Big Data Analytics Project

Google Play Store Apps Analysis

411021206 莊芳儒

411021209 張佩慈

411021231 林芷安

1. Dataset

來自網站 Kaggle，網址如下

https://www.kaggle.com/datasets/lava18/google-play-store-apps/data

選擇檔案為”googleplaystore.csv” 共 10842 筆

A	B	C	D	E	F	G	H	I	J	K	L	M
App	Category	Rating	Reviews	Size	Installs	Type	Price	Content Rating	Genres	Last Updated	Current Ver	Android Ver

使用到的項目有:

Category、Rating、Reviews、Size、Installs、Type、Price、Last Updated

2. 程式碼

主要分成 7 個部分:

①下載資料集

```
# kaggle API
os.system('kaggle datasets download -d lava18/google-play-store-apps')
# create directory
os.system('mkdir unzip')
# extract $zip_path and put it into $output_dir
def extract_files(zip_path, output_dir):
    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/unzip/googleplaystore.csv"
```

②資料預處理

原資料呈現:

	App	Category	Rating	Reviews	Size	Installs	Type	Price	Content Rating	Genres	Last Updated	Current Ver	Android Ver
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

```
usecols = ['Category', 'Rating', 'Reviews', 'Size', 'Installs',
                                            'Type', 'Price', 'Last Updated']
data = pd.read_csv(path, usecols=usecols)
#cleaning
data.drop(index=10472, inplace=True)
a = data.dropna()
# installs
Installs = a.Installs.apply(lambda x: x.replace('+', ""))
Installs = a.Installs.apply(lambda x: x.replace(',', ""))
Installs = pd.to_numeric(a.Installs, errors="coerce")
```

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

③圖表分析

```
# category-count-bar
value_count = a['Category'].value_counts()
value_count.plot(kind='bar', color='lightslategray')
plt.ylabel('Count')
plt.show()
# category-Install-bar
S = a.groupby('Category')['Installs'].sum()
S.plot.bar(color = 'lightslategray')
plt.ylabel('Installs')
plt.show()
# rating-Count-bar(Free,Paid)
C_Free = a.groupby('Type').get_group('Free')
C_Paid = a.groupby('Type').get_group('Paid')
C_Freec = C_Free['Rating'].value_counts()
C_Paidc = C_Paid['Rating'].value_counts()
C_Freec.plot.bar(color='lightslategray')
plt.ylabel('Count')
plt.title('Free')
plt.show()
C_Paidc.plot.bar(color='lightslategray')
plt.ylabel('Count')
plt.title('Paid')
```

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

④相關性分析

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

⑤聚類

rating-install

```python
X = a.iloc[:, [1, 4]]
# elbow
wcss = []
for i in range(1, 11):
    kmeans = KMeans(n_clusters=i, n_init='auto', random_state=42)
    kmeans.fit(X)
    wcss.append(kmeans.inertia_)
plt.plot(range(1, 11), wcss)
plt.show()
# cluster show
kmeans = KMeans(n_clusters=3, init='k-means++', random_state=42)
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()
```

review-install

```
X = a.iloc[:, [2, 4]]
# elbow
wcss = []
for i in range(1, 11):
    kmeans = KMeans(n_clusters=i, n_init='auto', random_state=42)
    kmeans.fit(X)
    wcss.append(kmeans.inertia_)
plt.plot(range(1, 11), wcss)
plt.show()
# cluster show
kmeans = KMeans(n_clusters=3, init='k-means++', random_state=42)
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()
```

size-install

```
X = a.iloc[:, [3, 4]]
# elbow
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 show
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()
```

⑥預測

數據分類依日期分配

```
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']
# divide data in 2 parts, 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']]
```

數據分類是電腦隨機分配

```
a['Last Updated'] = pd.to_datetime(a['Last Updated'])
x = a[['Rating', 'Reviews', 'Size']]
y = a[['Installs']]
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3, 
random_state=42)
```

線性迴歸

```
# 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]))
# 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()
# accuracy
print("train_accuracy : ", regr1.score(x_train, y_train))
print("test_accuracy : ", regr1.score(x_test, y_test))
b = pd.DataFrame(y_predict, columns=['Predict'])
b.to_csv("test.csv", index=False)
```

決策樹

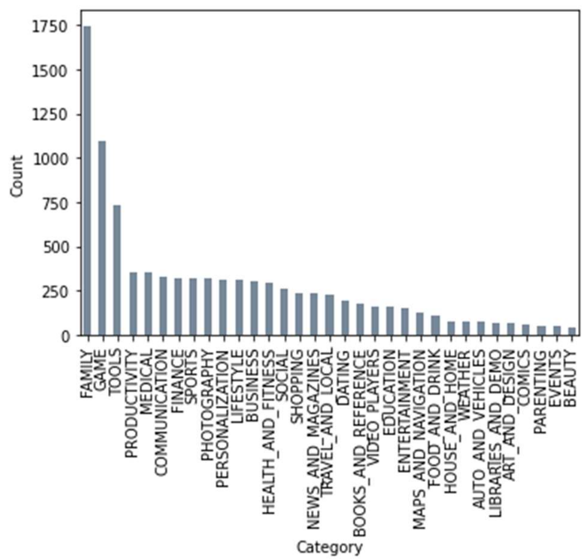
```
dct = DecisionTreeRegressor()
dct.fit(x_train, y_train)
print("test_accuracy : ", dct.score(x_test, y_test))
```

隨機森林迴歸器

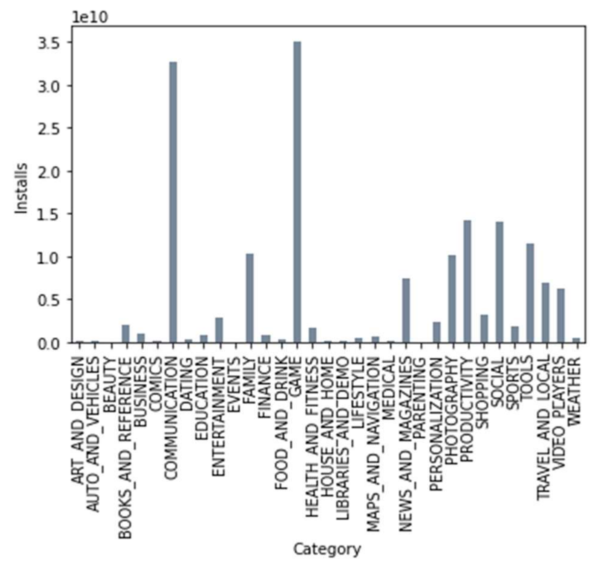
```
Randomforestreg = RandomForestRegressor(n_estimators=100, n_jobs=-1, oob_score=True,
bootstrap=True, random_state=42)
Randomforestreg.fit(x_train, y_train)
y_prediction_randomforest = Randomforestreg.predict(x_test)
randomforestscore = Randomforestreg.score(x_test, y_test)
print("test_accuracy : ", randomforestscore)
```

3. 測試結果展示

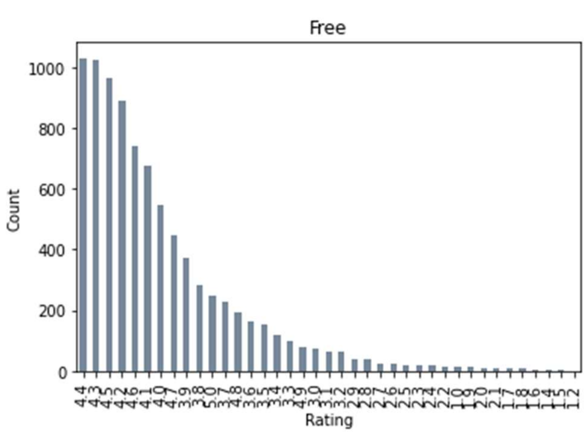
■ Category-Count Bar



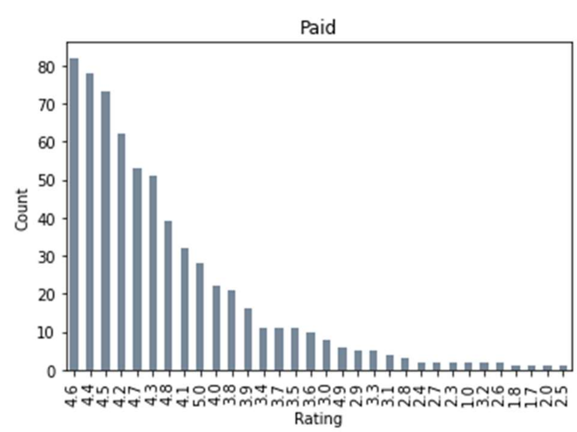
■ Category-Install Bar



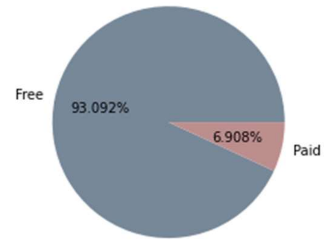
■ Count-Rating in Free app



■ Count-Rating in Paid app



■ Free vs Paid app pie



Correlation

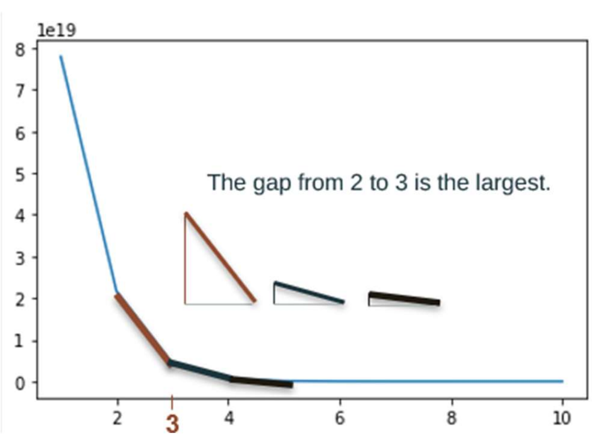
我們選擇相關係數大於 0.05 的項目: Rating、Reviews、Size。

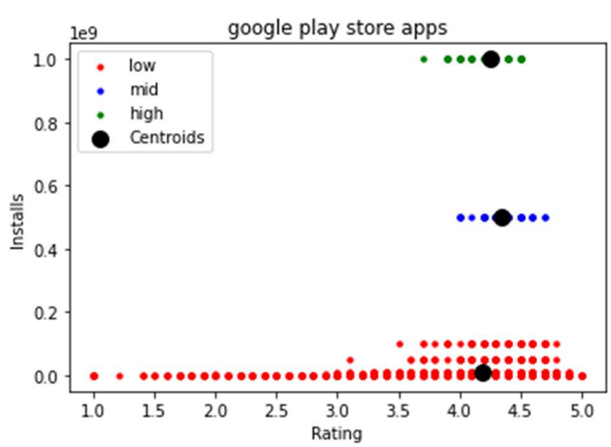
Clustering

在每個 clustering 的圖中會發現，級距很明顯且彼此之間都沒有其他數據，這是因為數據 Installs 原本是以 ‘+’ 代替後面的非整數，因此最後處理完的結果只會有 $5 * 10^n$ 和 $1 * 10^n$ 兩種類型的數據，如右圖。此外，圖中單位為 10^9，所以 low 為小於等於 10^8，mid 為 $5 * 10^8$，high 為大於等於 10^9。

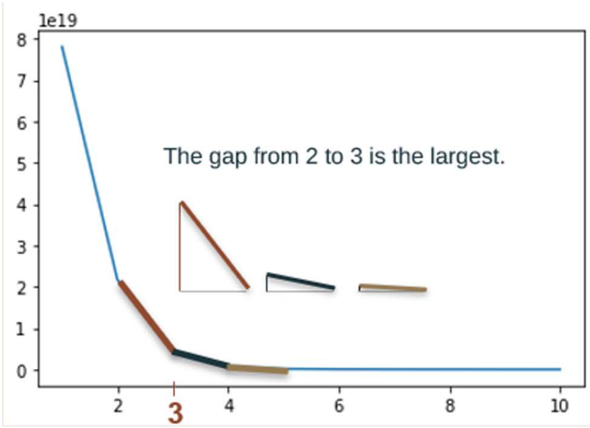
Installs	
1000000	1577
10000000	1252
100000	1150
10000	1010
5000000	752
1000	713
500000	538
50000	467
5000	432
100000000	409
100	309
50000000	289
500	201
500000000	72
10	69
1000000000	58
50	56
5	9
1	3

■ Install-Rating

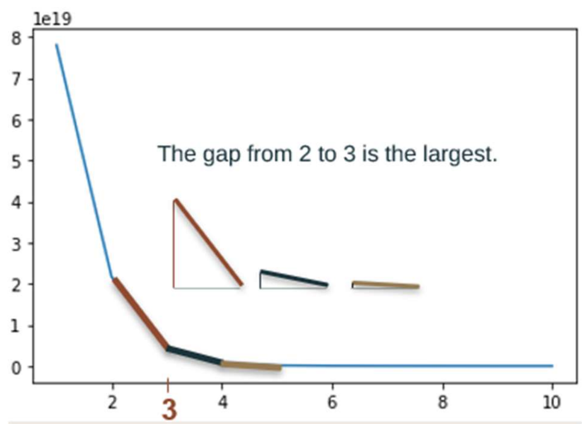


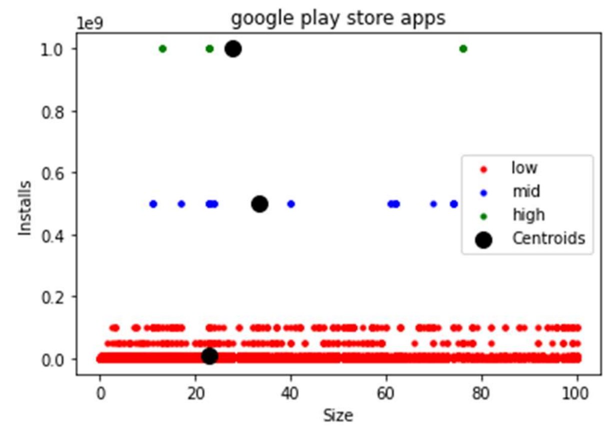


■ Install-Reviews



■ Install-Size



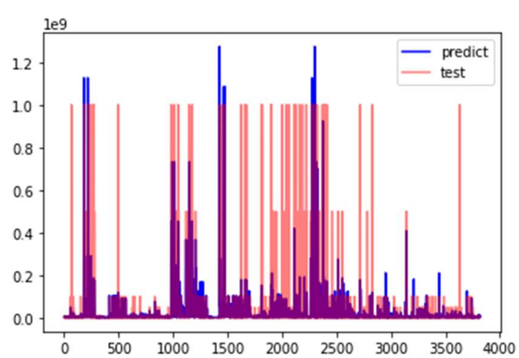


Forecasting

■ Linear regression

由下圖可知 x 係數皆沒有過小的，因此沒有再進一步刪除項目而是直接預測

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



✓ 依日期分類預測

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

✓ 電腦隨機分類預測(此準確率並非固定值，但平均大於 70%)

```
test_accuracy :  0.8499635555700153
```

- Decision tree
  - ✓ 依日期分類預測

    ```
    test_accuracy :  0.2389810917196915
    ```

  - ✓ 電腦隨機分類預測(此準確率並非固定值，但平均大於 70%)

    ```
    test_accuracy :  0.857678394220075
    ```

- Random forest regressor
  - ✓ 依日期分類預測

    ```
    test_accuracy :  0.2521013309986284
    ```

  - ✓ 電腦隨機分類預測(此準確率並非固定值，但平均大於 70%)

    ```
    test_accuracy :  0.8622321162661485
    ```

4. 心得與反思

做完此報告後才理解到老師所說的軍火展示跟真正分析的差別，原本我們的想法是偏於軍火展示，經過與老師的討論後才明白要確定目的並為此目的篩選所需項目，但那時已交完計畫報告，所以聚類分析即使後來覺得好像沒什麼用依然選擇做出來。預測的部分，由於準確率很低，所以我們又嘗試了其他種方法 (decision tree、random forest) 去提升它的準確率，但效果皆不顯著。除非測試集不分日期而是採電腦隨機分配，準確率才會超過 50%。對此，我們覺得數據集可能選錯了，我們到最後才發現 Install 這個數據並不是一個準確數據而是一個範圍數據，所以最後呈現的結果看起來不正常。經過這學期的實作練習，雖然最後專題結果不是很好，但對於大數據分析有比基本了解再更深入了，也學到一些網路爬蟲技巧以及分析數據相關技巧。