

Rating prediction with user business review

一、資料初探：

拿到 **training_data.csv**，我們做的第一件事，是在未對資料做任何進一步處理的情況下，嘗試使用不同的分類機，大致離不開 **Ensemble** 與 **SVC**。對參數做不同的調整，用 **score** 的高低，初步決定使用 **LinearSVC** 作為我們的分類機。

二、資料前處理：

在資料的前處理上，我們首先將同時出現在 3 星以下(含)及三星以上的詞彙抓出來，我們希望可以視為雜訊，但卻又發現數量極多，達 18527 筆，且若有只在一部份出現一次、卻在另一部份出現許多次的情況，這樣的處理有失精準。因此我們改用對文字的詞性的處理。

----- prefix -----	--- possible Adj ---	----- verbs -----
DT a the	JJ Adjective	VB Verb, base form
RB Adverb	JJR Adjective, comparative	VBD Verb, past tense
RBR Adverb, comparative	JJS Adjective, superlative	VBG Verb, gerund or present participle
RBS Adverb, superlative		VBN Verb, past participle
	--- other ---	VBP Verb, non-3rd person singular present
	UH Interjection	VBZ Verb, 3rd person singular present

對於文字的詞性，用動詞+形容詞的 **pattern** 來抓取，並且將與 "not" 相連的形容詞都直接併在一起，避免在學習的時候碰到 **good** 與 **not good** 而降低精準度。

此外我們還有嘗試 **Textblob** 套件中的 **correct**，試圖想要修正使用者因為輸入錯誤造成的偏誤，但效果並不是很理想，這個套件修正過後的詞彙常常和正確的答案有出入，因此最後也棄掉了。

因為 **Textblob** 套件中的 **correct** 效果並不好，我們還突發奇想，想把不屬於 'qazwxdvyhujik' 這些字元的單字，若有重複則去除掉，想要藉此來修正輸入錯誤或誇示的表示法，但效果也非常不好，所以作罷 XD。

三、開始學習：

資料的前處理大致上到這邊，接著使用 **TfidfVectorizer**，其中較重要的參數

max_df、min_df 皆設為 1，ngram_range 設為 (1, 1)，stopword 在嘗試過後決定乾脆不加入，結果較好。

在分類機的選擇上，我們使用了 LinearRegression、LogisticRegression、SVC、RandomForestClassifier 四種 Classifier，並且透過不斷的加權來調整最終的預測結果。

四、最終結果：

透過手動調整與不斷地上傳，我們在 0.65*SVR+0.36*SVC 的時候得到最佳的結果，RMSE 為 0.8282、準確度 0.5047。

```
In [59]: cptext = testdata["text"]
cpx = vectorizer.transform(cptext)

#finalpred = 0.11*Logis.predict(cpx) + 0.77*svc.predict(cpx) + 0.12*rf.predict(cpx) + 0.12*Lr.predict(cpx)
#finalpred = 0.11*Logis.predict(cpx) + 0.77*svc.predict(cpx) + 0.06*rf.predict(cpx) + 0.06*Lr.predict(cpx)
#finalpred = 0.16*Logis.predict(cpx) + 0.7*svc.predict(cpx) + 0.07*rf.predict(cpx) + 0.07*Lr.predict(cpx)
#finalpred = 0.16*Logis.predict(cpx) + 0.7*svc.predict(cpx) + 0.14*Lr.predict(cpx) #0.8892
#finalpred = 0.16*Logis.predict(cpx) + 0.7*svc.predict(cpx) + 0.14*svr.predict(cpx) #0.8817
#finalpred = 0.7*svc.predict(cpx) + 0.3*svr.predict(cpx) #0.8544
#finalpred = 0.6*svc.predict(cpx) + 0.3*svr.predict(cpx) + 0.1*Lr.predict(cpx) #0.8440
#finalpred = 0.7*svr.predict(cpx) + 0.3*svc.predict(cpx) #0.8340
#finalpred = 0.64*svr.predict(cpx)+0.36*svc.predict(cpx) #0.8307
#finalpred = 0.64*svr.predict(cpx)+0.32*svc.predict(cpx) #0.8288
#finalpred = 0.66*svr.predict(cpx)+0.33*svc.predict(cpx) #0.8282
finalpred = 0.65*svr.predict(cpx)+0.36*svc.predict(cpx)
```

五、Github 連結：

<https://github.com/a693691122/cp2>