a pandas DataFrame object called Apple.
Apple = pd.read_csv(apple)

# Observe the first three entries like you did with your other data.
Apple.head(3)
```

Out[68]:

	Unnamed: 0	id	track_name	size_bytes	currency	price	rating_count_tot	rating_count_ver	user_rating	user_rating_ver	ver	cont_rating	prime_genre	sup_devices.num
0	1	281656475	PAC-MAN Premium	100788224	USD	3.99	21292	26	4.0	4.5	6.3.5	4+	Games	38
1	2	281796108	Evernote - stay organized	158578688	USD	0.00	161065	26	4.0	3.5	8.2.2	4+	Productivity	37
2	3	281940292	WeatherBug - Local Weather, Radar,	100524032	USD	0.00	188583	2822	3.5	4.5	5.0.0	4+	Weather	37

Cleaning, Transforming & Visualization (1)

```
# Use the unique() pandas method on the Price column to check its unique values.
Google.Price.unique()
```

```
array(['0', '$4.99', '$3.99', '$6.99', '$1.49', '$2.99', '$7.99', '$5.99',
       '$3.49', '$1.99', '$9.99', '$7.49', '$0.99', '$9.00', '$5.49',
       '$10.00', '$24.99', '$11.99', '$79.99', '$16.99', '$14.99',
       '$1.00', '$29.99', '$12.99', '$2.49', '$10.99', '$1.50', '$19.99',
       '$15.99', '$33.99', '$74.99', '$39.99', '$3.95', '$4.49', '$1.70',
       '$8.99', '$2.00', '$3.88', '$25.99', '$399.99', '$17.99',
       '$400.00', '$3.02', '$1.76', '$4.84', '$4.77', '$1.61', '$2.50',
       '$1.59', '$6.49', '$1.29', '$5.00', '$13.99', '$299.99', '$379.99',
       '$37.99', '$18.99', '$389.99', '$19.90', '$8.49', '$1.75',
       '$14.00', '$4.85', '$46.99', '$109.99', '$154.99', '$3.08',
       '$2.59', '$4.80', '$1.96', '$19.40', '$3.90', '$4.59', '$15.46',
       '$3.04', '$4.29', '$2.60', '$3.28', '$4.60', '$28.99', '$2.95',
       '$2.90', '$1.97', '$200.00', '$89.99', '$2.56', '$30.99', '$3.61',
       '$394.99', '$1.26', 'Everyone', '$1.20', '$1.04'], dtype=object)
```

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

```
array(['0', '$4.99', '$3.99', '$6.99', '$1.49', '$2.99', '$7.99', '$5.99',
       '$3.49', '$1.99', '$9.99', '$7.49', '$0.99', '$9.00', '$5.49',
       '$10.00', '$24.99', '$11.99', '$79.99', '$16.99', '$14.99',
       '$1.00', '$29.99', '$12.99', '$2.49', '$10.99', '$1.50', '$19.99',
       '$15.99', '$33.99', '$74.99', '$39.99', '$3.95', '$4.49', '$1.70',
       '$8.99', '$2.00', '$3.88', '$25.99', '$399.99', '$17.99',
       '$400.00', '$3.02', '$1.76', '$4.84', '$4.77', '$1.61', '$2.50',
       '$1.59', '$6.49', '$1.29', '$5.00', '$13.99', '$299.99', '$379.99',
       '$37.99', '$18.99', '$389.99', '$19.90', '$8.49', '$1.75',
       '$14.00', '$4.85', '$46.99', '$109.99', '$154.99', '$3.08',
       '$2.59', '$4.80', '$1.96', '$19.40', '$3.90', '$4.59', '$15.46',
       '$3.04', '$4.29', '$2.60', '$3.28', '$4.60', '$28.99', '$2.95',
       '$2.90', '$1.97', '$200.00', '$89.99', '$2.56', '$30.99', '$3.61',
       '$394.99', '$1.26', 'Everyone', '$1.20', '$1.04'], dtype=object)
```

- Dollar ($) sign
- Python consider this as String

```
Google[Google['Price']=='Everyone']
```

	Category	Rating	Reviews	Price
10472	1.9	19.0	3.0M	Everyone

```
Google = Google[Google['Price'] != 'Everyone']

# Check again the unique values of Google
Google
```

	Category	Rating	Reviews	Price
0	ART_AND_DESIGN	4.1	159	0
1	ART_AND_DESIGN	3.9	967	0
2	ART_AND_DESIGN	4.7	87510	0
3	ART_AND_DESIGN	4.5	215644	0
4	ART_AND_DESIGN	4.3	967	0
...	...	...	...	...
10836	FAMILY	4.5	38	0
10837	FAMILY	5.0	4	0
10838	MEDICAL	NaN	3	0
10839	BOOKS_AND_REFERENCE	4.5	114	0
10840	LIFESTYLE	4.5	398307	0

10840 rows × 4 columns

```
Apple.dtypes
#Subset_G
```

```
prime_genre          object
user_rating         float64
rating_count_tot      int64
price               float64
dtype: object
```

```
Google.dtypes
```

```
Category     object
Rating      float64
Reviews      object
Price        object
dtype: object
```

Cleaning, Transforming & Visualization (2)

```
nosymb = Google['Price'].str.replace('$','')

# Now we need to do two things:
# i. Make the values in the nosymb variable nume
# ii. Assign this new set of numeric, dollar-sig
# You can do this in one line if you wish.
Google['Price'] = pd.to_numeric(nosymb)
```

```
Google.Price.unique()
```

```
array([  0.  ,   4.99,   3.99,   6.99,   1.49,   2.99,   7.99,   5.99,
         3.49,   1.99,   9.99,   7.49,   0.99,   9.  ,   5.49,  10.  ,
        24.99,  11.99,  79.99,  16.99,  14.99,   1.  ,  29.99,  12.99,
         2.49,  10.99,   1.5 ,  19.99,  15.99,  33.99,  74.99,  39.99,
         3.95,   4.49,   1.7 ,   8.99,   2.  ,   3.88,  25.99, 399.99,
        17.99, 400.  ,   3.02,   1.76,   4.84,   4.77,   1.61,   2.5 ,
         1.59,   6.49,   1.29,   5.  ,  13.99, 299.99, 379.99,  37.99,
        18.99, 389.99,  19.9 ,   8.49,   1.75,  14.  ,   4.85,  46.99,
       109.99, 154.99,   3.08,   2.59,   4.8 ,   1.96,  19.4 ,   3.9 ,
         4.59,  15.46,   3.04,   4.29,   2.6 ,   3.28,   4.6 ,  28.99,
         2.95,   2.9 ,   1.97, 200.  ,  89.99,   2.56,  30.99,   3.61,
       394.99,   1.26,   1.2 ,   1.04])
```

```
# Use the function dtypes.
Google.dtypes
```

```
Category     object
Rating      float64
Reviews      object
Price       float64
dtype: object
```

No dollar ($) sign……YAY 👐

Cleaning, Transforming & Visualization (3)

```
Apple['platform'] = 'apple'
```

```
Google['platform'] = 'google'
```

```
Google[:1]
```

	Category	Rating	Reviews	Price	platform
0	ART_AND_DESIGN	4.1	159	0.0	google

```
Apple[:1]
```

	Category	Rating	Reviews	Price	platform
0	Games	4.0	21292	3.99	apple

Create new column ‘platform’ to join both datasets

```
df = Google.append(Apple, ignore_index = True)

# Using the sample() method with the number 12 passe
df.sample(12)
```

	Category	Rating	Reviews	Price	platform
6190	FAMILY	NaN	1	0.00	google
6409	GAME	4.3	125	0.00	google
2101	FAMILY	4.2	47031	0.00	google
10962	Games	3.5	53821	0.99	apple
8861	FAMILY	5.0	3	0.00	google
2755	SHOPPING	4.5	315908	0.00	google
2406	MEDICAL	4.2	64	29.99	google
11739	Games	4.5	14724	6.99	apple
5622	FAMILY	4.3	107765	0.00	google
10038	PERSONALIZATION	4.3	111634	0.00	google
15101	Games	3.5	256	0.00	apple
4711	FAMILY	4.7	1667	14.99	google

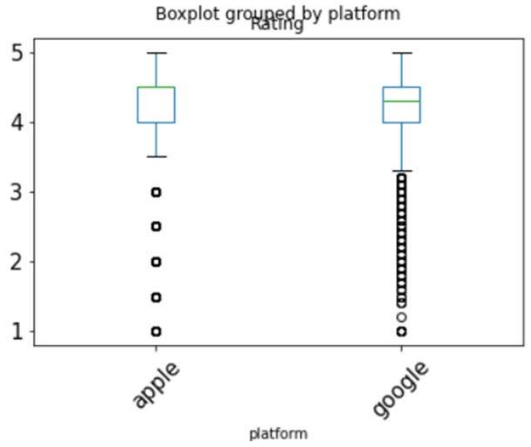
Cleaning, Transforming & Visualization (4)

```
df.groupby(by='platform')['Rating'].describe()
```

platform	count	mean	std	min	25%	50%	75%	max
apple	6268.0	4.049697	0.726943	1.0	4.0	4.5	4.5	5.0
google	9366.0	4.191757	0.515219	1.0	4.0	4.3	4.5	5.0

```
df.boxplot(by='platform', column = ['Rating'], grid=False, rot=45, fontsize=15)
```

```
/usr/local/lib/python3.7/dist-packages/matplotlib/cbook/__init__.py:1376: Visible
(which is a list-or-tuple of lists-or-tuples-or ndarrays with different lengths o
ype=object' when creating the ndarray.
  X = np.atleast_1d(X.T if isinstance(X, np.ndarray) else np.asarray(X))
<matplotlib.axes._subplots.AxesSubplot at 0x7fbb30efce90>
```



Observed Mean Difference = 4.19 - 4.04 = 0.14 (almost the same)

Modeling (1)

```
apple_normal = stats.normaltest(apple)
print(apple_normal)
```

```
NormaltestResult(statistic=1778.9974234584017, pvalue=0.0)
```

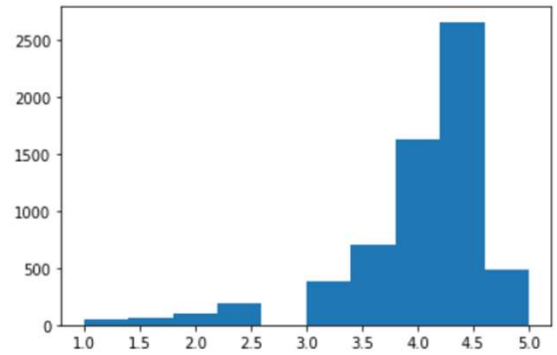
```
# Do the same with the google data.
# Save the result in a variable called google_normal
google_normal = stats.normaltest(google)
print(google_normal)
```

```
NormaltestResult(statistic=3678.6157187516856, pvalue=0.0)
```

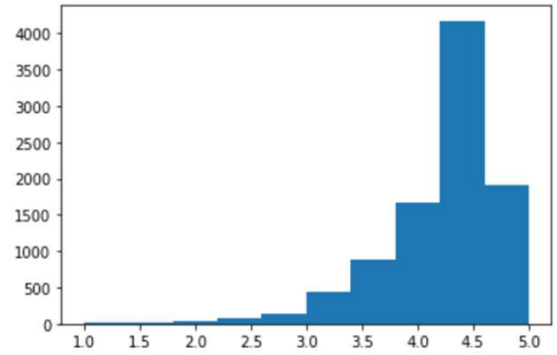
- Data distribution
- Lower p-value ⇒ Data is non-normal (not normally distributed)

Modeling (2)

```
histoApple = plt.hist(apple)
```




```
# Create a histogram of the google data
histoGoogle = plt.hist(google)
```



- Visual representation to verify p-value
- Symmetric
- Unimodal (one-hump)
- Roughly identical mean, median, mode

Modeling (3)

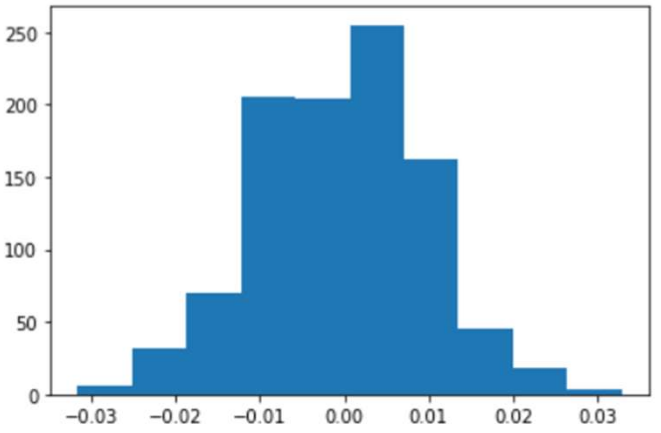
```
df['Permutation1'] = np.random.permutation(df['Rating'])

# Call the describe() method on our permutation grouped by
# We'll use this structure: df.groupby(by='platform')['Per
df.groupby(by='platform')['Permutation1'].describe()
```

platform	count	mean	std	min	25%	50%	75%	max
apple	6268.0	4.127983	0.612738	1.0	4.0	4.3	4.5	5.0
google	9366.0	4.139366	0.613087	1.0	4.0	4.3	4.5	5.0

```
# Lets compare with the previous analytical summary: use d
df.groupby(by='platform')['Rating'].describe()
```

platform	count	mean	std	min	25%	50%	75%	max
apple	6268.0	4.049697	0.726943	1.0	4.0	4.5	4.5	5.0
google	9366.0	4.191757	0.515219	1.0	4.0	4.3	4.5	5.0

```
histo = plt.hist(difference)
```



- Permutation test to check the mean observed difference
- Observed difference = 4.13 - 4.12 = 0.0011 (hugely significant)
- Platform does impact on ratings

Modeling (4)

```
obs_difference = np.mean(apple) - np.mean(google)

# Make this difference absolute with the built-in a
obs_difference = abs(obs_difference)

# Print out this value; it should be 0.142060547451
obs_difference
```

```
0.1420605474512291
```

Evaluation & Conclusion

- Zero difference are as extreme as observed difference
- P-value of the observed is 0
- Platform does impact the rating
- Observed mean of Google play store is 0.14 higher then Apple store
- My choice => build an interface with Google play store