Big Data Analytic project

Google Play Store Apps Analysis

411021206 莊芳儒、411021209 張佩慈、411021231 林芷安

Table of contents

01

Introduction

Introduce the dataset

Data selection

kaggle

LAVANYA · UPDATED 5 YEARS AGO

Google Play Store Apps

Web scraped data of 10k Play Store apps for analysing the Android market.

Data Explorer

Version 6 (9.03 MB)

- googleplaystore.csv
- googleplaystore_user_reviews.csv
- license.txt

Data feature

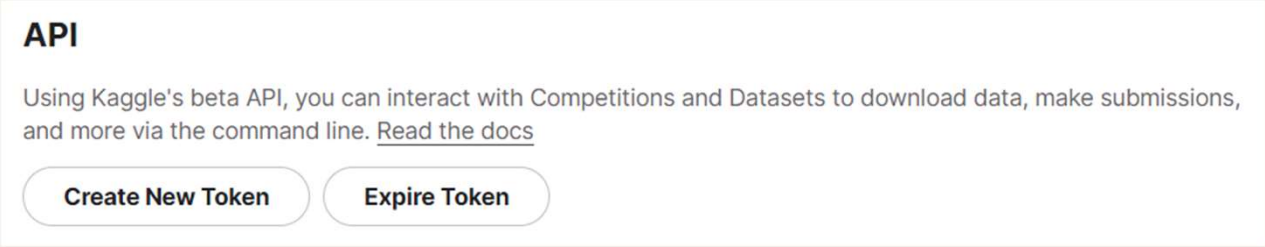
Googleplaystore.csv

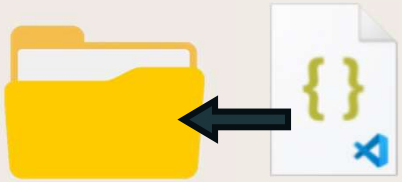
App	The name of the app.	**Price**	The price of the app.
Category	The category of the app.	**Content Rating**	The appropriate target audience of the app.
Rating	The rating of the app.	**Genres**	The genre of the app.
Reviews	The number of reviews of the app.	**Last Updated**	The date when the app was last updated.
Size	The size of the app.	**Current Ver**	The current version of the app.
Install	The number of installs of the app.	**Android Ver**	The minimum Android version required to run the app.
Type	Free or Paid.		

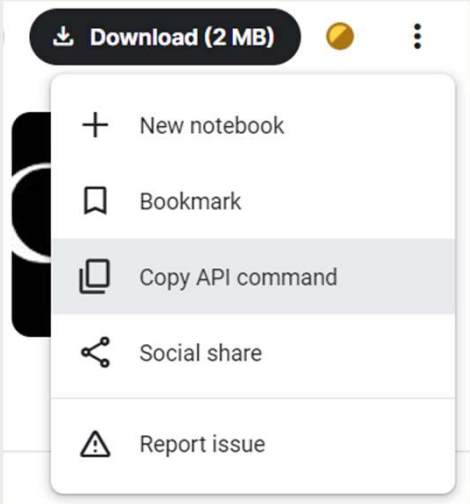
1	App	Category	Rating	Reviews	Size	Installs	Type	Price	Content Rating	Genres	Last Updated	Current Ver	Android Ver
5	Sketch - Draw & Paint	ART_AND_DESIGN	4.5	215644	25M	50,000,000+	Free	0	Teen	Art & Design	8-Jun-18	Varies with device	4.2 and up

Download

```
PS E:\code\python> pip install kaggle
```







Download

```
os.system('kaggle datasets download -d lava18/google-play-store-apps') # kaggle API
os.system('mkdir unzip') # create directory

def extract_files(zip_path, output_dir): # unzip the file
   with zipfile.ZipFile(zip_path, 'r') as zipf:
        zipf.extractall(output_dir)

extract_files('google-play-store-apps.zip', 'unzip’)
path = "C:/Users/asus/.spyder-py3/project/unzip/googleplaystore.csv"
```

02

Analysis

data preprocessing and chart analyzing.

Python Package Index

```
import numpy as np
import pandas as pd
import seaborn as sn
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from sklearn import linear_model
import zipfile
import os
```

Data Preprocessing

Cleaning

Remove the missing value

1	App	Category	
10474	Life Made WI-Fi Touchscreen Photo Frame		1.9
10475	osmino Wi-Fi: free WiFi	TOOLS	

```
data.drop(index = 10472, inplace = True)
a = data.dropna()
```

Feature 'Installs'

Remove '+' and ',' then convert string to number

Installs
10,000+
10,000+

```
a.Installs = a.Installs.apply(lambda x: x.replace('+', ""))
a.Installs = a.Installs.apply(lambda x: x.replace(',', ""))
a.Installs = pd.to_numeric(a.Installs)
```

Data Preprocessing

Feature ‘Price’

Remove ‘$’ then convert string to number

Price
$1.49
0

```
a['Price'] = a['Price'].apply(lambda x: x.replace('$', ""))
a['Price'] = pd.to_numeric(a['Price'])
```

Feature ‘Reviews’

Convert string to number

```
a['Reviews'] = pd.to_numeric(a['Reviews'])
```

Data Preprocessing

Feature ‘Size’

Change the unit to MB

Size
19M
14M

```python
def convert(size):
    if type(size) == str:
        if 'k' in size:
            return format(float(str(size).replace('k', "")) / 1024, '.3f')
        if 'M' in size:
            return float(str(size).replace('M', ""))
    else:
        return(size)
a['Size'] = a['Size'].str.replace('Varies with device', '0.00')
a['Size'] = a['Size'].apply(lambda x: convert(x))
a['Size'] = pd.to_numeric(a['Size’])
a['Size'].fillna(format(a['Size'].mean(), '.3f'), inplace = True )
a['Size'] = pd.to_numeric(a['Size'])
```

Chart Analysis

Category-Count bar

Display the number of apps in each category

```
value_count = a['Category'].value_counts()
value_count.plot(kind = 'bar’,
                 color = 'lightslategray’)
plt.ylabel('Count’)
plt.show()
```

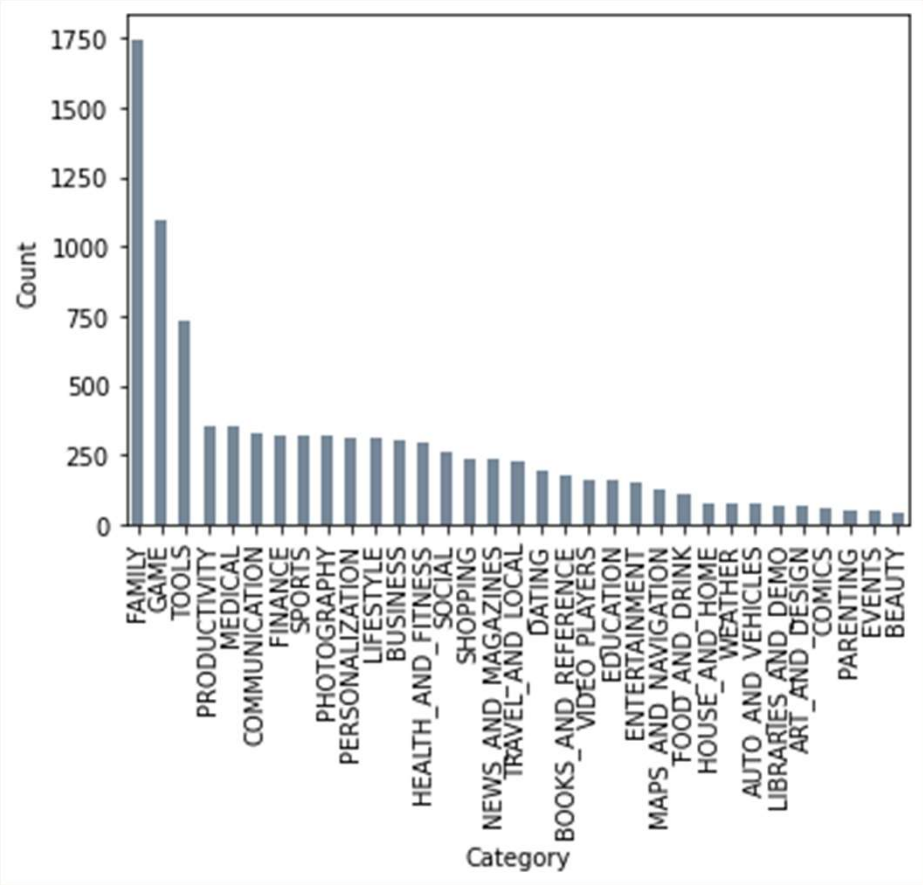


Chart Analysis

Category-Install bar

Display the downloads in each category

```
S = a.groupby('Category')['Installs'].sum()
S.plot.bar(color = 'lightslategray’)
plt.ylabel('Installs’)
plt.show()
```

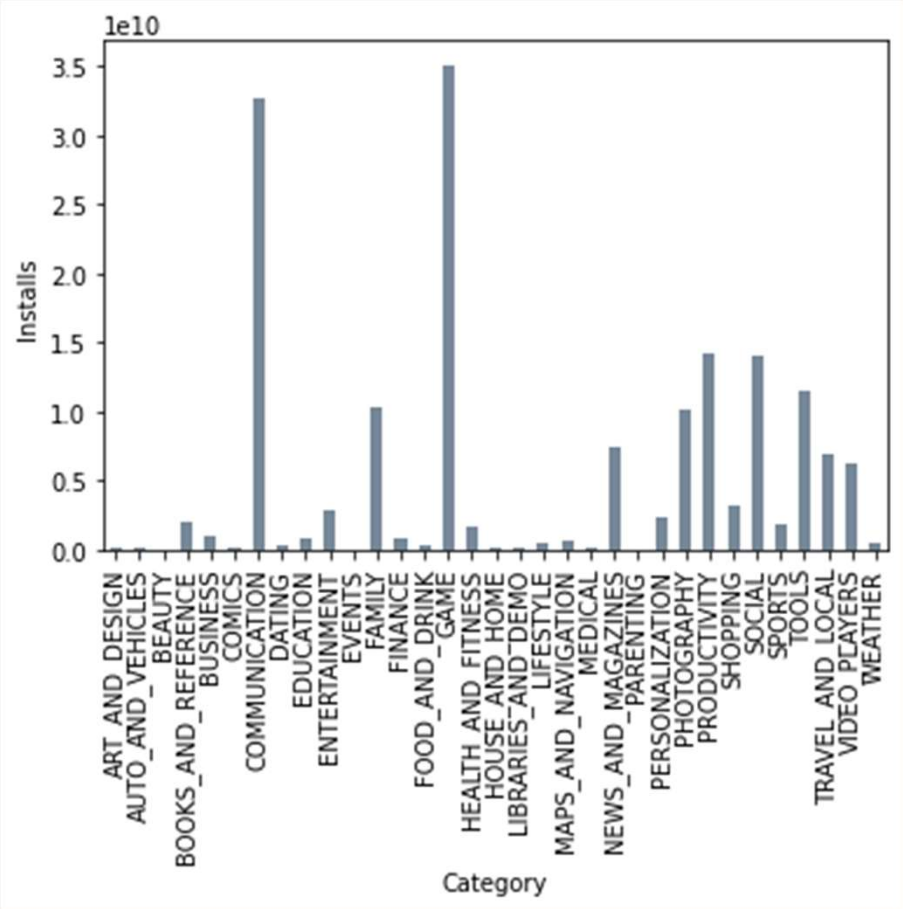


Chart Analysis

Free vs Paid app pie

Display the number of apps in free/paid.

```
C_Freecnt = C_Freec.sum()
C_Paidcnt = C_Paidc.sum()
x = [C_Freecnt, C_Paidcnt]
name = ["Free", "Paid"]
color = ['lightslategray', 'rosybrown’]
plt.pie(x, autopct = ‘%.3f%%’, colors = color,
        labels = name)
plt.show()
```

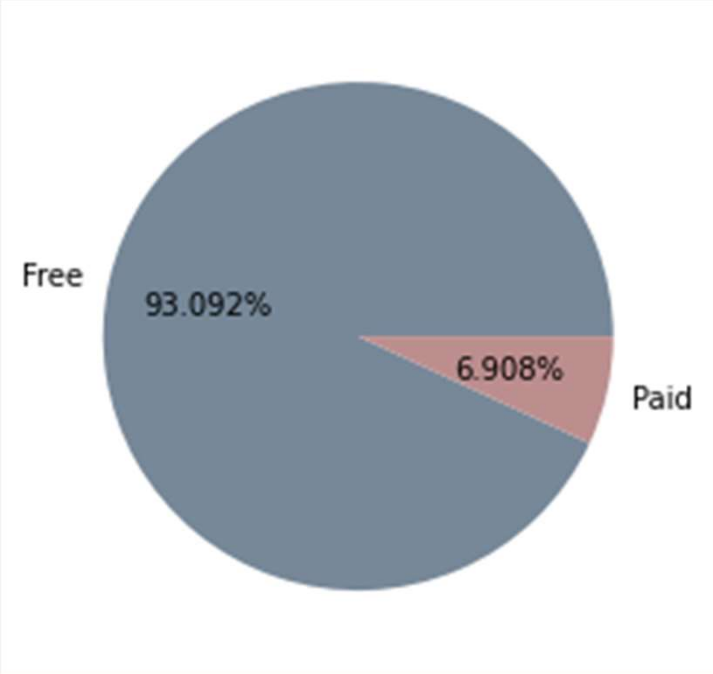


Chart Analysis

Count-Rating in Free app

Display the number of rating in free app.

```
C_Free = a.groupby('Type').get_group('Free’)
C_Freec = C_Free['Rating'].value_counts()
C_Freec.plot.bar(color = 'lightslategray’)
plt.ylabel('Count’)
plt.title('Free’)
plt.show()
```

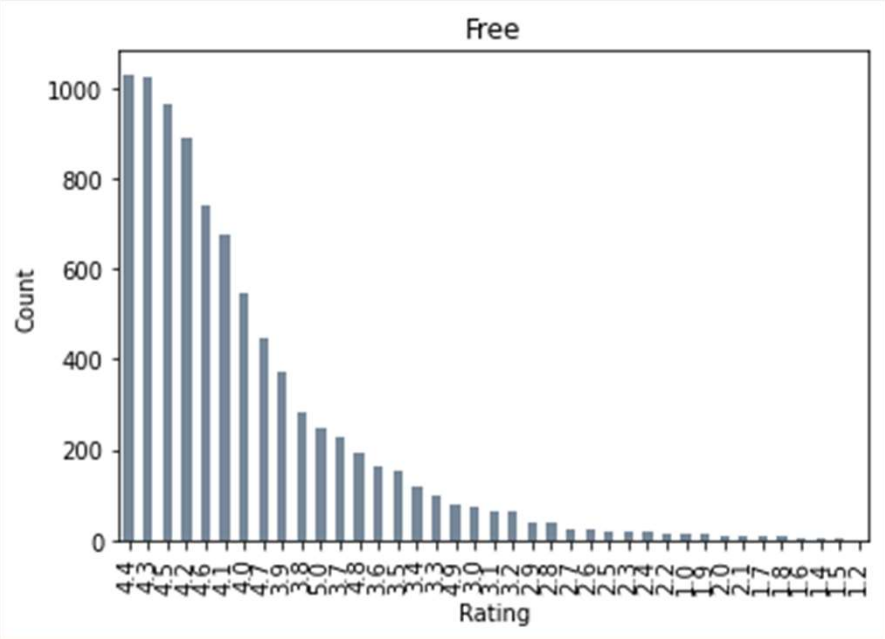


Chart Analysis

Count-Rating in Paid app

Display the number of rating in paid app.

```
C_Paid = a.groupby('Type').get_group('Paid’)
C_Paidc = C_Paid['Rating'].value_counts()
C_Paidc.plot.bar(color = 'lightslategray’)
plt.ylabel('Count’)
plt.title('Free’)
plt.show()
```

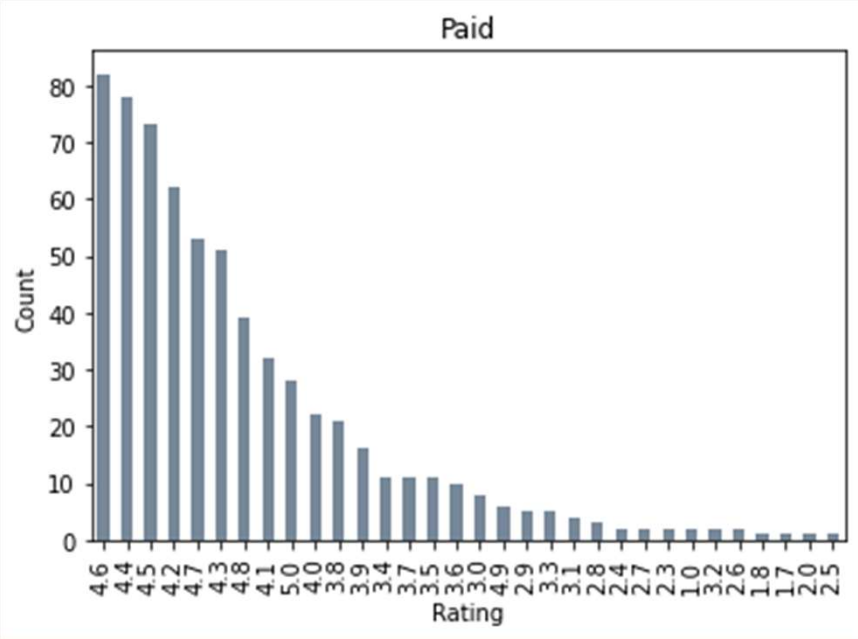
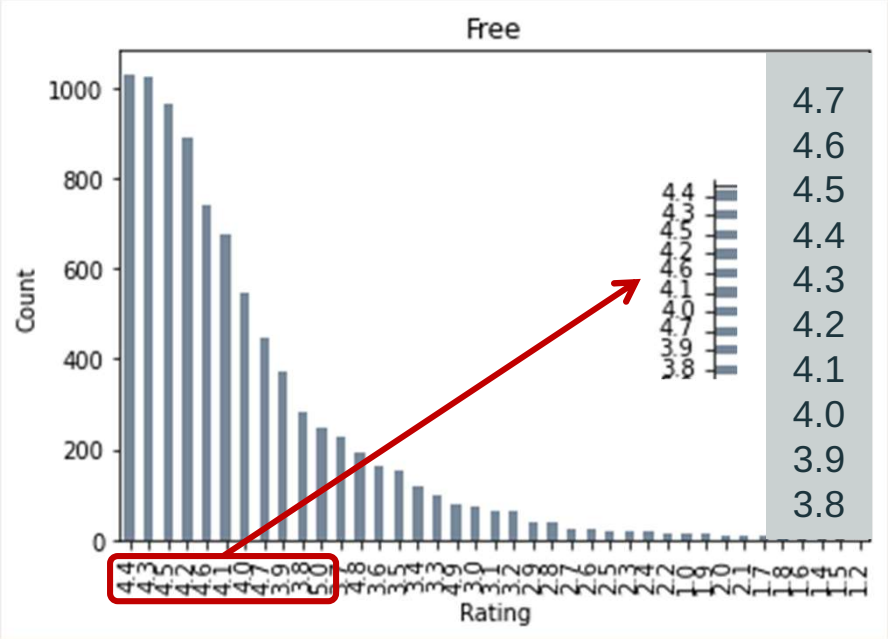
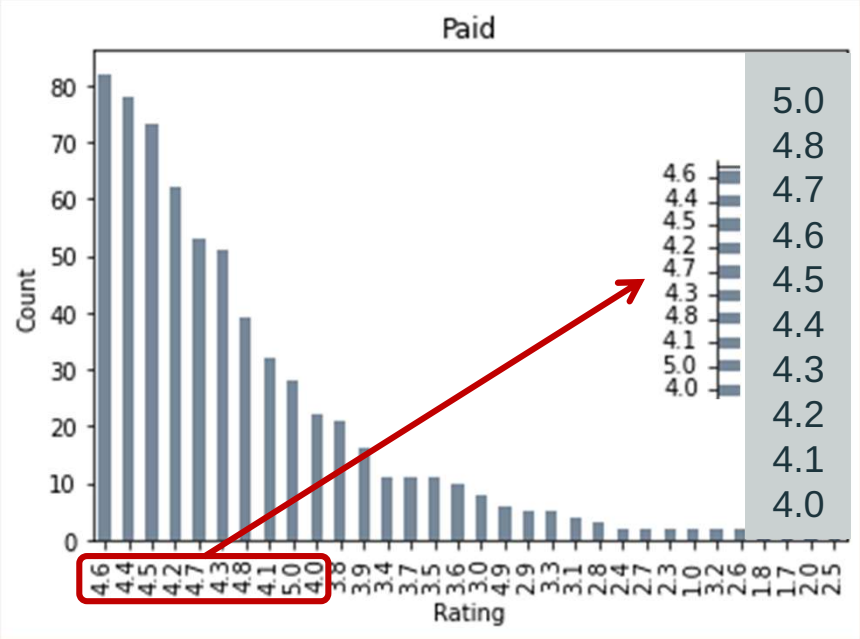


Chart Analysis

Count-Rating: Free vs Paid app





03

Forecasting

Correlation analysis and downloads prediction.

Correlation analysis

Filtering feature

We select features with correlation coefficients greater than 0.05.

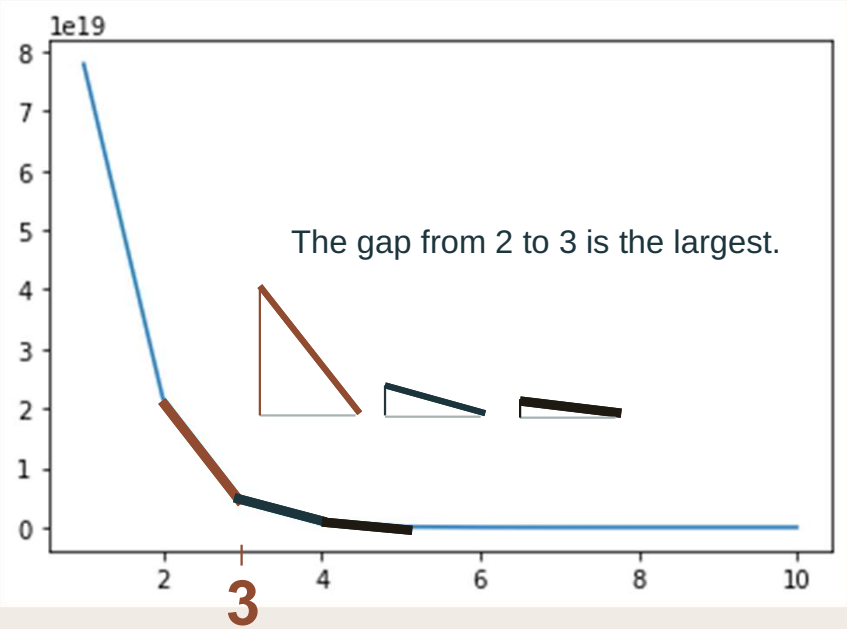
```
b = pd.DataFrame(a.iloc[:,[1, 2, 3, 4, 6]])
corr_matrix = b.corr()
sn.heatmap(corr_matrix, annot = True)
plt.show()
```

Cluster

Rating-Installs

Determine the number of clusters.

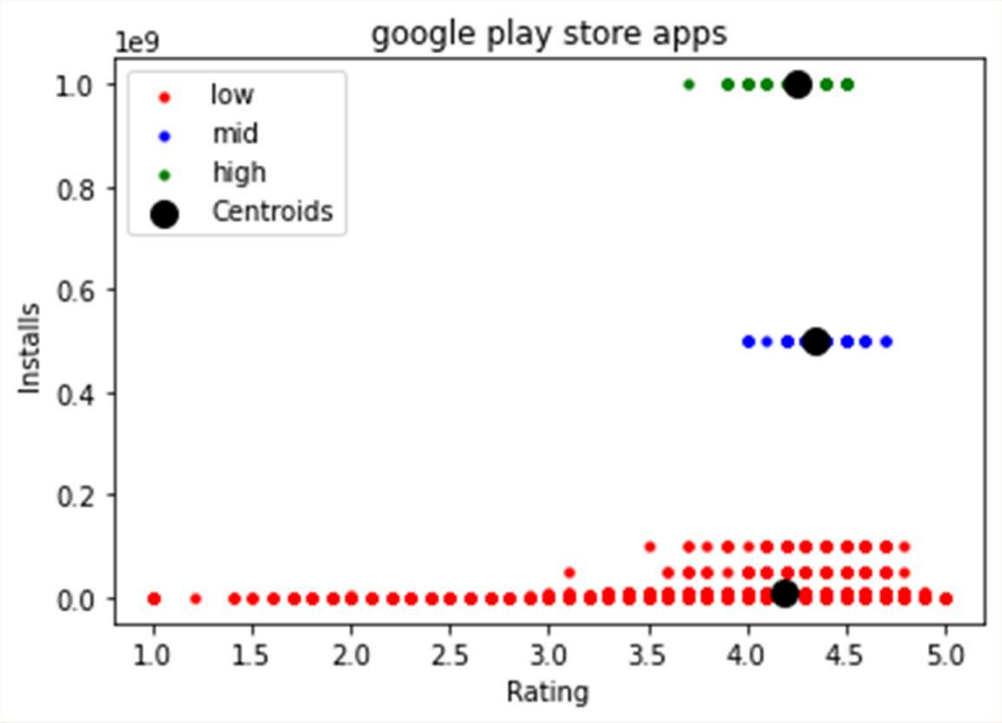
```
X = a.iloc[: ,[1, 4]]
wcss = []
for i in range(1, 11):
    kmeans = KMeans(n_clusters = i,
            n_init = 'auto', random_state = 0)
    kmeans.fit(X)
   wcss.append(kmeans.inertia_)
plt.plot(range(1, 11), wcss)
plt.show()
```



Cluster

```
kmeans = KMeans(n_clusters = 3, init = 'k-means++', random_state = 0)
y_kmeans = kmeans.fit_predict(X)
X = np.array(X)
plt.scatter(X[y_kmeans == 0, 0], X[y_kmeans == 0, 1], s = 10, c = 'red', label = 'low')
plt.scatter(X[y_kmeans == 1, 0], X[y_kmeans == 1, 1], s = 10, c = 'blue', label = 'mid')
plt.scatter(X[y_kmeans == 2, 0], X[y_kmeans == 2, 1], s = 10, c = 'green', label = 'high')
plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1],
                                        s = 100, c = 'black', label = 'Centroids')
plt.title('google play store apps')
plt.xlabel('Rating')
plt.ylabel('Installs')
plt.legend()
plt.show()
```

Cluster

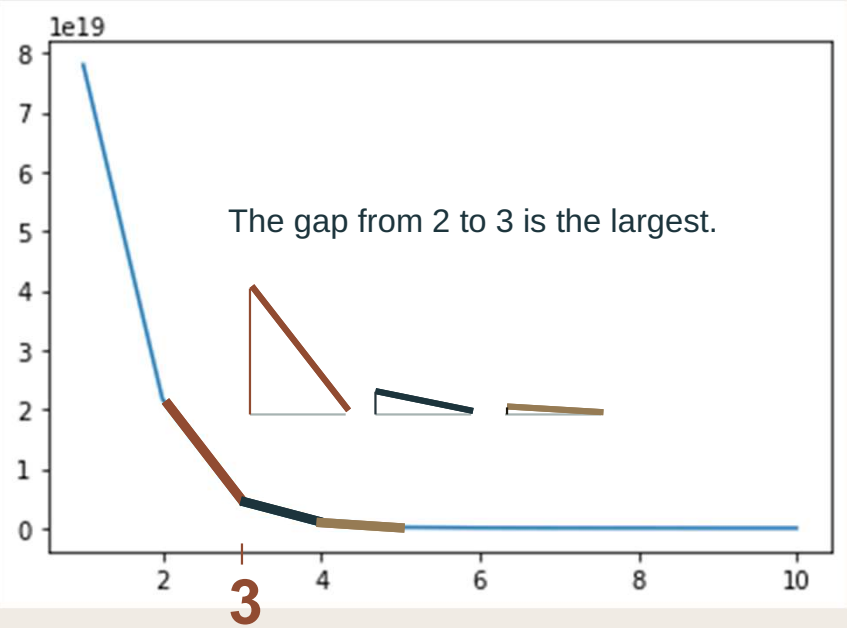

google play store apps
1e9
1.0
0.8
0.6
0.4
0.2
0.0
Installs
low
mid
high
Centroids
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
Rating

Cluster

Reviews-Installs

Determine the number of clusters.

```
X = a.iloc[: ,[2, 4]]
wcss = []
for i in range(1, 11):
    kmeans = KMeans(n_clusters = i,
            n_init = 'auto', random_state = 0)
    kmeans.fit(X)
   wcss.append(kmeans.inertia_)
plt.plot(range(1, 11), wcss)
plt.show()
```



Cluster

```
kmeans = KMeans(n_clusters = 3, init = 'k-means++', random_state = 0)
y_kmeans = kmeans.fit_predict(X)
X = np.array(X)
plt.scatter(X[y_kmeans == 0, 0], X[y_kmeans == 0, 1], s = 10, c = 'red', label = 'low')
plt.scatter(X[y_kmeans == 1, 0], X[y_kmeans == 1, 1], s = 10, c = 'blue', label = 'mid')
plt.scatter(X[y_kmeans == 2, 0], X[y_kmeans == 2, 1], s = 10, c = 'green', label = 'high')
plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1],
                                        s = 100, c = 'black', label = 'Centroids')
plt.title('google play store apps')
plt.xlabel('Reviews')
plt.ylabel('Installs')
plt.legend()
plt.show()
```

Cluster

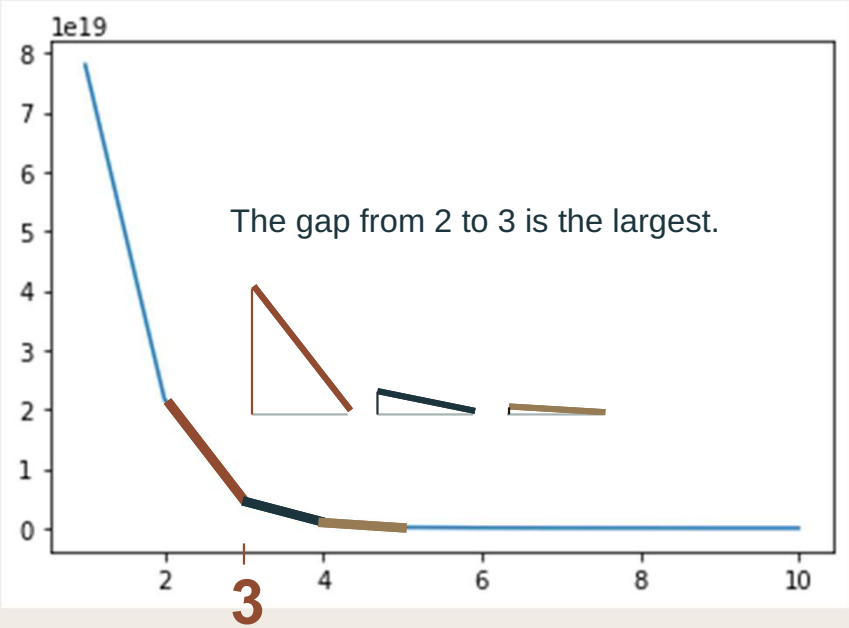

google play store apps
1e9
1.0
0.8
0.6
0.4
0.2
0.0
Installs
0
1
2
3
4
5
6
7
8
Reviews
1e7
low
mid
high
Centroids

Cluster

Size-Installs

Determine the number of clusters.

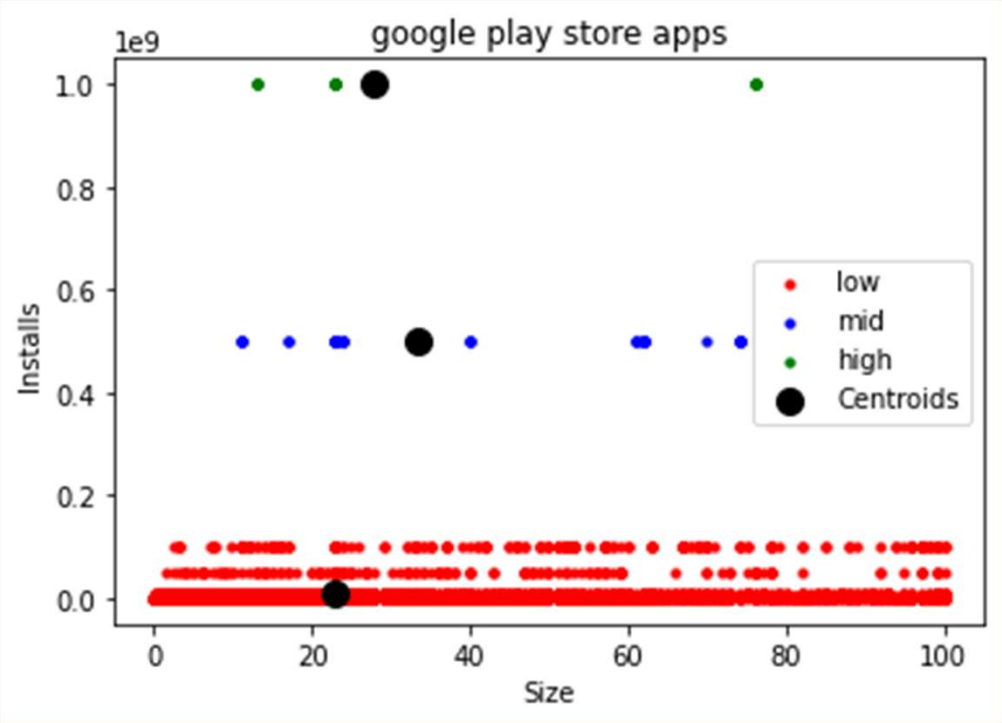
```
X = a.iloc[: ,[2, 4]]
wcss = []
for i in range(1, 11):
    kmeans = KMeans(n_clusters = i,
            n_init = 'auto', random_state = 0)
    kmeans.fit(X)
   wcss.append(kmeans.inertia_)
plt.plot(range(1, 11), wcss)
plt.show()
```



Cluster

```
kmeans = KMeans(n_clusters = 3, init = 'k-means++’, random_state = 0)
y_kmeans = kmeans.fit_predict(X)
X = np.array(X)
plt.scatter(X[y_kmeans == 0, 0], X[y_kmeans == 0, 1], s = 10, c = 'red', label = 'low')
plt.scatter(X[y_kmeans == 1, 0], X[y_kmeans == 1, 1], s = 10, c = 'blue', label = 'mid')
plt.scatter(X[y_kmeans == 2, 0], X[y_kmeans == 2, 1], s = 10, c = 'green', label = 'high')
plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1], s = 100,
                                                c = 'black', label = 'Centroids')
plt.title('google play store apps')
plt.xlabel('Size')
plt.ylabel('Installs')
plt.legend()
plt.show()
```

Cluster


google play store apps
1e9
1.0
0.8
0.6
0.4
0.2
0.0
Installs
0
20
40
60
80
100
Size
low
mid
high
Centroids

Linear Regression

```
# split data into 2 parts, before and after 2018.07
a['Last Updated'] = pd.to_datetime(a['Last Updated'])
before = a[a['Last Updated'] < '2018-07-01']
after = a[a['Last Updated'] >= '2018-07-01']

# select feature and put into traning set and test set
x_train = before[['Rating', 'Reviews', 'Size']]
y_train = before[['Installs']]
x_test = after[['Rating', 'Reviews', 'Size']]
y_test = after[['Installs']]
```

Linear Regression

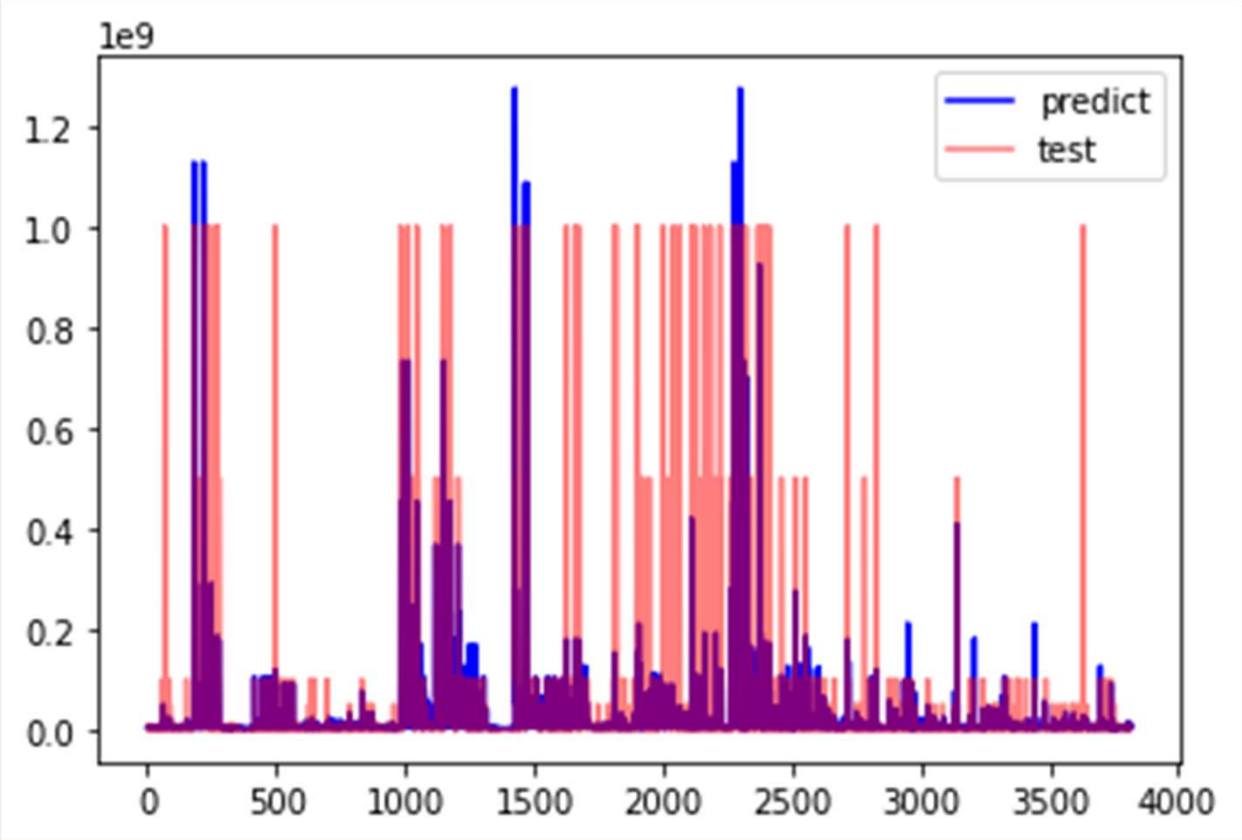
```
# train and construct model
regr1 = linear_model.LinearRegression()
regr1.fit(x_train,y_train)
print("f(X1, X2, X3) = " + str(regr1.coef_[0][0]) + "X1 + " + str(regr1.coef_[0][1]) + "X2 -",
                                str(-regr1.coef_[0][2]) + "X3 + " + str(regr1.intercept_[0]))
```

```
f(X1, X2, X3) = 447582.62405776535X1 + 16.268902752069355X2 - 4744.358168040289X3 + 699848.9572047135
```

Linear Regression

```
# use the trained regression model to predict test dataset's target
y_predict = regr1.predict(x_test)
plt.plot(range(len(y_predict)), y_predict, 'b', label = "predict")
plt.plot(range(len(y_predict)), y_test, 'r', label = "test", alpha = 0.5)
plt.legend(loc = "upper right")
plt.show()
```

Prediction


1e9
1.2
1.0
0.8
0.6
0.4
0.2
0.0
0
500
1000
1500
2000
2500
3000
3500
4000
predict
test

Prediction

```
print("train_accuracy : ", regr1.score(x_train, y_train)) # accuracy
print("test_accuracy : ", regr1.score(x_test, y_test))
```

Output :

```
train_accuracy :  0.34033503614315164
test_accuracy :  0.38172182613702543
```

```
b = pd.DataFrame(y_predict, columns=['Predict'])
b.to_csv("test.csv", index = False)
```

Output :

```
[[4185903.05404785]
 [2790748.31718829]
 [2755251.32242758]
 ...
 [2650401.53661713]
 [2920747.4636996 ]
 [9103845.80874038]]
```

04

Demonstration

Run the Code

Thanks!