CP2

H24056118 蔡育哲、H24056095 馬少倫、H24054077 陳彥宏

Description：

拿到資料的一開始，我們先用老師上課教的方法，對 text 分別做 Count Vectorizer 和 Tfidf Vectorizer 之後帶入 Logistic Regression、Linear Regression 以及 SVR，並使用 Grid Search CV 選擇參數，得到最好的結果是 RMSE=0.8595，方法如下圖。

```python
import pandas as pd
df1 = pd.read_csv("training_data.csv")
df2 = pd.read_csv("test_data.csv")

text_train = df1["text"]
text_test = df2["text"]
y_train = df1["stars"]

from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer = TfidfVectorizer()
vectorizer.fit(text_train)

X_train = vectorizer.transform(text_train)
X_test = vectorizer.transform(text_test)

from sklearn.svm import SVR
#from sklearn.model_selection import GridSearchCV, KFold
#param_grid = {'C': [0.001, 0.01, 0.1, 1, 10], 'gamma': [0.001, 0.01, 0.1, 1]}

#cv = KFold(shuffle=True)
#grid = GridSearchCV(SVR(), param_grid=param_grid, cv=cv, verbose=3)
#grid.fit(X_train, y_train)
#grid.predict(X_test)
#print(grid.best_score_) # 0.4652
#print(grid.best_params_) # {'C': 10, 'gamma': 1}
clf = SVR(C=10, gamma=1)
clf.fit(X_train, y_train)
y_test = clf.predict(X_test)
```

後來，我們發現 text 裡面有許多相同的字，但因評論者使用的時態不同而有些許差異，於是我們找到了 Snowball Stemmer 去除時

態，將它與 Count Vectorizer 結合之後定義為 Stemmed Count Vectorizer，把 text 丟進去處理，得到的稀疏矩陣再丟進 Tfidf Transformer 計算每個字的重要性。使用 Truncated SVD 將維度降到 3000 之後，最後再使用 SVR 得到最後的結果。

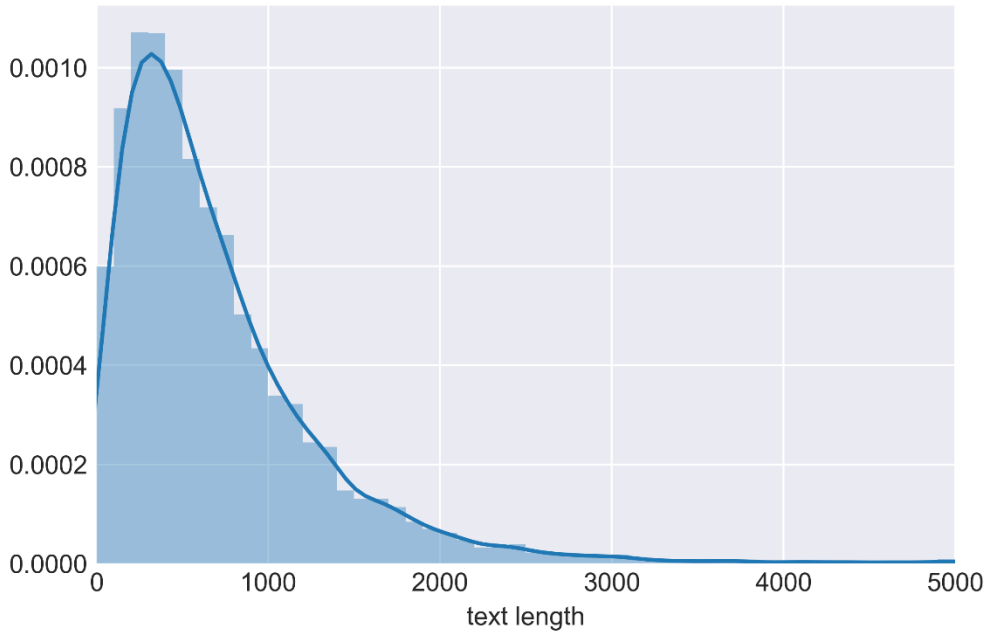
Analysis：

過程中，我們發現比起分類器，用迴歸器做預測得到的 RMSE 相對低很多，因為前者是將預測結果分為[1,2,3,4,5]五類，後者則是將結果當成連續函數預測，不過要注意的是，回歸器做出的預測可能會小於 1 或大於 5，因此我們將小於 1 的值都改成 1，大於 5 的都改成 5，這樣才可以讓 RMSE 得到更小的結果；另外，我們也試過將 text 去掉 stop words 後再處理，得到的結果卻沒有比較好。

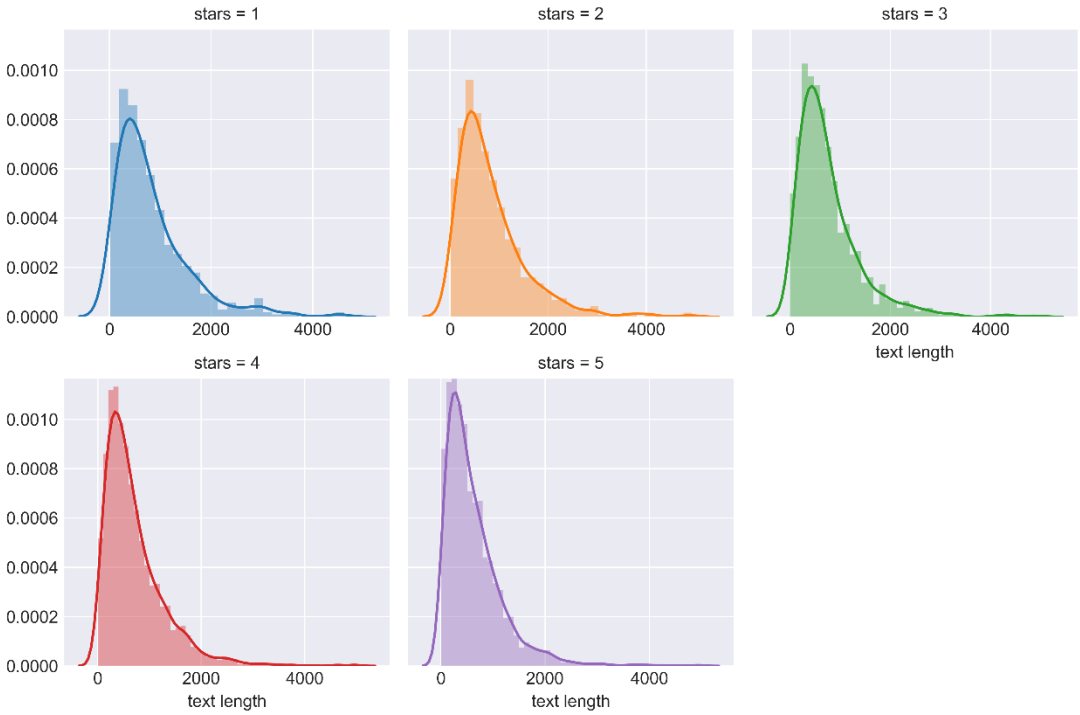
在丟進 Stemmed Count Vectorizer 的時候，我們把參數 ngram_range 設為 1 到 2，把相鄰兩個字變成一個 feature，因為我們考慮到 text 中如果有雙重否定或是有一個正面的形容詞前面加一個 not，就會使原本字的定義跟想要表達的語意相反，而如果一次看兩個字的話就能避免這個情況造成計算的偏差。

Visualization：

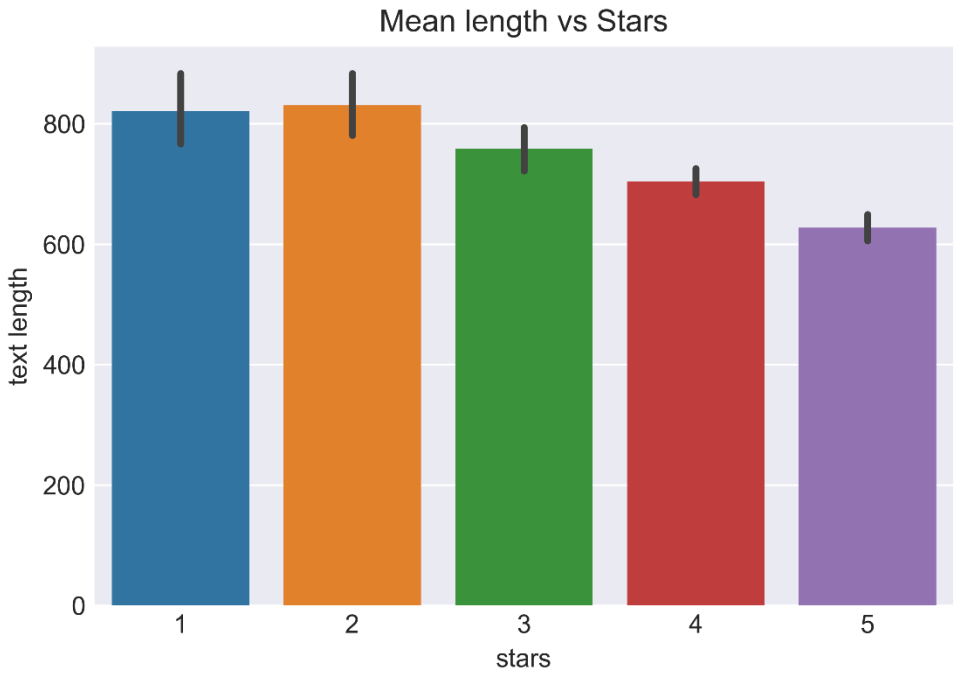
從下列六張圖可以看出在 training_data 裡面，text 的分布以及 text 和 stars 的關係。從「text 長度的平均與 stars 的關係」、「text 長度的平均與 stars 的關係」這兩張圖中，我們可以發現一個趨勢－－stars 愈多 text 的長度愈短，可見得愈爛的電影大家愈會寫很多負面文字來評論，當作抒發心情的管道。



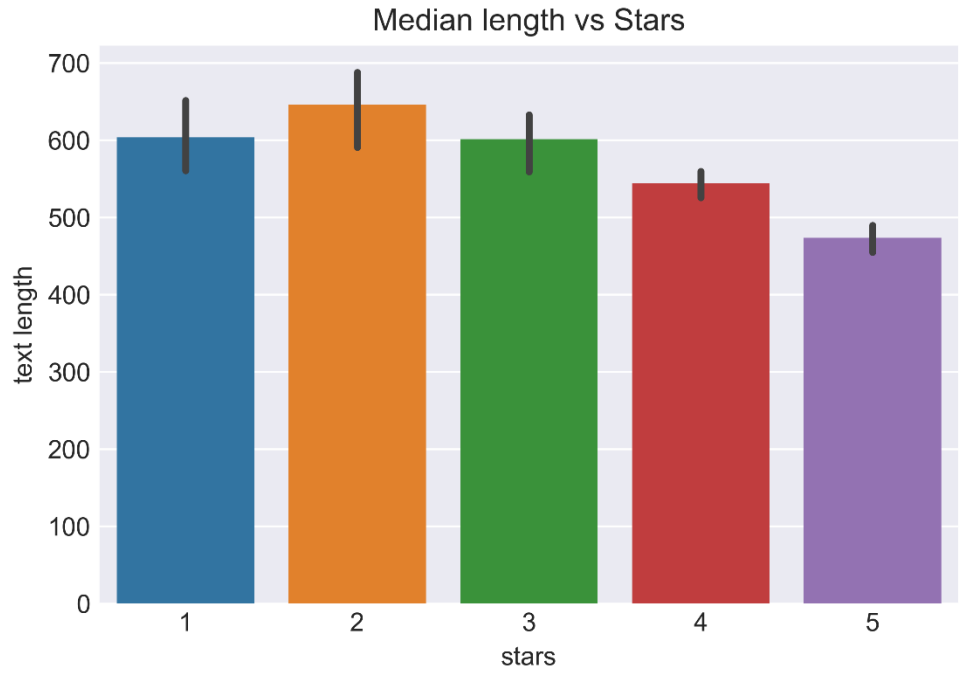
text 長度的分布



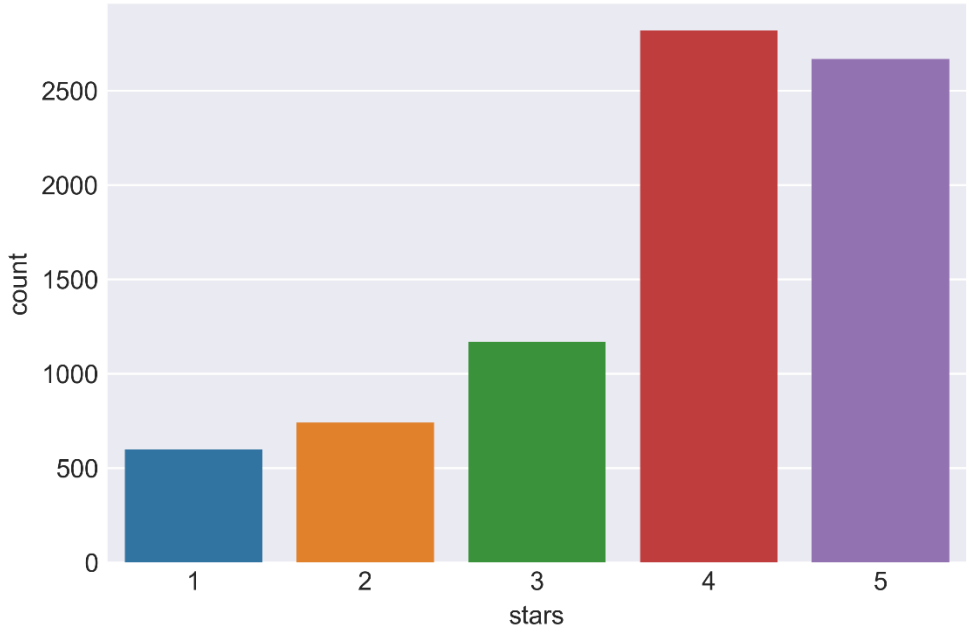
text 長度在不同 stars 的分布



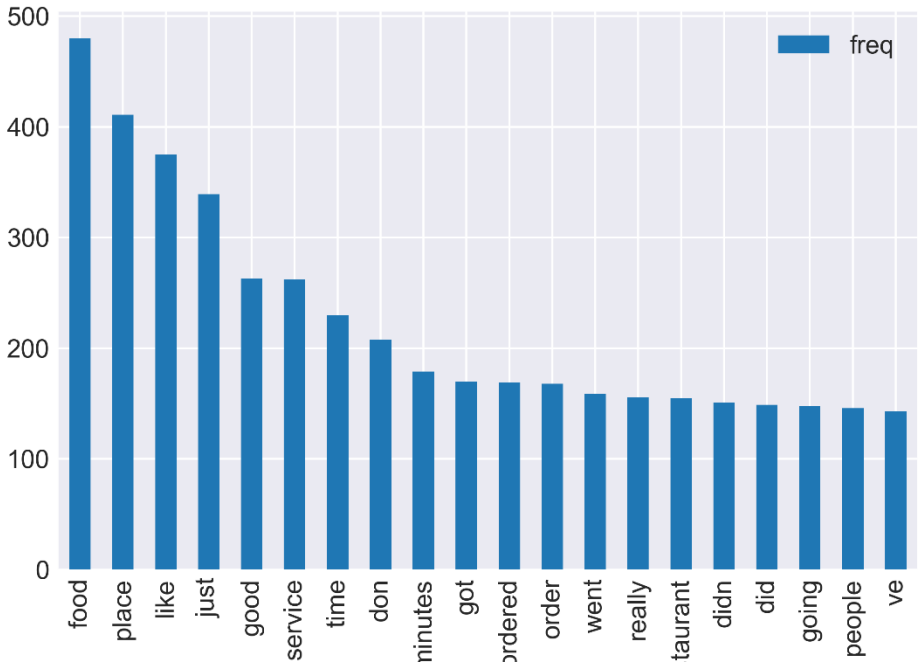
text 長度的平均與 stars 的關係



text 長度的中位數與 stars 的關係



每個 stars 的數量



最常出現的 20 個字

github 連結：

https://github.com/Roytsai27/Rating-Prediction-from-Movie-Review-.git