NCKU Data Science 2018

Competition 2

Rating Prediction with User Business Review

Group2

醫工 107　F94036089　吳旻昇

藥學 109　I84031111　何明洋

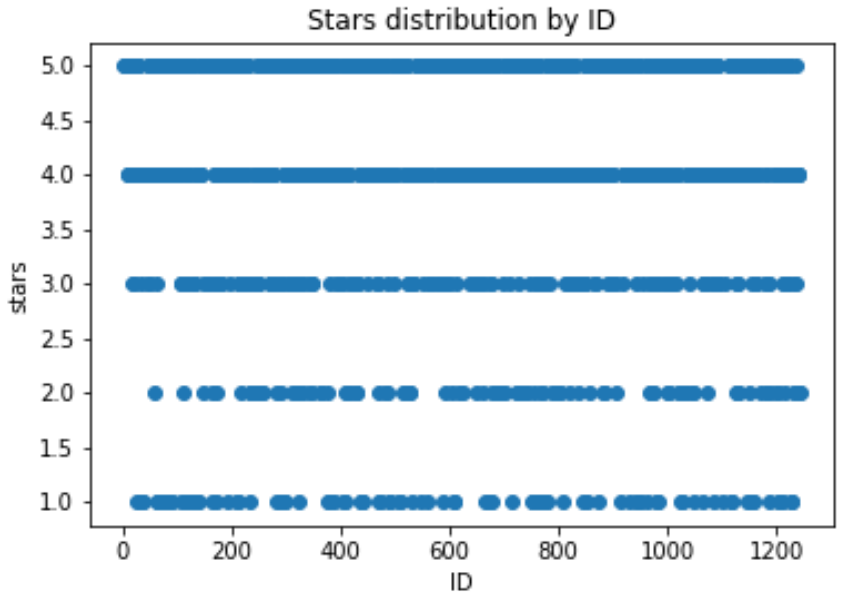
Description

STEP I: 整體資料檢視

在進行資料處理前，我們先檢視了整體資料的一些特性，我們檢視了 train data 中各評分(stars)的個數，如下表：

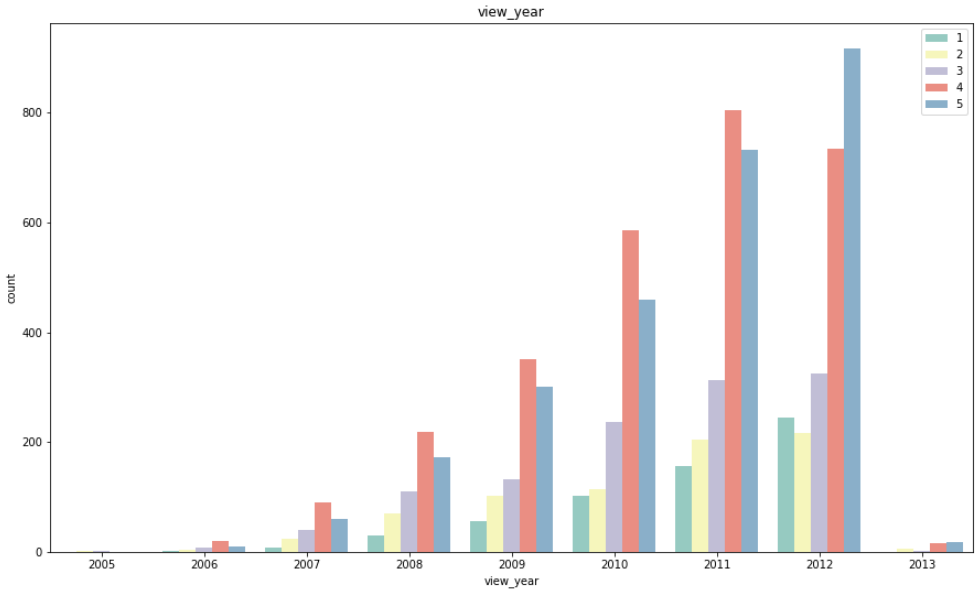
stars	count	Ratio(%)
5	2669	33.38
4	2820	35.26
3	1168	14.61
2	741	9.27
1	599	7.49

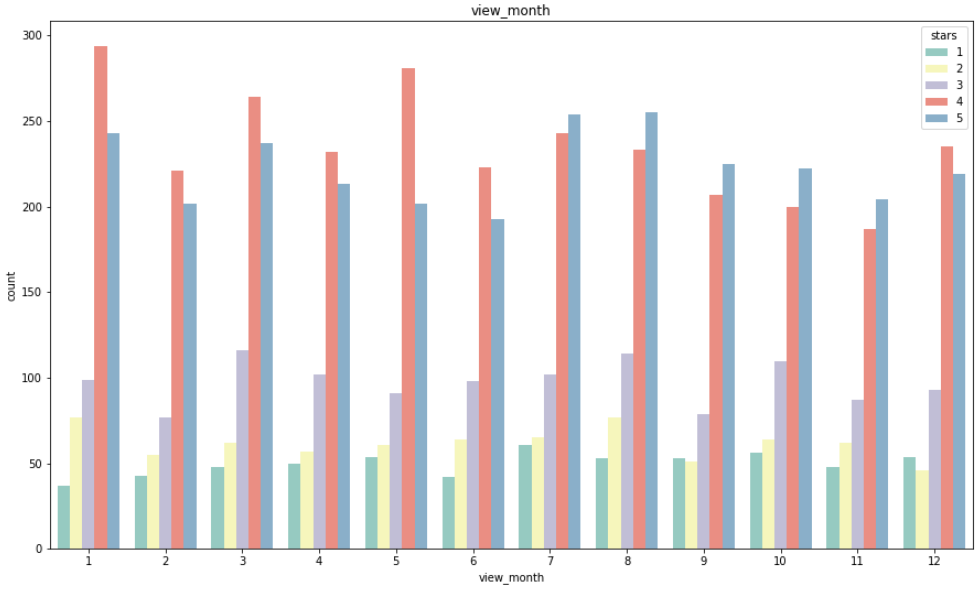
從中可以得知，評分 4 為最多，佔了整體資料的 35.26%，而平均的評價則為 3.78，此外我們也觀察了各評分是否在不同 ID 上有分布上的特異，如 ID 在前面者評分會較高，ID 在後面者評分會較低等(如**圖 1**)，然而根據圖中的結果而言似乎是沒有這樣分布特異性的，各評價在不同的 ID 上分布的其實很平均。

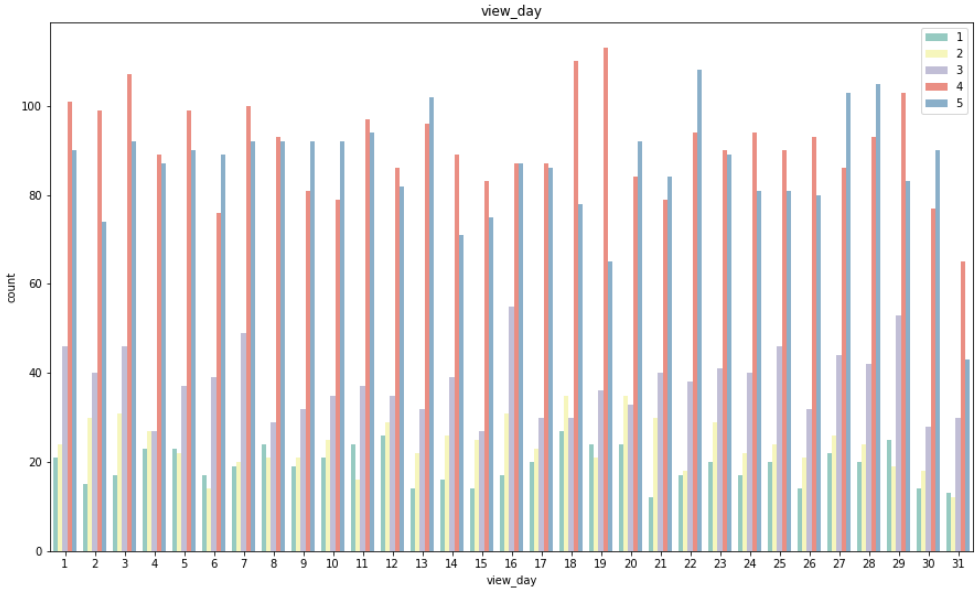


▲圖 1

而原始資料中有含評分的日期，因此我們將日期拆成年、月、日，分別去看是否在不同的年月日下評分會有相關的特異性(如**圖 2**)，然而根據圖中的分布是沒有這樣特異性的，因此在之後建立 model 時，我們並沒有將這些參數考慮進去。


view_year
1
2
3
4
5
count
0
200
400
600
800
2005
2006
2007
2008
2009
2010
2011
2012
2013
view_year


view_month
stars
1
2
3
4
5
count
0
50
100
150
200
250
300
1
2
3
4
5
6
7
8
9
10
11
12
view_month


view_day
1
2
3
4
5
count
0
20
40
60
80
100
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
view_day

▲圖 2

STEP 2: 文章清理（Text Cleaning）

為了有效萃取文章中的詞彙特徵，文章清理的前處理步驟非常重要。我們曾以未處理的文字下去建立模型，但觀察以 Tf-idf 跑出的幾萬筆文字中有許多是沒有意義的，推測是大家在評價時常會以非正規的文字來強調語氣，因此為了提升預測準確，我們進行了文章的清理，以下將介紹我們在這次比賽中使用的方法。

首先，我們利用正規表示式（Regular Expression），去除標點符號與數字等非字母字元、還原縮寫（如：助詞、be 動詞、not、複數縮寫）等。

再者，使用自然語言工具包（Natural Language Toolkit, NLTK）做詞性還原（Lemmatize），將動詞的時態和語態、形容詞與副詞的比較級、名詞的單複數等全部轉成原型的詞元（Lemma）。

最後，我們把網路上找到的停頓詞（Stop Words）結合 sklearn 中內建的停頓詞，拿來過濾每一則評論常出現但不具有特徵意義的詞彙。

在此我們舉 training data 中的第 5 則評論，其原始資料語經過我們清理後的資料如下(**圖 3、4**)。

```
We have tried this spot a few times and each visit we find just enough we like to keep coming back.  The whitefish on bialy is a
favorite and the selection of bagels is extensive.  Try an oversized sandwich with a vanilla phosphate... All wonderful.  The on
ly downside is that most items are $10+ and that adds up for a family of 5.
```

▲圖 3

```
try spot time visit like come whitefish bialy favorite selection bagel extensive try oversized sandwich vanilla phosphate wonder
ful downside item add family
```

▲圖 4

因圖片稍模糊，以下另外打出來檢視並標記原始資料與清理後資料之差別：

原始資料

We have tried this spot a few times and each visit we find just enough we like to keep coming back. The whitefish on bialy is a favorite and the selection of bagels is extensive. Try an oversized sandwich with a vanilla phosphate… All wonderful. The only downside is that most items are $10+ and that adds up for a family 5.

清理後資料

try spot time visit like come whitefish bialy favorite selection bagel extensive try oversized sandwich vanilla phosphate wonderful downside item add family

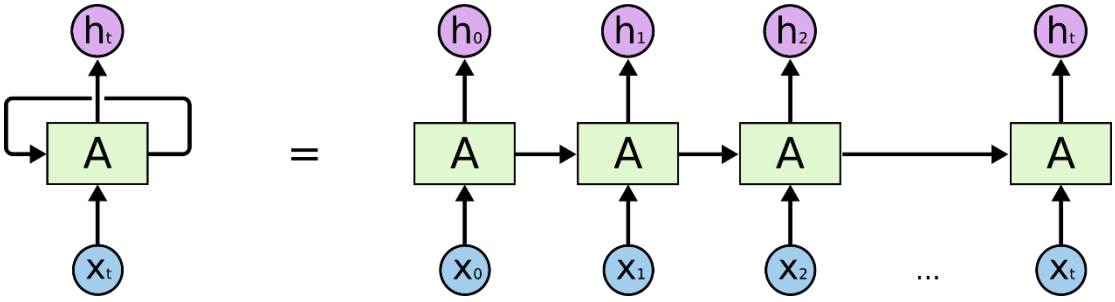
在原始資料中有使用顏色標記的字都是最後留下來的字，其中標記**藍色**的字表示未更動，標記**紅色**表示經過時態轉換，標記**綠色**則表示單複數的變化，亦可

以發現所有的大寫也都變成小寫、符號都完全清除了。因此經過文章清理後，我們已經將原本冗贅詞彙過多且因單複數時態等複雜化的文字轉變成重點式之原形呈現的句子，以利接下來有效的分析。

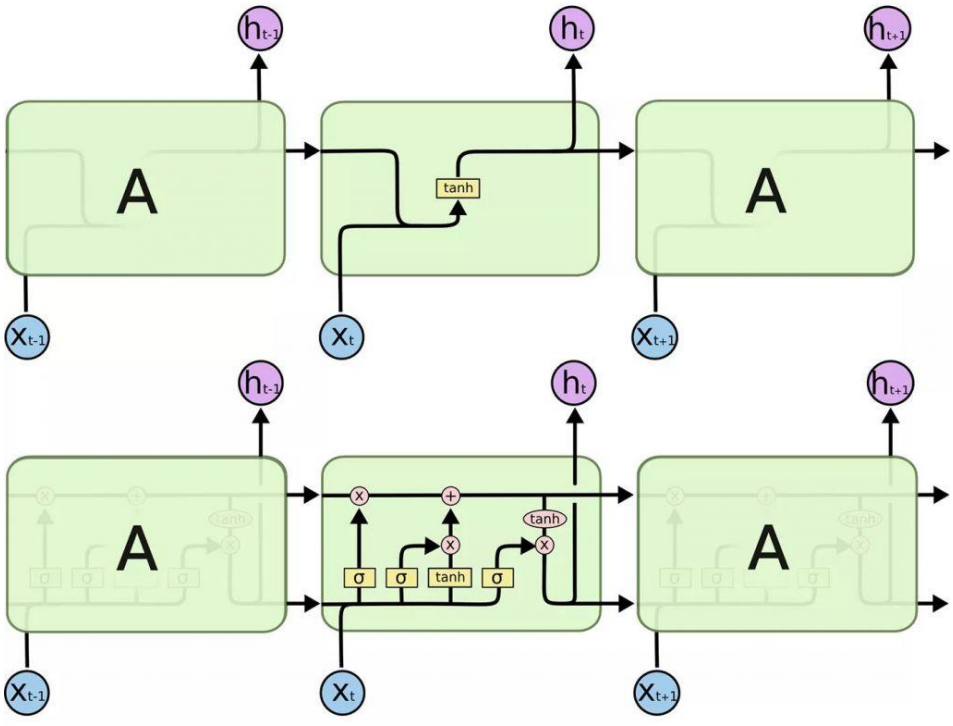
STEP 3: 長短期記憶模型（LSTM Model）

現今的自然語言處理大多運用深度學習中的遞歸神經網路（RNN）以及改良後的長短期記憶模型（LSTM），因為這些模型具備記憶能力，能夠記住詞彙的順序，也就具備理解語句的能力。

RNN 透過將前一時刻的輸出加入到當前輸入中的方式，學習時間序列資料的關係，而 LSTM 則導入 Forget Gate、Input Gate、Output Gate 來強化記憶能力。



▲RNN 示意圖



▲RNN（上）與 LSTM（下）的內部結構

我們一開始認為採用此方式，較能有效學習文章的語句特徵，於是使用過 LSTM、Bidirectional LSTM、GRU，且嘗試過加入 1D Convolution、Pooling、Dropout 等神經層，但是都產生嚴重的 overfitting，後來實際查看評論資料才發現，其實每則評論的字數都不多，可能不適用這種複雜的神經網路模型架構。

Step 4 : Tf-idf

在分析語句中詞彙除了要考量其在同個句子出現的次數外，在所有 data 中的句子中有幾句中有出現也是重要的考量點，基於這樣的想法，我們於是使用 sklearn 中的 TfidfVectorizer 對這些評論語句進行剖析，轉變成向量。而我們在後期也考慮到如果出現次數過低如只有在某評論中出現的字在建立預測模型上沒有太大意義，因此在 TfidfVectorizer 參數調整中使用了 min_df=2，將那些出現在小於兩次評論的單字清除掉。

Step 5 : Xgboost Regressor Model

最終，我們採用 Xgboost Regressor Model，把原先的分類問題改為回歸問題，把整組資料下去做訓練，並以 MSE 作為 Loss Function，才使得成效提升。

在測試部分，我們從所有 data 中分出 20%下去觀察 Accuracy、RMSE，並經由不斷的教調，最佳結果 Xgboost Regressor 的參數設置為 max_depth=4、n_estimators=300、learning_rate=0.05 等。

Step 6 : Final result

經過以上處理後我們將結果上傳到比賽網站中，得到的最佳成績是：

RMSE=0.9016、Accuracy=0.4478

Analysis

A. 在這次比賽中我們能提高 accuracy 原因：

1. 先檢視 training data 之 stars 分布

經由圖示化的檢視，我們得知其實 stars 在不同 ID、年、月、日等分布上其實是相當平均的，因此並沒有必要將這些 feature 考慮進來。而經過時測候，也確實發現未採用這些參數的 RMSE 會較低。

2. 對於句子先進行了文章清理

因原先評論的句子中含有許多無意義的單字，又或是因為進行了時態變化導致同一個單字被區分成不同單字，以上若丟進 Tf-idf 中則可能會產生過多不必要的詞彙，除了因為可能只出現一次而導致 model 有所偏差，其它因為時態單複數變化而變成不同字的單字則會失去其原先的權重。實測上我們在進行文章清理後才有效將 RMSE 降低到 0.95 內。

3. 使用 LSTM Model

由於我們有考量到語句前後文必定會影響整個句子的意思，例如一個否定詞加上某個動詞後導致整個句子變成否定狀態，這並非單純分析一個否定詞彙能夠解決的問題。因此我們採用 LSTM Model 來處理語句前後文，但實測上結果搭配 LinearSVC 竟然比未處理的資料跑出來的 RMSE 還要高，這恐怕是因為原先語句中辭彙真的太亂加上樣本過少才導致的非正常結果。然而我們最後使用 LSTM Model 加上 Xgboost Regressor 的方式才有效將 RMSE 降低到 0.91，因此整體而言 LSTM 處理過的資料應該是有其效用與意義的。

4. 使用 Tf-idf、考慮使用 word2vec

以邏輯上思考比起 bag of words 的方式，採用 Tf-idf 應較能夠對每個字彙產生出有意義的向量，但我們並未對 bag of words 的方式進行實測是否會較使用 Tf-idf 效果還要差。

另外我們亦有考慮使用 word2vec 的方式，以此種方式產生結果因又考慮單字與單字間相近性可能又會比 Tf-idf 更有邏輯上的意義，然而礙於原本樣本過少，以過少樣本產生的 word2vec model 可能會失去其對每個單字衡量之意義，雖網路上 google 有提供已經建立好的資料庫模型能使用，但考慮到時間等因素我們並未去測試 word2vec 是否會讓 RMSE 降到更低。

5. 使用 Xgboost

Xgboost是在GBDT(Gradient Boosting Decision Tree)基礎上改進boosting演算法，其強勢之處在於形成一棵決策樹後，下一棵樹是從上一次的預測基礎中取最優的一步進行分裂/建樹，最後加總所有決策樹的預測結果得到答案，在這樣最優化的狀況下預測精準度較其他演算法高，也就成為競賽中之翹楚。而這也是最後讓我們 RMSE 能夠突破 0.95 大關之最佳利器

B. 其他我們測試過的方式：

1. 只有去除 Stop words 的資料

我們曾經試過只去除 Stop words 的 data 下去做預測，然而這樣的方式反而比未去除 stop words 跑出來的結果的還要糟糕，我們認為是因為原始資料使用的字彙真的太過於混雜加上樣本數過小，才導致的異常結果。

2. 其他使用過的 model

我們也使用過 GaussianNB、LogisticRegression、LinearSVC、SVC 等模型下去測試，在未使用 Xgboost 我們實測下預測率最高的是 LinearSVC，而次好的為 LogisticRegression，LinearSVC 是讓我們 RMSE 終於降至 1.0 以下的 model，但如果單純使用 LinearSVC 加上只提取評論搭配 Tf-idf 則 RMSE 會比 LinearSVC 搭配 LSTM 與 Tf-idf 還低，其原因應該同上，原始資料過於混雜。但是使用 LinearSVC 跑出的 Accuracy 都能夠維持在 50%以上，反倒是使用 Xgboost 雖然 RMSE 降低了，但 Accuracy 也跟著降低了

C. 整體分析

雖然在這次比賽中我們使用了很多以邏輯思考上應該有助於提升 Accuracy 及降低 RMSE 的資料前處理方法，但幾乎都無法有效的達到目標，推測後恐怕是因為原始資料評論者使用的字彙很混亂，加上資料量太小等因素導致。而也推測可能是不同評分者標準太不一，雖然有些人使用較為不錯的文字評論，然而卻給予較低的 star，導致產生出的 model 一直無法準確預測，讓 Accuracy 非常的低。

Github

https://github.com/Kaminyou/NCKU_DATA-SCIENCE_CP2.git

References

1.本次我們使用 xgboost 演算法作為模型訓練使用，其所用的方式是決策樹(回歸樹)，整體安裝與使用 Xgboost document 參考以及網路上其他人使用的資料

http://xgboost.readthedocs.io/en/latest/

2.本次我們也使用了 Keras 作為深度學習上使用，參考了 github 中其他人的做法以及原始 document

https://keras.io/

3.本次我們亦使用了 nltk 套件，作為文字處理之使用，參考了 github 中其他人的做法以及原始 document

https://www.nltk.org/