

Report - DataLabCup1: Predicting News Popularity

Team123 鄭程哲 彭嘉洛 楊晴雯 李浩榮

1. 如何選擇 Model ?

我們認為此次 competition 的重點在於 feature engineering, 如果能妥善處理及分析 page content 裡的內容, 選出適當的 feature 來 train model, 對整體的 performance 的進步會最明顯, 因此對 model 的嘗試較少, 分別實驗了 lecture 學過的 SVM 及 RandomForest, 還有 Kaggle 上非常熱門的 XGBoost 作為我們主要的三種 model。

- SVM, 選擇 SVC 的原因是因為他可以解決高維特徵的分類問題, 不過實驗證明結果是最糟的。
- RandomForest, 是使用 Bagging 加上隨機特徵採樣的方法所產生出來的整體學習演算法, 會選此 model 的原因是因為認為 RandomForest 結果比較不容易過度擬合, 實驗證明後也是三種 model 中 score 最高的。(test AUC: 0.5867)
- XGBoost 全名為 eXtreme Gradient Boosting(極限梯度提升), 屬於 boosting tree 的工具包, 是在Kaggle大大小小競賽中都很常見的強大 model , 其優點除了精準度很高外 , Training 的速度也非常快。我們此次實驗的是 XGBClassifier (test AUC: 0.5804)

Model	Validation score
SVM	0.572
RandomForest	0.584
XGBoost	0.589

雖然在 Validation score 的部分, XGBoost 的表現是最佳的(0.589), 但在 testing score (0.5804) 的部分卻意外地輸給 RandomForest (0.5867), 我們認為可能的原因為:

- XGBoost 在目標函數添加了標準化。因為模型在訓練時為了 fit training data, 會產生很多高次項的函數, 但反而容易被 noise 干擾導致 overfitting 。
- RF 則是使用隨機數據樣本獨立訓練每棵 tree。這種隨機性有助於使模型比單個決策樹更健壯, 相較於 XGBoost , 發生 overfitting 的機率也較低。

因此我們最後決定使用 **RandomForest** 作為我們主要訓練的 model。

2. Feature selection 如何選擇？

結合了文字與數字的資訊來做為 feature，文字的部分包含 {title、tag}，數字的部分包含 {channel、fig_count、link_count、weekday、month、word_count、emo_val...等}。

- 文章相關資訊：

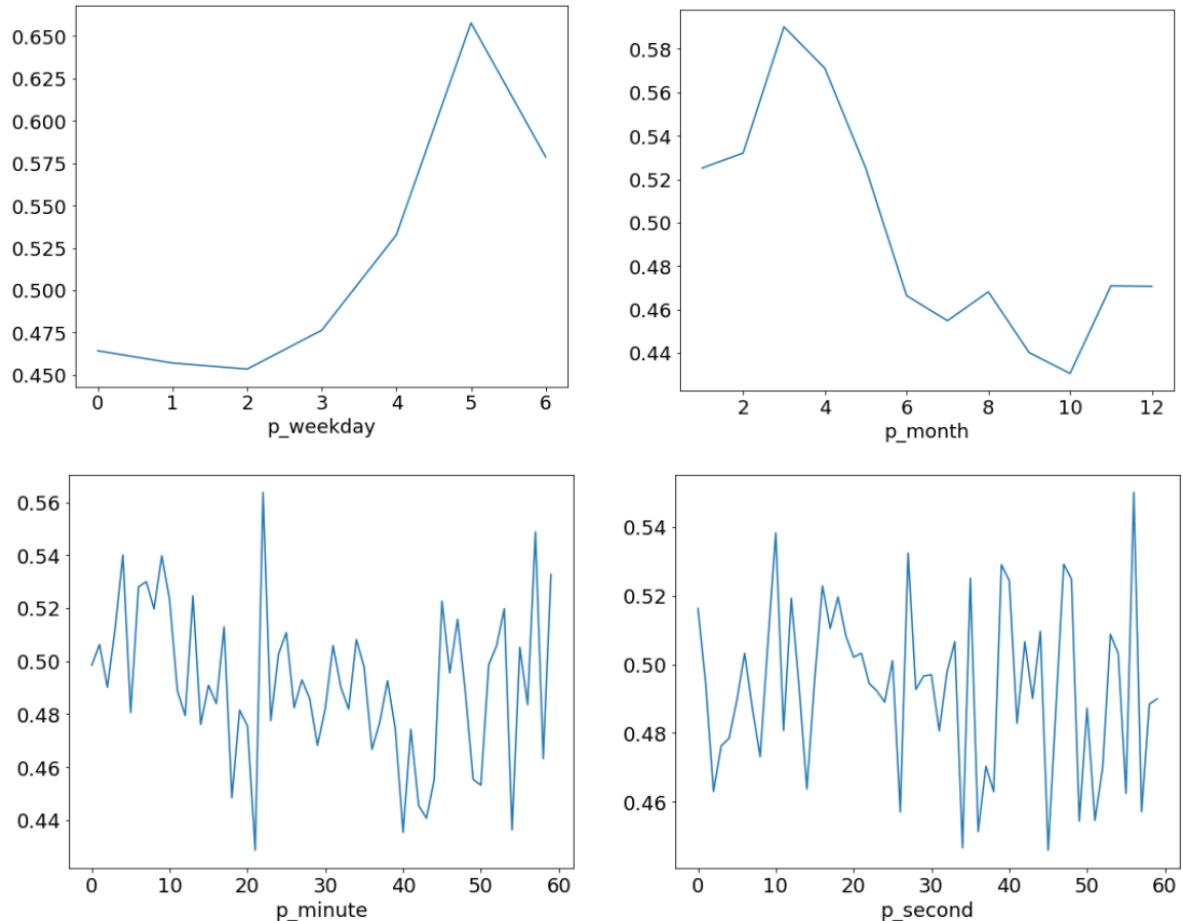
1. title
 - 透過 HashingVectorizer 轉為vector
2. content
 - 透過 HashingVectorizer 轉為vector
3. tag
 - 透過 HashingVectorizer 轉為vector
4. channel
 - 透過 Label Encoding 轉為數值
5. word_count
 - 單字數量
6. unique_word_count
 - 文章中出現幾種不一樣的字
7. sentence_count
 - 語句數量
8. words_in_quotes_count
 - 在引號中的單字數量
9. avg_sentlength
 - 文章平均的語句長度
10. htags_count
 - hashtag數量
11. fig_count
 - 圖片數量
12. link_count
 - 連結數量
13. emo_val
 - 判斷文字的情緒高低程度
 - EX: {'Angry': 0.12, 'Fear': 0.42, 'Happy': 0.04, 'Sad': 0.33, 'Surprise': 0.08}
 - 將各情緒分數平方後相加成emo_val, 越高代表文字更具某種情緒

在文章相關資訊的特徵選擇上，我們的model一開始只選擇較好量化的feature(4~12)，而忽略了文字帶來的資訊，即便這樣的model在testing AUC分數就有來到0.56左右，但不斷嘗試這些特徵之間的組合帶來的效益都不太高，於是我們加入了另一個量化數值，情緒分數，我們認為越是帶有情緒的標題與內容(如：聳動標題)，越能吸引讀者點閱，意味著熱門程度可能越高，我們將title, content分別計算情緒分數並丟入模型中處理，最後得出只使用標題的emotion value在testing的表現中是最好的(0.56→0.57)。

接下來我們嘗試多種特徵組合都沒辦法有大幅度的進步，於是我們回到根本，使用了文字本身的資訊，文章的熱門與否應該和文字有最直接的相關性，我們將title, content, tag分別丟入 TfidfVectorizer, HashingVectorizer計算單字與文章的頻率，最終實驗得出，將title與tag透過 HashingVectorizer降到8維，並與原本的數值特徵合併，這樣的input能夠在testing上達到最好的表現(0.576→0.586)。

- 時間資訊:

- weekday
 - $\text{weekday} = 0 \text{ if } \text{time.weekday} > 4 \text{ else } 1$
- month
 - $\text{month} = 1 \text{ if } \text{time.month} > 5 \text{ else } 0$
- timestamp
 - $\text{timestamp} = \text{time.timestamp}()$



根據上圖所示，相較於第二列的 minute 與 second 有過多的雜訊，第一列的 weekday 與 month 有較明顯的趨勢，{以 weekday 為例，五六日人們較有時間閱讀新聞，因此新聞 popular 的機率也會比較高}。

- 分類相關資訊:

- tag
- channel
 - is_lifestyle
 - is_entertainment
 - is_bus
 - is_socmed
 - is_tech
 - is_world

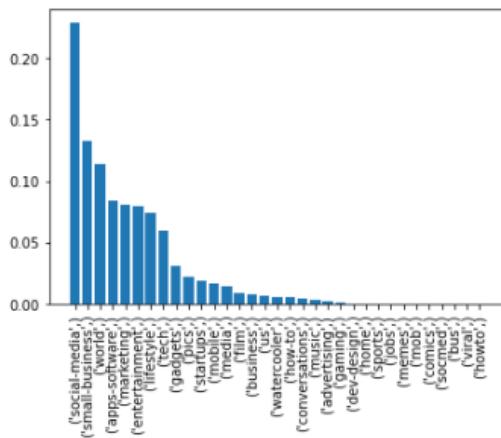
- ❖ 最後選擇的 feature = ['text', 'tag', 'channel', 'fig_count', 'link_count', 'weekday', 'month', 'timestamp', 'sentence_count', 'words_in_quotes_count', 'emo_val', 'is_lifestyle', 'is_entertainment', 'is_bus', 'is_socmed', 'is_tech', 'is_world']

3. 遇到的問題 or 發現

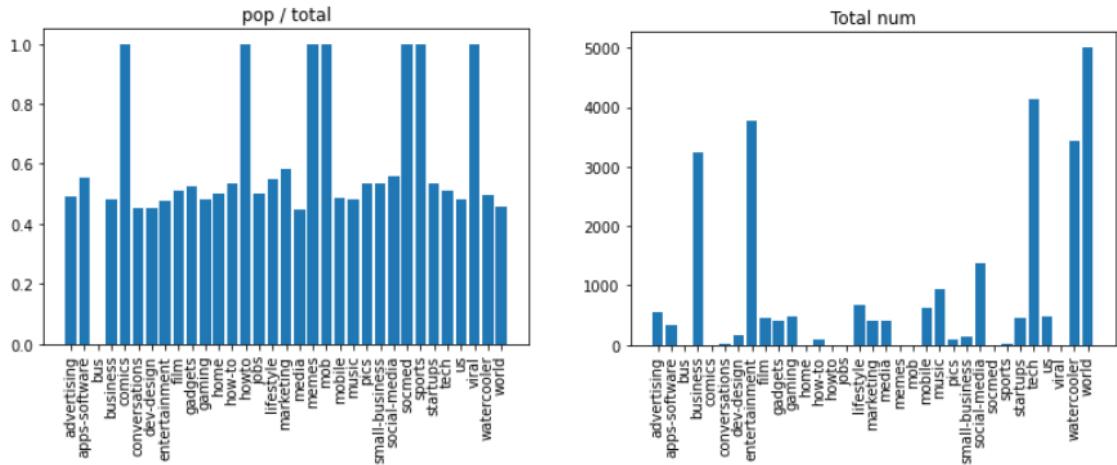
在文章相關資訊中，總共用了種feature(上一頁中的4.~10.)進行不同的組合，看怎麼樣才會有最好的auc。一開始單獨採用word_count就有明顯的進步，但再加上unique_word_count之後反而掉下來一點點，讓我們發現**增加越多feature不一定會有更好的表現**。也發現有些**feature其實是沒有幫助的**，像是htags_count，因為文章中不太會出現hashtag，所以對於分類基本上是沒有幫助；還有avg_sentlength，其實就是把word_count去除以sentence_count，是在原有的feature上再去做線性的轉換，所以對於分類一樣是幫助不大。

下圖是我單獨將 33 種 channel 作為 feature，利用 XGBoost 的 feature importance 所畫出來的圖，可以觀察到以此結果來說 "social_media"、"small_business"、"world"為前三大重要的 feature，但當我將這三個 feature 放入原本預測 news popularity 的 model，得到的 performance 反而是更糟的。

```
[0.00237224 0.0835878 0. 0.00764743 0. 0.00446142
 0.00053075 0.07926163 0.00942628 0.03078168 0.00112224 0.
 0.0050902 0. 0. 0.07462882 0.08019226 0.01437287
 0. 0. 0.01631064 0.00352136 0.02212426 0.13256465
 0.22884166 0. 0. 0.01835994 0.05946128 0.00631874
 0. 0.00555936 0.11346247]
```



左下圖是將 channel 個別的 pop / total 的結果，希望可以從此表觀察到哪個 channel 的 popularity 比例較高，從下圖可觀察到特別突出的分別是 commics、howto、memes、mob、scomed、sports、virus。右下圖則是每個 channel 的新聞總數，可以發現pop / total 較高的 channel，其實都是那些總數非常少的例如:{sports: 只有6筆，但六筆都是popularity}，因此對加入channel的合理性感到懷疑，不過經過實驗證明加入適當的 feature 對 performance 還是有正向的影響。



bus	comics	howto	sports
total: 1	total: 1	total: 1	total: 6
popularity: 0	popularity: 1	popularity: 1	popularity: 6
unpopularity: 1	unpopularity: 0	unpopularity: 0	unpopularity: 0
non / tital: 0.0	non / tital: 1.0	non / tital: 1.0	non / tital: 1.0
socmed	mob	memes	viral
total: 2	total: 2	total: 2	total: 1
popularity: 2	popularity: 2	popularity: 2	popularity: 1
unpopularity: 0	unpopularity: 0	unpopularity: 0	unpopularity: 0
non / tital: 1.0			

4. 其他重要的內容

在做feature engineering時, **domain knowledge**就顯得很重要。像我們做的數字處理方面, 只是單純去計算各方面的count總數, 並沒有深入到文字本身。在文字處理方面, 我們把title跟content轉成vector, 但也沒有仔細去研究怎麼樣的詞彙或語句會產生高的popularity。也許今天由一個記者或是新聞編輯來發想feature的種類, 他們能更洞見熱門文章中的特殊之處, 或是提出冷門文章無趣的說明或過於情緒化的字眼。

所以經由這次competition, 我們了解到光是有好的model、好的hyperparameter、甚至是大量的data都沒用, 如果data沒有經過**良好的preprocessing**, 能達到的accuracy一定有限。AI領域不只是大家熟悉的model、algorithm重要, data analysis更是幫助training順利的第一步驟。