Data Science Final Repot

Title: What the fxxc does this have to do with app rating?!

Group: 19

Group Member: 吳治緯(r07942152)、林士鈞(b05501032) 、張哲銘(r07942147)

The Data(資料集介紹):

我們使用的是 Kaggle 網站中 17K Mobile Strategy Games 的資料集，裡面包含了 Apple Appstore 中所有的策略遊戲，我們的目標是根據資料的各種屬性預測出擁有某些特徵的遊戲會有比較高的使用者評價 (Average User Rating)。

資料集中包含了以下表格的資訊，其中與遊戲無關的屬性會在資料預處理時被剔除 (e.g. URL、ID)，另外有些文字型態的資料也會被轉換成 List 型態以便於分析 (e.g. Language、Genres 轉換成 List)，我們會用前一筆資料補缺失的資料。

The columns(attributes) of the Data Source: → 18 個 attributes

Column Name	Type	Simple Idea
URL	Text(url)	need to drop
ID	Number(key_id)	need to drop
Name	Text (Title)	Word Cloud
Subtitle	Text (Sub_Title)	Word Cloud
Icon URL	Text(url)	Crawling Image
Average User Rating	Float(Score)	Countplot
User Rating Count	Int(Counting)	Outlier, Pie Chart
Price	Float(Price)	Outlier, Pie Chart
In-app Purchases	Float(Price)	preprocessing, minmax graph
Description	Text	need to drop
Developer	Text	WordCloud?
Age Rating	Int(Ordinal)	'+' remove
Language	Text	TreeMap, Countplot
Size	Int (Byte)	Unit Converting, distplot
Primary Genre	Text	treemap
Genres	Text	treemp, network graph
Original Release Date	Date	Time Series
Current Version Release Date	Date	Time Series

One row of the Data Source(刪除了 “URL” 和 “ID”):

Name	Subtitle	Icon_URL	Average_User_Rating	User_Rating_Count	Price	In_app_Purchases	Description	Developer
Antivirus	NaN	https://is4-ssl.mzstatic.com/image/thumb/Purpl...	3.0	187.0	0.00	1.99	"NEW Antivirus V1.2 just released!!! \nAntivir...	DeadRatGames

Developer	Age_Rating	Size	Primary_Genre	Genres	Original_Release_Date	Current_Version_Release_Date
DeadRatGames	9+	25563136.0	Games	Games, Entertainment, Simulation, Strategy	3/02/2009	9/07/2015

How I process it(Data Source): → 預處理

Step1: 先將原始的 data 使用 pd.read_csv 讀取進來

Code:
```
MobileGame_data = pd.read_csv('appstore_games.csv')
```

Step2: 去除不重要的 attributes. Ex: “URL” and “ID”

Step3: 由於遊戲的 size 大小是以 bytes 來進行表式，我先對其進行處理，轉換換成 MB 的形式.

Code:
```
MobileGame_data['Size'] = MobileGame_data['Size'].apply(lambda b : b
                                                        //((2**10)*1000))
```

Step4: 將“Price”這個 column 內的 data 進行 ceil 處理，以方便後面進行運算

Step5: 將“In app purchase”這個 column 內的 data 進行取值處理時，因為有些 Row 中的售價會有太多,EX: `['1.99', ' 0.99', ' 1.99', ' 0.99', ' 4.99', ' 1.99', ' 1.99']`

因此只取最大值，以及浮點數問題，之後對 data 進行 ceil 處理，以方面後方分析 data 運算.

Step6: 將“Geners”的分類進行區分之後，用數字表示不同的類別，以方面後續續處理.

The tools (open ones):

Python 套件：

1. **import** pandas
2. **import** numpy
3. **import** time
4. **import** math
5. **import** seaborn
6. **import** matplotlib
7. **from sklearn import** metrics
8. **from bokeh.plotting import** figure
9. **from sklearn.model_selection import** train_test_split

```
10. from bokeh.io import show, output_notebook
11. from sklearn.metrics import classification_report,confusion_matrix
```

所使用的預測 Model:

1. **from sklearn.ensemble import** RandomForestClassifier
2. **from sklearn.tree import** DecisionTreeClassifier
3. **from sklearn.neighbors import** KNeighborsClassifier

我們在後續 Methodology 和 Implementation 和 finding 部分，

分別使用了三種不同預測模型去進行分析:

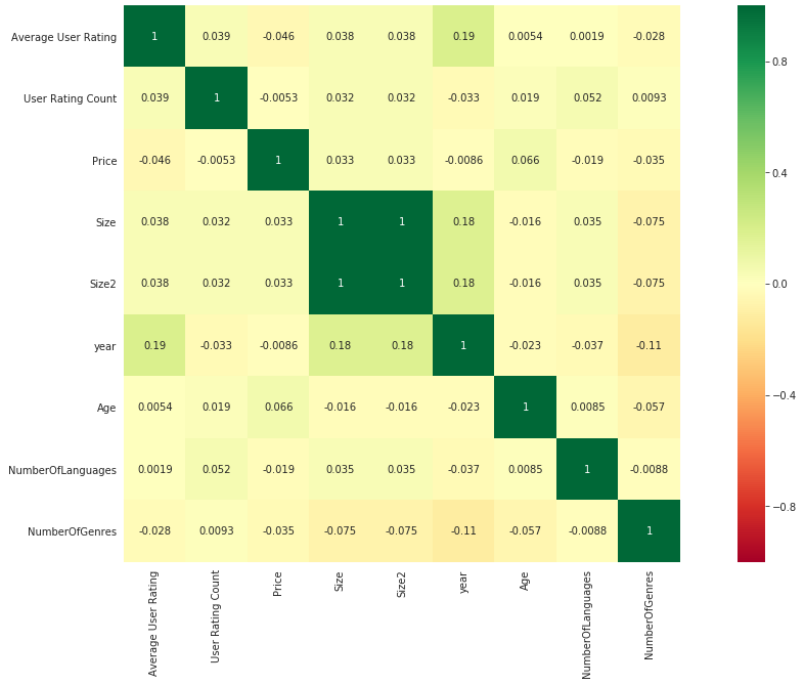
My methodology – 1: KNN(KNeighborsClassifier)

我們的分析大致分成兩個部分，在初步分析時，會先淘汰掉相關性較低的屬性，此舉是為了避免預測模型分析時發生 Curse of Dimensionality。

初步分析(剔除不重要的 attributes)：

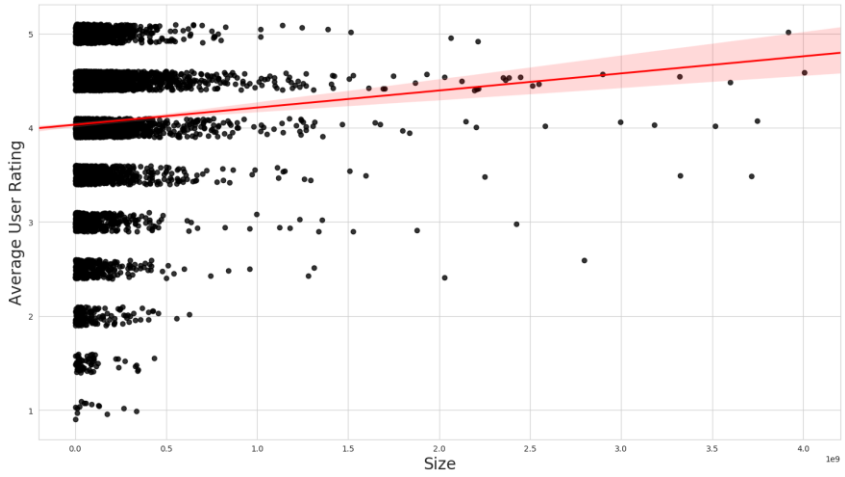
1. Correlation Heatmap:

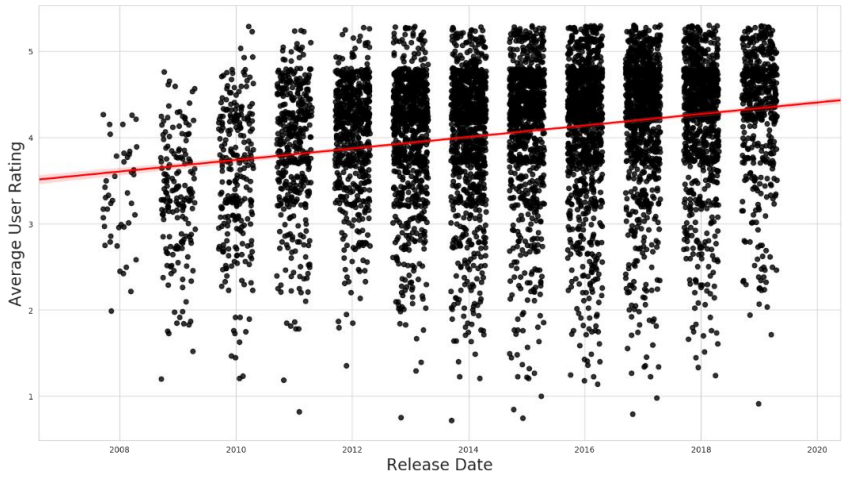
將能數值化的屬性挑出，利用 correlation heatmap，挑出與 Average User Rating 的 correlation 較大的屬性。從此圖可以看出遊戲售價 (Price)、遊戲大小 (Size)、發行時間 (Year)、遊戲分類數量 (NumberOfGenres) 對於 Average User Rating 擁有較高的相關性，因此第二部分模型分析時主要會以這幾個屬性作為訓練特徵。

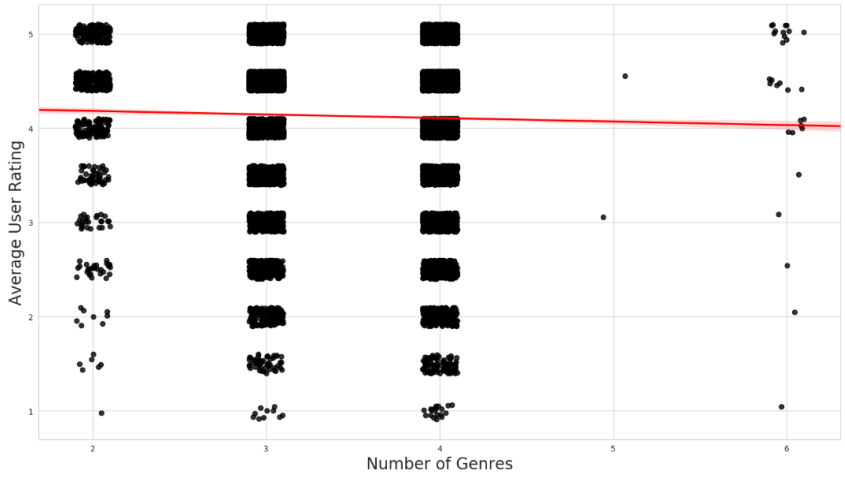


2. Regression:

利用回歸分析可以大致看出屬性如 Size、Year、Genres 的數量，對於 Average User Rating 畫出來的回歸線都具有高擬合度。







My implementation – 1: 我使用 KNN(KNeighborsClassifier)進行預測

KNN 在分類時是以距離為基準，因此很容易受到 Curse of Dimensionality 的影響而得不出好結果，因此在選擇訓練模型時，不能一次性使用太多屬性作為訓練特徵。

Findings – 1: KNN(KNeighborsClassifier)

使用 Cross validation 測試每個 Features 的重要性，可以發現使用的五個特徵重要性很接近，然而訓練出來的模型只能達到 0.2840 的預測正確率。

將屬性減少雖然能減少 Curse of Dimensionality 的影響，但是由於五個屬性都有發揮作用，實際上減少屬性也只能得出相近的正確率，無法提高預測正確率。

Features	Importance
Price	0.377053
Size	0.285408
Year	0.222797
Number of Languages	0.329904
Number of Genres	0.329904

My methodology – 2: Decision Tree

由於決策樹非常適合對於離散類型的資料來進行分類，換句話說，對於連續型的資料，我們必須想辦法將它做適當的區間分割，才能有效的使用決策樹的分析工具。因此，針對我們的預測目標 'User_Rating'，我們發現 User_Rating 的是一個以 0.5 為區間，介於 1~5 之間的數字，所以我選擇以 1.0 為一個區間進行離散化，也就是說 User_Rating < 2.0 的情況我們就把它 label 設為 0，同理，User_Rating >= 2.0 and User_Rating < 3.0 的狀況我們就把 labels 設為 1，以此類推。當然，這邊還有另外一種離散化的方法，也就是以 0.5 為區間指定 label，但我們實測後發現，如果預測目標分割得這麼細的話，會讓整體的準確率大幅下降，而且對於實際應用來說，我們也只需要大概粗分使用者對於該程式的評價，不需要分成 10 幾種不同的評價區間。

接著，'Age_Rating' 的部分相對好處理，因為它只有 4+, 9+, 12+, 17+這 4 種可能的年齡分層，因此我們就依序指定他們的值為 0~3。另外就是 'Genre' 的轉換，由於根據我們的統計和合併的處理後，遊戲的類別只有可能是：Puzzle, Action, Adventure, Role, Family，所以如年齡分類一樣，依序指定為 0~4 的數字即可。

最後，我們認為程式的更新頻率也是個會影響評價的指標，而從我們能獲得的資訊裡面，'Current_Version_Date' 和 'Release_Date' 之間的時間差會和更新頻率呈現反比，所以我們新增一個屬性 'Update_Gap' 來表示上述兩個數值之間的差。

綜合以上所述，我們可以將 Price, Age_Rating, Size, Genre, Update_Gap 的屬性組合成一個 5 維的 feature vector，還有經過離散化的 User_Rating 作為相應的 label，以此來當作我們的訓練資料，來訓練一個能利用 5 種屬性來預測該程式使用者評價的決策樹。

My implementation – 2: 我使用 Decision Tree 進行預測

首先，在 5 維 feature vector 的情況下，我們利用不同的 information gain 當作最佳分割指標得到以下的預測準確率：

Criterion	Accuracy
Gini	46.99%
Entropy	46.36%

從表中可以發現，這個變數對於分類準確育的影響較小，但是 gini index 可以獲得稍微好的結果。

接著，我們試著去調整決策樹分類終止的條件，因為預設的終止條件是分類直到每個 leaf node 都只剩一個 training sample，所以這邊就是調整不同的最小分類樣本數如下：

Min_samples_split	Accuracy
2	46.99%
3	46.57%
4	44.59%
5	46.26%
6	50.62%
7	46.67%
8	46.57%
9	49.48%
10	49.27%

從表中可以發現，較寬鬆的中止條件能獲得比較好的分類準確率，然而差距並不構顯著，而且超過 10 之後會呈現飽和的狀態，所以這個變數可以說有可調整的空間，但不足以影響全局。

再來，我們試著分析是否有些 feature 對於預測準確率相對較不重要，所以我們做了 2 種實驗，分別捨棄 Genre 和 Age_Rating 的 attribute，藉此將 feature vector 減少成 4 維，並且重複上述的分析，看看是否有甚麼變化，結果如下：

Feature vector = [Price, Age_Rating, Size, Update_Gap]

Criterion	Accuracy

Gini	44.39%
Entropy	44.39%

Min_samples_split	Accuracy
2	44.39%
3	45.22%
4	44.39%
5	46.78%
6	43.97%
7	45.32%
8	47.82%
9	45.63%
10	50.52%

Feature vector = [Price, Size, Genre, Update_Gap]

Criterion	Accuracy
Gini	45.63%
Entropy	44.80%

Min_samples_split	Accuracy
2	45.63%
3	48.02%
4	45.74%
5	44.39%
6	46.88%
7	45.74%
8	50.73%
9	47.51%
10	49.06%

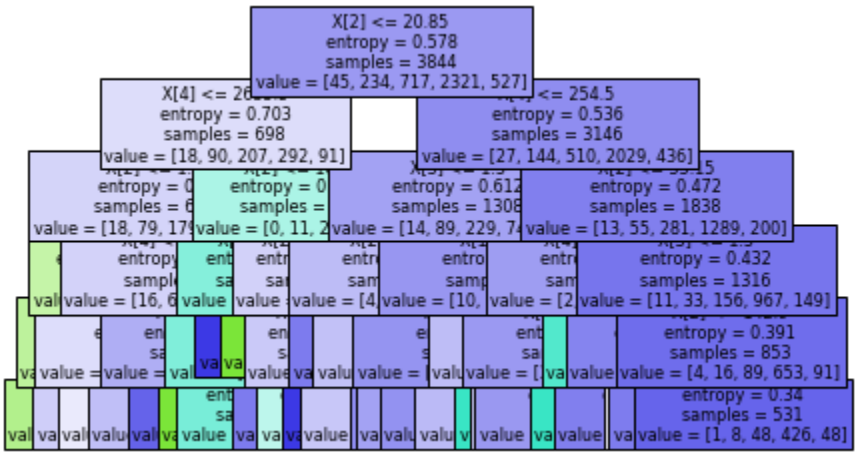
從上面的結果可以發現，在同樣的最小分類樣本數下，還是 5 維的 feature vector 能獲得稍佳的準確率，因此，減少 feature vector 的維度似乎不是一個增加預測準確率的好方式。

最後，我們發現調整決策樹的深度是一個足以改變全局的變數，也就是說，原本決策樹預設的分類方式會不斷地增加決策樹的深度直到每個 leaf node 符合最小分類樣本數的條件，而這個做法會導致建構出來的決策樹 overfit to training data，而解決辦法就是適當的控制決策樹的深度，所以我們就作了以下的實驗(5 維 feature vector)：

Max_depth	Accuracy
2	59.77%
3	59.36%

4	59.77%
5	62.27%
6	58.52%
7	57.38%
8	57.9%
9	55.20%
10	57.17%

從表中可以發現準確率顯著的提升，並且將決策樹深度限制在 5，能獲得高達62.27%的分類準確率。除了觀察準確率以外，我們也視覺化決策樹分類的過程如下(X[i], i ∈ [1,5])：



其中 X 就代表我們的 5 維 feature vector。

Findings – 2: Decision Tree

由於分類的過程條件複雜，我們試著觀察那些指標在分類的過程出現最多次，或是在決策樹的頂端，結果是 Size 使用了 10 次，Update_Gap 使用了 9 次，而且可以發現決策樹的前幾次分類依據標準也都是 Size 或是 Update_Gap。

因此，針對我們的實驗觀察，我們認為以決策樹的分類來說，應用程式的大小以及更新頻率對於其使用者評價的影響是相對比較高的。

My methodology – 3: Random Forests

在進行完 data 的預處理後，我們認為，評價越高的，此款 App 就越成功，

因此我們開始著手進行，相對於評價分數重要的 attributes 選取，去了解影響一個產品評價高低的因素。

我們的預測模型:

Y 軸的 attribute: “Average User Rating”

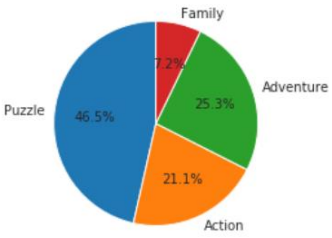
X 軸的 attributes: “剩餘的 attributes 進行選取”

X 軸的 attribute 選取方法:

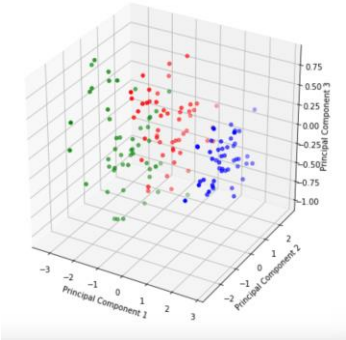
1. 我使用 Association rule，去觀察每一個 attributes(columns)對於我們所感興趣的 Y 軸 output: “Average User Rating”(使用者評價)的“Support”和“Confience”，來當作標準去選取我們要進行分析的 attributes。
   Ex: (row 上的數字代表不同的 attributes)

	precision	recall	f1-score	support
1	0.00	0.00	0.00	0
2	0.00	0.50	0.01	2
3	0.03	0.40	0.05	57
4	0.96	0.64	0.77	4688

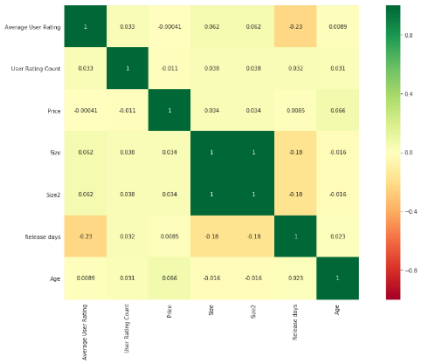
2. 我使用 Classification，去將不同類別的 Strategy Games 進行區分，找出評價較為高的一群，在分析時，以此給予較高的權重，來進行預測模型處理。
   Ex:



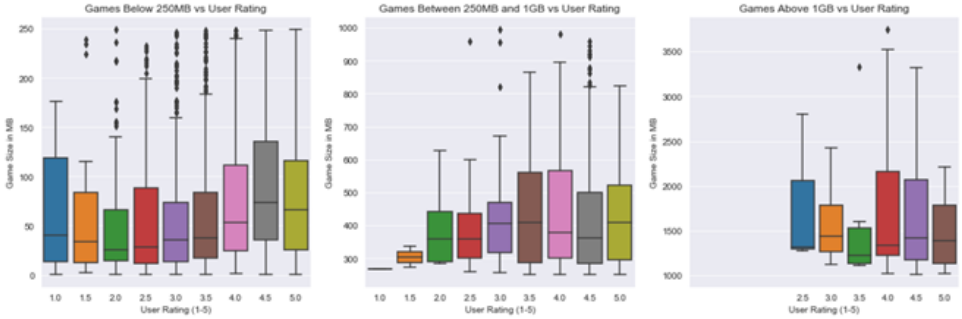
3. 我使用 SVD，來去觀察扣除”Average User Rating”後，剩下的 attributes 中哪幾個所佔的權重較大，拿這幾個 attributes 進行分析。
   Ex:



4. 我是用 Correlation matrix，去比較不同的 attributes 之間的關聯性，關聯性高的，拿來進行處理。Ex:



5. 使用四分衛圖進行分析和處理，找尋不同的 attributes 對於“Average User rating 的關係”。 Ex: size v.s. average user rating



6. 使用三維圖式，同時比較兩種不同的 attributes 對於“Average User rating”的關係。 Ex:

My implementation – 3: 我使用 Random Forests 進行預測

根據上方的 methodology 所選取出的 attributes 進行資料分析，我所實踐的方法為:

Step1: 將總共 17k 筆的資料，切割成 0.7*17k 為訓練資料，0.3*17k 為測試資料

，並且區分成 X 軸和 Y 軸的 attributes。

Step2: 將 Training 的 data 放入 **RandomForestClassifier** 中，開始進行訓練。

Code:

```
#n_estimators代表要使用多少CART樹（CART樹為使用GINI算法的決策樹）
rfc = RandomForestClassifier(n_estimators=100,n_jobs = -1,
                             random_state =50, min_samples_leaf = 10,
                             max_depth=13)
```

Step3: 從訓練資料中建立模型。

Code:

```
#從訓練組資料中建立隨機森林模型
rfc.fit(X_train_New, y_train_New)

#預測測試組的是否發生
rfc_pred_new = rfc.predict(X_test_New)
```

Step4: 將我模型進行預測，準確率為 0.57。

Code:

```
rfc.score(X_test_New,y_test_New)
```

Step5: 我將模型進行 Cross Validation，準確率提升為 0.62。

Code:

```
rfc.score(X_test_New,y_test_New)
```

```
0.6212032137958063
```

Findings – 3: Random Forests

在上方的預測結果中，使用了四個 attributes，分別是: Price, Size, In app purchase, Genres，我在 methodology 中分析了六種不同方法，因而找這齣四個重要的 attributes，來去進行分析。

➔ 下方是四種不同的 attributes 對於預測模型的重要性:
Price: 0.22360035
Size: 0.39928416
In app Purchase: 0.06974065
Genres: 0.30737484

可以發現遊戲的 Size 對於遊戲評價高低，佔有很重的影響，可能是因為遊戲 Size 較高的遊戲，遊戲的操作性或者畫面的精緻度，以及遊戲內容的豐富度較高，更能引起使用者的興趣。

遊戲的 Genres 對於遊戲的評價高低也有影響，因為在這個分類裡面，大多人喜歡游玩偏向思考類的遊戲，使得其評價會比其他種的 Genres 較為弱勢。

至於遊戲的價格和在遊戲裡所購買的產品價格，對於遊戲評價與否，產生了重要的影響，可以得出，在使用者心中，付費遊戲的評價可能會比免費遊戲好，也就是付費遊戲會比免費遊戲來得有趣耐玩，因為既然付費所以品質會較高。

Contribution:

我們處理並且分析應用的**不同屬性**對於**使用者評價**的影響，也建立三種不同的預測模型。

(1) 使用 **KNN 模型**來預測使用者的評價狀況，根據我們所選取的 attributes 進行預測，可以發現預測的準確度不高，可能的原因是 KNN 所找出的分類模型變成 overfitting 又或者是 K 的選擇不夠好。

(2) 使用**決策樹模型**來預測使用者的評價狀況，並利用決策樹的分類過程，讓我們分析出對於使用者評價潛在有較大影響的屬性，藉此來提高預測模型的準確性。

(3) 使用 **Random Forests 模型**來預測使用者的評價狀況，藉由我們所選取的 attributes 進行預測，並且在不同的 Decision Tree，找出一個最佳解，但是也很依賴我們所選取的參數，例如:子樹的深度。

我們在此期末專題中，使用了三種不同的預測模型，去對我們所選取的 attributes 進行對於**”Average User Rating“**，也就是**使用者評價高低**進行預測分析，藉由這分析的過程，我們分別使用了 **KNN** 和**決策樹**和 **Random Forests** 的模型對我們的 data 進行處理和預測。

我們所一開始選取的 Attributes 總共有六種，分別是: Price, Age_Rating, Size, Genre, Update_Gap, In app Purchase。在這六種的 attributes 中，根據我們所使用區分種重要性 attributes 的方法，EX: Association rule，分別在不同模型中，放入不同數量的 attributes 進行預測，最後根據三種不同模型所統整出的結果，可以發現，影響評價高低，也就是**”Average User Rating“**的最重要因素有: **Size, Genres, Price**。

我們所設計出的模型，可以讓開發商在設計遊戲前，根據這三個影響遊戲受歡迎程度的預測模型，著重開發遊戲的某些部份。例如在 Size 部分，開發商可以著重於提高遊戲品質和體驗，使得遊戲更加受歡迎;又或者是在 Genres 部分，開發商可以偏向於某一部份的設計，讓使用者的黏著度提高，以增加遊戲的討論度，藉此提高遊戲評價;亦或是在 Price 部分，許多高消費或者對於遊戲有期待者，會偏向找尋付費遊戲，如果可以得到這方面客群青睞，更能得到高度評價。

Conclusion:

在一個 Dataset 處理的過程中，**最重要的事情是資料的前處理**，在這次的期末專題中，我們學習到對於遺失資料和不同型態資料的處理，往往是花費最多時間和精力的部分。接著，不能夠直接將整個 Dataset 丟入我們的預測模型中，因為這樣所分析出來的結果，可能會浪費時間，或者考慮太多不必要的 attributes，所以**需要先對整筆 dataset 的 attributes 進行重要性分析**，才能在後續處理時，**針對相對於其他 attributes 相對重要的 attributes 進行預測處理**。

使用我們選取出的重要性較高的 attributes 進行預測分析，才能使分析起來更有意義，而不是隨便去訓練出一個預測模型，考慮太多不重義的因素，反而會使預測的效果失去真實性，只需要考慮 Critical Attributes，更能看出此預測模型，是否符合我們的期待，夠不夠準確。