- [1]：import packages

  [2]：load yelp.csv，查看 yelp 資料，沒有缺值

```
In [1]: import pandas as pd
        import warnings
        warnings.filterwarnings('ignore')
        from sklearn.feature_extraction.text import CountVectorizer
        from sklearn.feature_extraction.text import TfidfTransformer
        from sklearn.feature_extraction.text import TfidfVectorizer
        from sklearn.ensemble import RandomForestClassifier
        from sklearn import metrics
        from sklearn.feature_extraction import text
        import pickle
        import re
```

```
In [2]: yelp = pd.read_csv('./data/yelp.csv')
        yelp.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 10 columns):
 #   Column       Non-Null Count  Dtype 
---  ------       --------------  ----- 
 0   business_id  10000 non-null  object
 1   date         10000 non-null  object
 2   review_id    10000 non-null  object
 3   stars        10000 non-null  int64 
 4   text         10000 non-null  object
 5   type         10000 non-null  object
 6   user_id      10000 non-null  object
 7   cool         10000 non-null  int64 
 8   useful       10000 non-null  int64 
 9   funny        10000 non-null  int64 
dtypes: int64(4), object(6)
memory usage: 781.4+ KB
```

- [3]：yelp.csv 只保留 text 和 stars 兩個欄位，存為 data

```
In [3]: # 讀取csv檔僅保留"text"、"stars"兩個欄位
        data = yelp[['text','stars']]
        display(data.head(5))
```

	text	stars
0	My wife took me here on my birthday for breakf...	5
1	I have no idea why some people give bad review...	5
2	love the gyro plate. Rice is so good and I als...	4
3	Rosie, Dakota, and I LOVE Chaparral Dog Park!!...	5
4	General Manager Scott Petello is a good egg!!!...	5

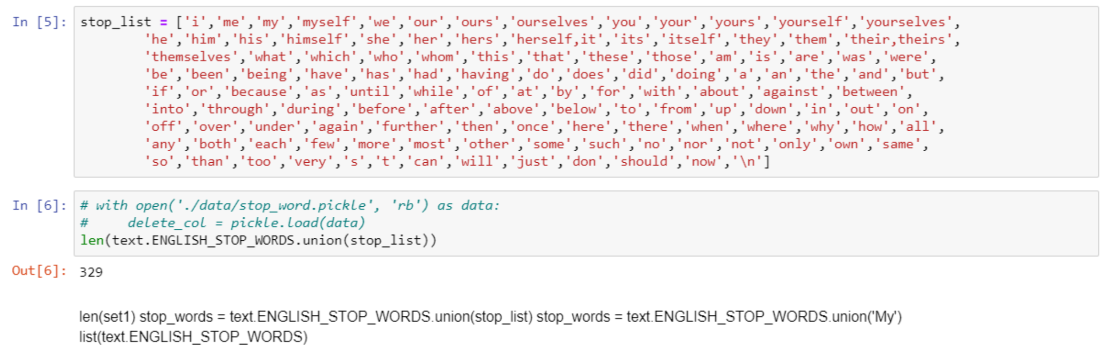
- [4]：將 stars 欄位內值大於等於 4 的轉成 1，小於 4 轉成 0。並將 text 中的文字全部轉成小寫。

```
In [4]: # 將stars欄位內值大於等於4的轉成1，其餘轉成0
        data.loc[data['stars']<4 , 'stars'] = 0
        data.loc[data['stars']>=4 , 'stars'] = 1
        data['text'] = data['text'].str.lower()


        display(data)
        # 1: positive, 0: negative
```

	text	stars
0	my wife took me here on my birthday for breakf...	1
1	i have no idea why some people give bad review...	1
2	love the gyro plate. rice is so good and i als...	1
3	rosie, dakota, and i love chaparral dog park!!...	1
4	general manager scott petello is a good egg!!!...	1
...	...	...
9995	first visit...had lunch here today - used my g...	0
9996	should be called house of deliciousness!\n\ni ...	1
9997	i recently visited olive and ivy for business ...	1
9998	my nephew just moved to scottsdale recently so...	0
9999	4-5 locations.. all 4.5 star average.. i think...	1

10000 rows × 2 columns

- [5]：自己建立 stop list(存成 stop_list)
  [6]：將 stop_list 加入內建的”english” stop word

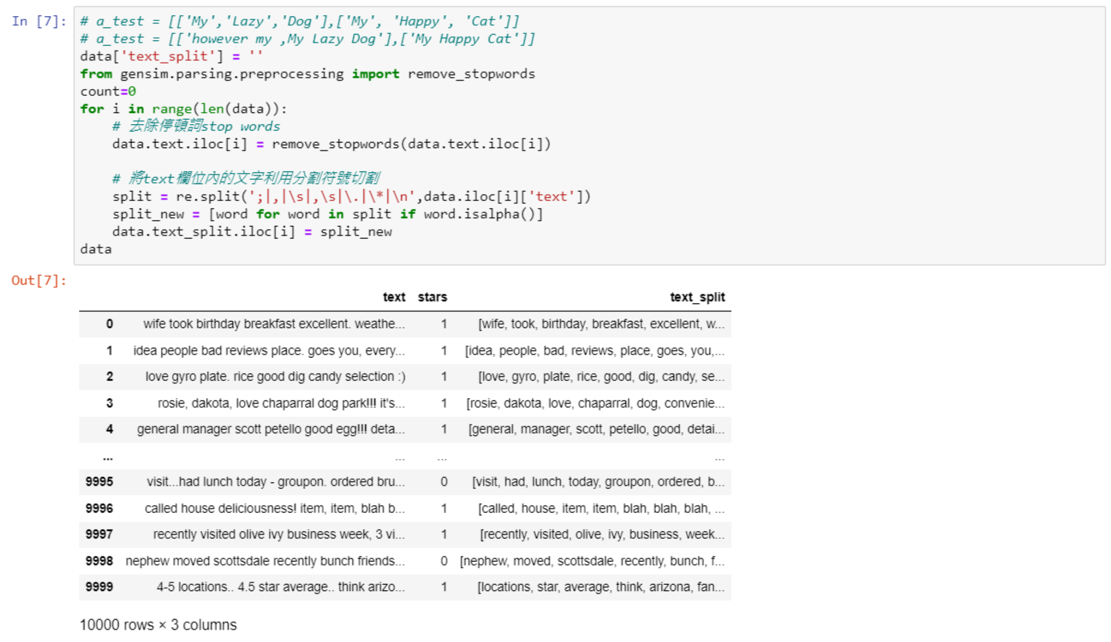


```
In [5]: stop_list = ['i','me','my','myself','we','our','ours','ourselves','you','your','yours','yourself','yourselves',
                'he','him','his','himself','she','her','hers','herself,it','its','itself','they','them','their,theirs',
                'themselves','what','which','who','whom','this','that','these','those','am','is','are','was','were',
                'be','been','being','have','has','had','having','do','does','did','doing','a','an','the','and','but',
                'if','or','because','as','until','while','of','at','by','for','with','about','against','between',
                'into','through','during','before','after','above','below','to','from','up','down','in','out','on',
                'off','over','under','again','further','then','once','here','there','when','where','why','how','all',
                'any','both','each','few','more','most','other','some','such','no','nor','not','only','own','same',
                'so','than','too','very','s','t','can','will','just','don','should','now','\n']
```

```
In [6]: # with open('./data/stop_word.pickle', 'rb') as data:
        #     delete_col = pickle.load(data)
        len(text.ENGLISH_STOP_WORDS.union(stop_list))

Out[6]: 329
```

len(set1) stop_words = text.ENGLISH_STOP_WORDS.union(stop_list) stop_words = text.ENGLISH_STOP_WORDS.union('My') list(text.ENGLISH_STOP_WORDS)

- [7]：用 gensim 內建 remove_stopwords 先移除部分 stop words，經過 re.split 分割，確定分割單位內皆為字母(isalpha)後存入 data[‘text_split’]



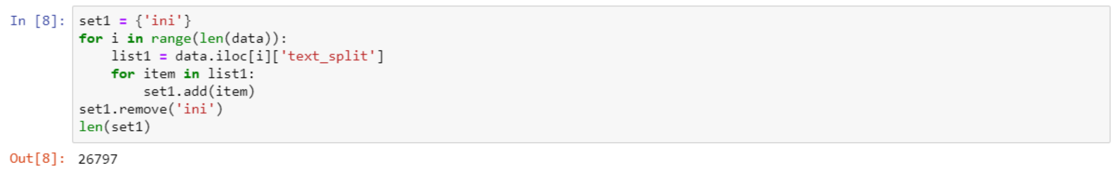
```
In [7]: # a_test = [['My','Lazy','Dog'],['My', 'Happy', 'Cat']]
        # a_test = [['however my ,My Lazy Dog'],['My Happy Cat']]
        data['text_split'] = ''
        from gensim.parsing.preprocessing import remove_stopwords
        count=0
        for i in range(len(data)):
            # 去除停頓詞stop words
            data.text.iloc[i] = remove_stopwords(data.text.iloc[i])

            # 將text欄位內的文字利用分割符號切割
            split = re.split(';|,|\s|,\s|\.|\*|\n',data.iloc[i]['text'])
            split_new = [word for word in split if word.isalpha()]
            data.text_split.iloc[i] = split_new
        data
```

Out[7]:

	text	stars	text_split
0	wife took birthday breakfast excellent. weathe...	1	[wife, took, birthday, breakfast, excellent, w...
1	idea people bad reviews place. goes you, every...	1	[idea, people, bad, reviews, place, goes, you,...
2	love gyro plate. rice good dig candy selection :)	1	[love, gyro, plate, rice, good, dig, candy, se...
3	rosie, dakota, love chaparral dog park!!! it's...	1	[rosie, dakota, love, chaparral, dog, convenie...
4	general manager scott petello good egg!!! deta...	1	[general, manager, scott, petello, good, detai...
...	...	...	...
9995	visit...had lunch today - groupon. ordered bru...	0	[visit, had, lunch, today, groupon, ordered, b...
9996	called house deliciousness! item, item, blah b...	1	[called, house, item, item, blah, blah, blah, ...
9997	recently visited olive ivy business week, 3 vi...	1	[recently, visited, olive, ivy, business, week...
9998	nephew moved scottsdale recently bunch friends...	0	[nephew, moved, scottsdale, recently, bunch, f...
9999	4-5 locations.. 4.5 star average.. think arizo...	1	[locations, star, average, think, arizona, fan...

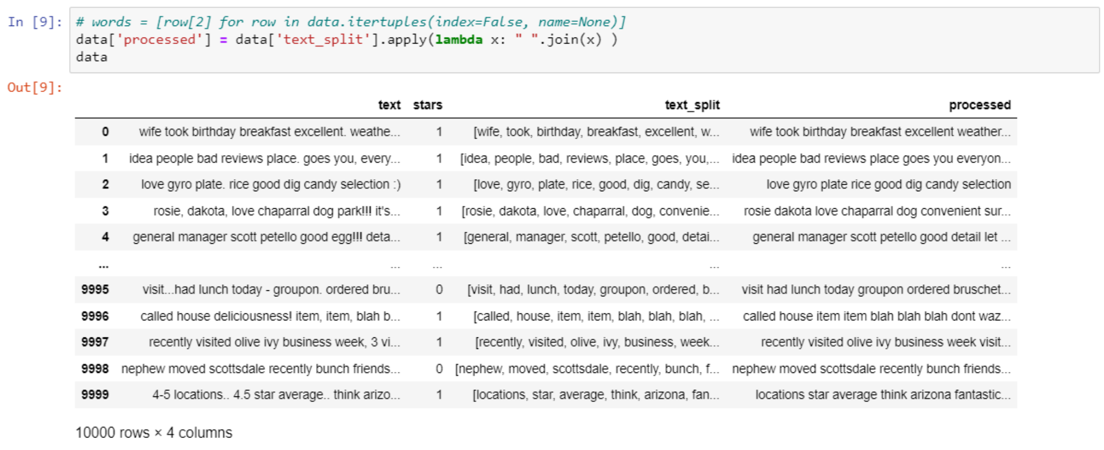
10000 rows × 3 columns

- [8]：確定 data[‘text_split’]中的詞彙數量，跟 countvectorizer 結果維度一樣



```
In [8]: set1 = {'ini'}
        for i in range(len(data)):
            list1 = data.iloc[i]['text_split']
            for item in list1:
                set1.add(item)
        set1.remove('ini')
        len(set1)

Out[8]: 26797
```

- [9]：將 data[‘text_split’]連回字串型態，存為 data[‘processed’]



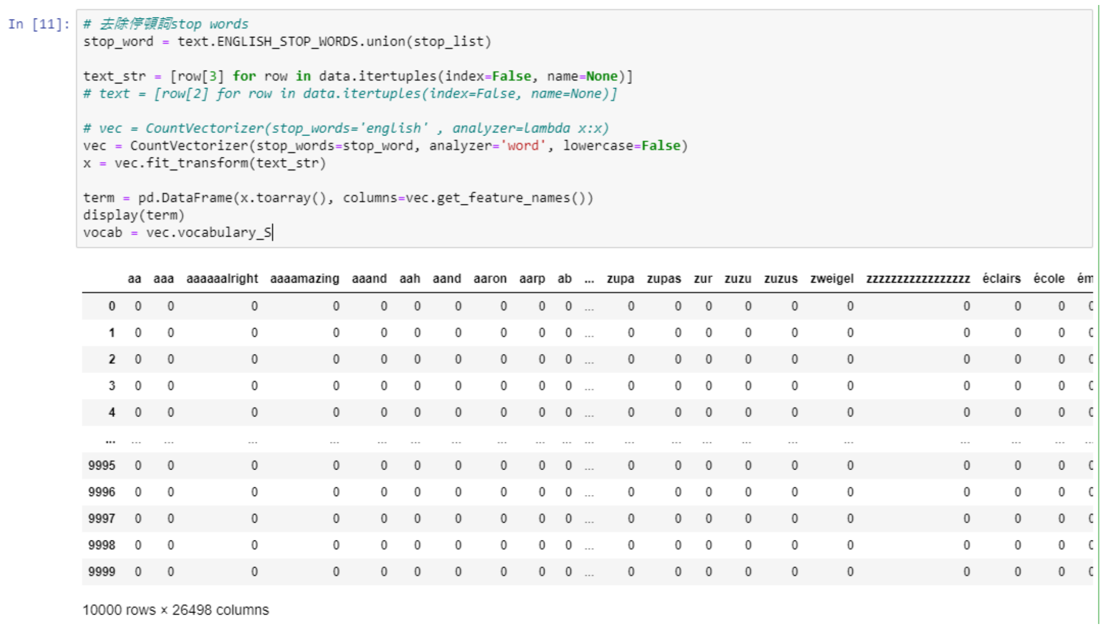
In [9]:
```
# words = [row[2] for row in data.itertuples(index=False, name=None)]
data['processed'] = data['text_split'].apply(lambda x: " ".join(x) )
data
```

Out[9]:

	text	stars	text_split	processed
0	wife took birthday breakfast excellent. weathe...	1	[wife, took, birthday, breakfast, excellent, w...	wife took birthday breakfast excellent weather...
1	idea people bad reviews place. goes you, every...	1	[idea, people, bad, reviews, place, goes, you,...	idea people bad reviews place goes you everyon...
2	love gyro plate. rice good dig candy selection :)	1	[love, gyro, plate, rice, good, dig, candy, se...	love gyro plate rice good dig candy selection
3	rosie, dakota, love chaparral dog park!!! it's...	1	[rosie, dakota, love, chaparral, dog, convenie...	rosie dakota love chaparral dog convenient sur...
4	general manager scott petello good egg!!! deta...	1	[general, manager, scott, petello, good, detai...	general manager scott petello good detail let ...
...	...	...	...	...
9995	visit...had lunch today - groupon. ordered bru...	0	[visit, had, lunch, today, groupon, ordered, b...	visit had lunch today groupon ordered bruschet...
9996	called house deliciousness! item, item, blah b...	1	[called, house, item, item, blah, blah, blah, ...	called house item item blah blah blah dont waz...
9997	recently visited olive ivy business week, 3 vi...	1	[recently, visited, olive, ivy, business, week...	recently visited olive ivy business week visit...
9998	nephew moved scottsdale recently bunch friends...	0	[nephew, moved, scottsdale, recently, bunch, f...	nephew moved scottsdale recently bunch friends...
9999	4-5 locations.. 4.5 star average.. think arizo...	1	[locations, star, average, think, arizona, fan...	locations star average think arizona fantastic...

10000 rows × 4 columns

- [11]：將 data['processed']丟進 countvectorizer 裡面，找出每個詞彙在每個 data[‘text’]列(row)中是否出現。將所有出現過的詞彙存成 vocab。

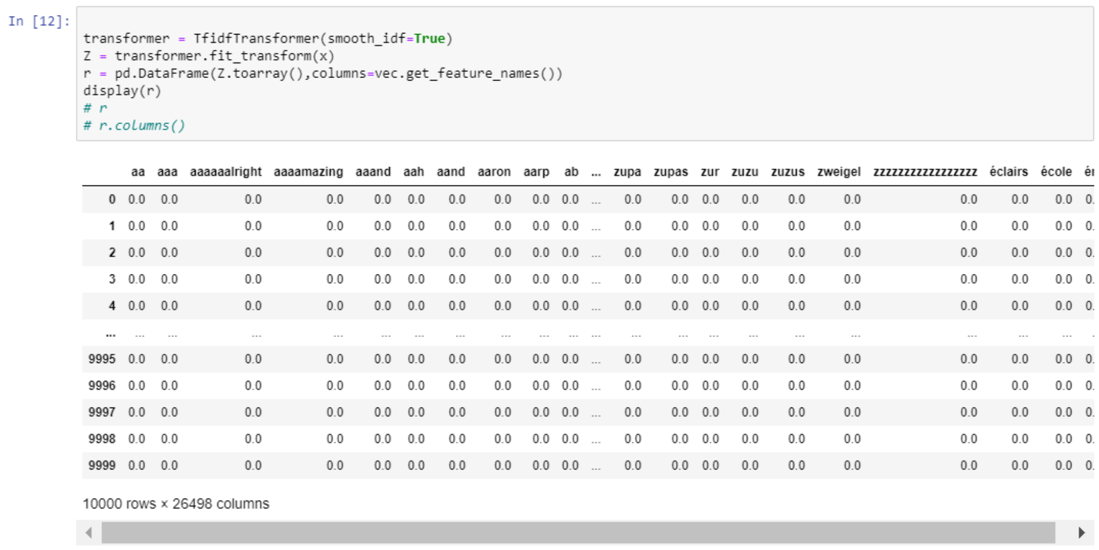


In [11]:
```
# 去除停頓詞stop words
stop_word = text.ENGLISH_STOP_WORDS.union(stop_list)

text_str = [row[3] for row in data.itertuples(index=False, name=None)]
# text = [row[2] for row in data.itertuples(index=False, name=None)]

# vec = CountVectorizer(stop_words='english' , analyzer=lambda x:x)
vec = CountVectorizer(stop_words=stop_word, analyzer='word', lowercase=False)
x = vec.fit_transform(text_str)

term = pd.DataFrame(x.toarray(), columns=vec.get_feature_names())
display(term)
vocab = vec.vocabulary_S
```

	aa	aaa	aaaaaalright	aaaamazing	aaand	aah	aand	aaron	aarp	ab	...	zupa	zupas	zur	zuzu	zuzus	zweigel	zzzzzzzzzzzzzzzz	éclairs	école	ém
0	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
9995	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
9996	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
9997	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
9998	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
9999	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0

10000 rows × 26498 columns

- [12]：用 TfidfTransformer 找到 Tfidf 值，存為 r

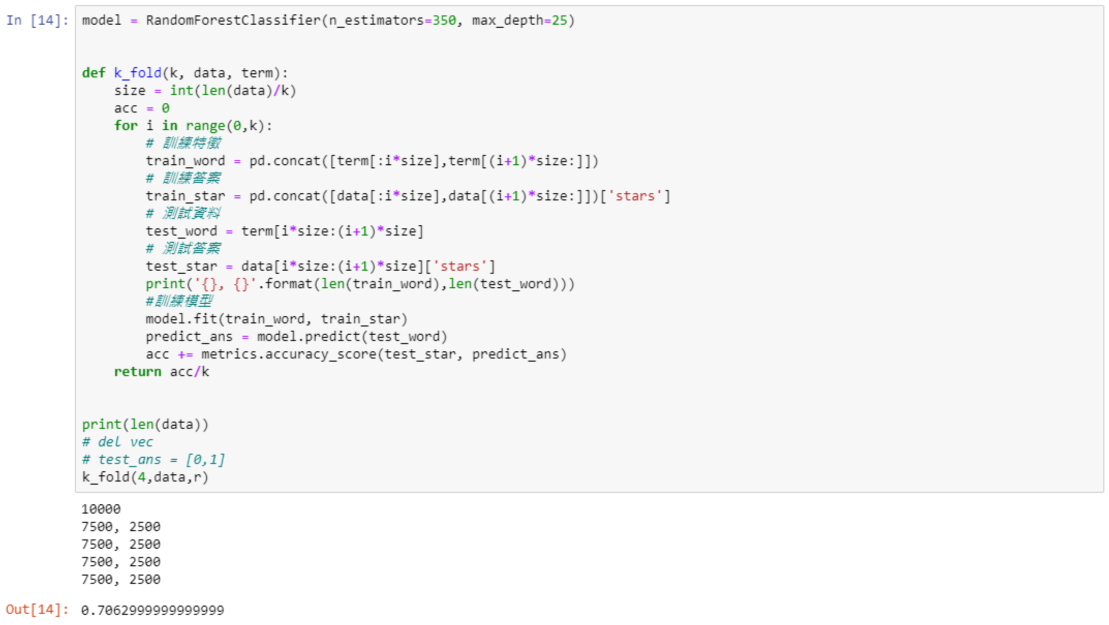


In [12]:
```
transformer = TfidfTransformer(smooth_idf=True)
Z = transformer.fit_transform(x)
r = pd.DataFrame(Z.toarray(),columns=vec.get_feature_names())
display(r)
# r
# r.columns()
```

	aa	aaa	aaaaaalright	aaaamazing	aaand	aah	aand	aaron	aarp	ab	...	zupa	zupas	zur	zuzu	zuzus	zweigel	zzzzzzzzzzzzzzzz	éclairs	école	é
0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.
2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.
3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.
4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	.
9995	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.
9996	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.
9997	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.
9998	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.
9999	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.

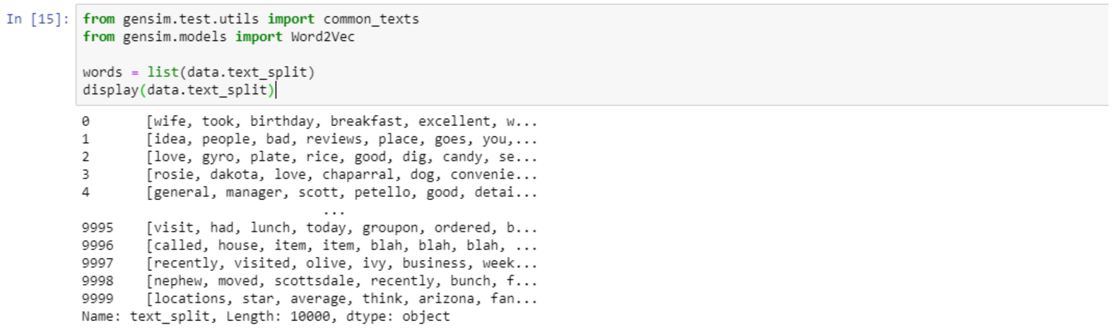
10000 rows × 26498 columns

- [14]：用隨機森林建模，呼叫 k_fole 函數，將資料分為四堆，每次拿三堆訓練一堆測試，得到平均 accuracy 結果約為 0.71



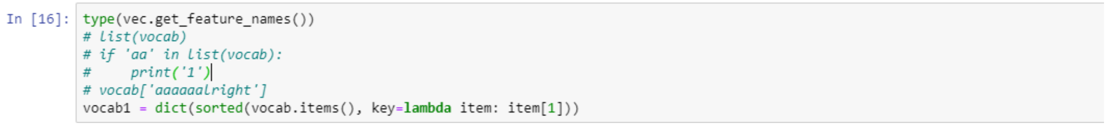
```
In [14]: model = RandomForestClassifier(n_estimators=350, max_depth=25)


         def k_fold(k, data, term):
             size = int(len(data)/k)
             acc = 0
             for i in range(0,k):
                 # 訓練特徵
                 train_word = pd.concat([term[:i*size],term[(i+1)*size:]])
                 # 訓練答案
                 train_star = pd.concat([data[:i*size],data[(i+1)*size:]])['stars']
                 # 測試資料
                 test_word = term[i*size:(i+1)*size]
                 # 測試答案
                 test_star = data[i*size:(i+1)*size]['stars']
                 print('{}, {}'.format(len(train_word),len(test_word)))
                 #訓練模型
                 model.fit(train_word, train_star)
                 predict_ans = model.predict(test_word)
                 acc += metrics.accuracy_score(test_star, predict_ans)
             return acc/k


         print(len(data))
         # del vec
         # test_ans = [0,1]
         k_fold(4,data,r)

         10000
         7500, 2500
         7500, 2500
         7500, 2500
         7500, 2500

Out[14]: 0.7062999999999999
```

- [15]：將 data[‘text split’]轉成 list 型態



```
In [15]: from gensim.test.utils import common_texts
         from gensim.models import Word2Vec

         words = list(data.text_split)
         display(data.text_split)

         0       [wife, took, birthday, breakfast, excellent, w...
         1       [idea, people, bad, reviews, place, goes, you,...
         2       [love, gyro, plate, rice, good, dig, candy, se...
         3       [rosie, dakota, love, chaparral, dog, convenie...
         4       [general, manager, scott, petello, good, detai...
                                       ...
         9995    [visit, had, lunch, today, groupon, ordered, b...
         9996    [called, house, item, item, blah, blah, blah, ...
         9997    [recently, visited, olive, ivy, business, week...
         9998    [nephew, moved, scottsdale, recently, bunch, f...
         9999    [locations, star, average, think, arizona, fan...
         Name: text_split, Length: 10000, dtype: object
```

- [16]：將 vocab 中的 item(所有 countVectorizer 出現過的詞彙)按照 value 進行排序(value 為 term columns 的 Index)



```
In [16]: type(vec.get_feature_names())
         # list(vocab)
         # if 'aa' in list(vocab):
         #     print('1')
         # vocab['aaaaaalright']
         vocab1 = dict(sorted(vocab.items(), key=lambda item: item[1]))
```

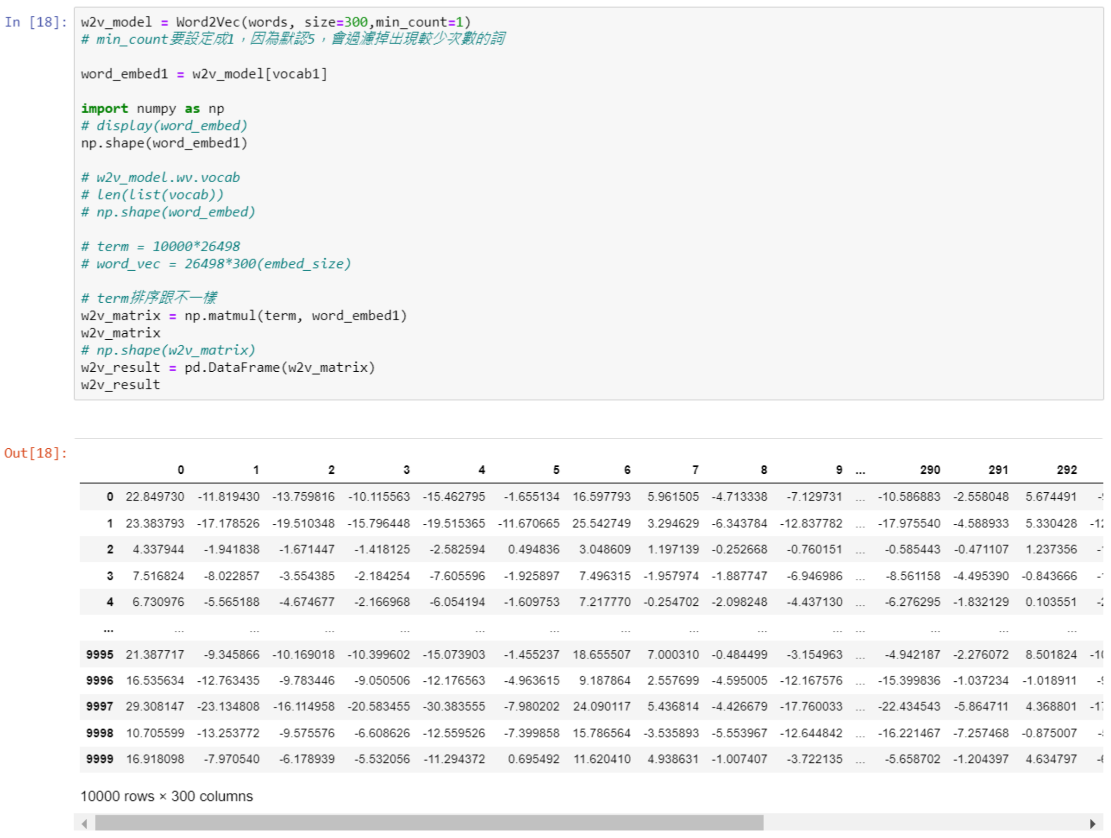
- [18]：訓練 word2vec 模型，訓練文字集合為 words(text 分割後的詞彙 list)，每個詞彙的 embedding size 設為 300，min_count 設為 1(避免默認 5 過濾掉出現次數較少的詞彙)。

  Word_embed1 為將 term 中的詞彙(columns)放入 w2v 模型中，計算出來的 embedding(按照 term columns 的出現順序排列)

  使用 term 和 word_embed1 進行矩陣相乘，會得到每一個 data[‘processed’]的 embedding，因為 term 表示該筆資料(data[‘processed’])中各詞彙的出現次數，word_embed1 則為各詞彙的 embedding。

  Term 的維度應該是 10000(資料筆數)*26498(總詞彙數)，word_embed1 的維度是 26498(總詞彙數)*300(embedding size)，相乘得到 10000*300 的矩

陣，作為該筆資料的 vector(w2v_result)。



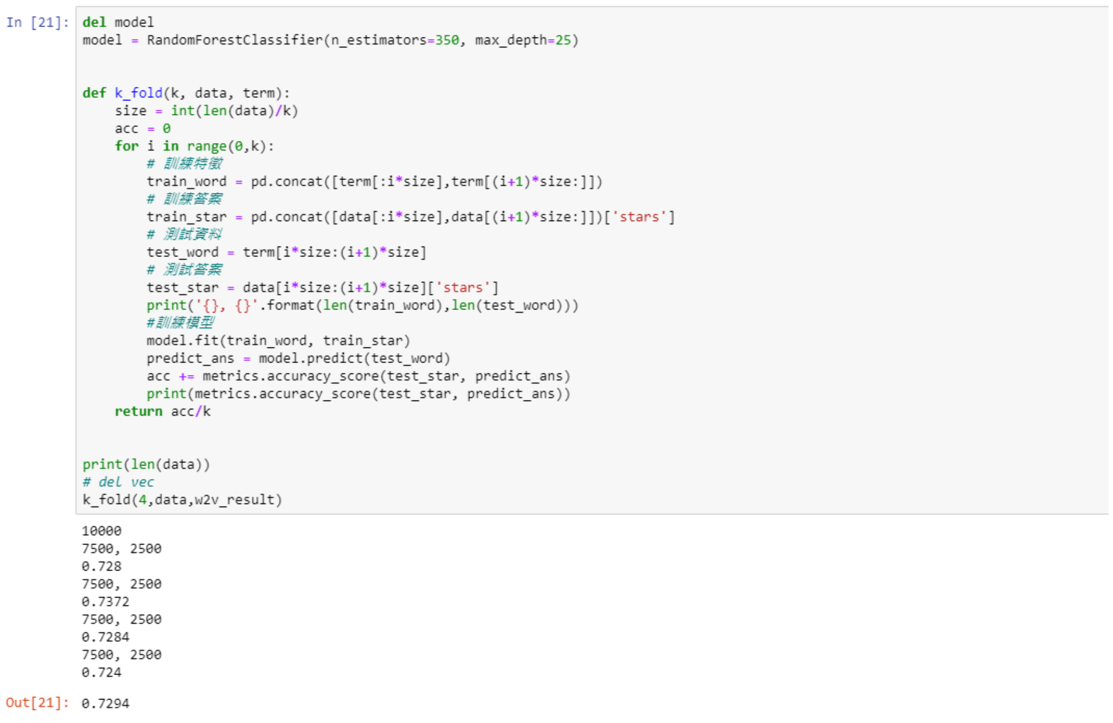
```
In [18]: w2v_model = Word2Vec(words, size=300,min_count=1)
         # min_count要設定成1，因為默認5，會過濾掉出現較少次數的詞

         word_embed1 = w2v_model[vocab1]

         import numpy as np
         # display(word_embed)
         np.shape(word_embed1)

         # w2v_model.wv.vocab
         # len(list(vocab))
         # np.shape(word_embed)

         # term = 10000*26498
         # word_vec = 26498*300(embed_size)

         # term排序跟不一樣
         w2v_matrix = np.matmul(term, word_embed1)
         w2v_matrix
         # np.shape(w2v_matrix)
         w2v_result = pd.DataFrame(w2v_matrix)
         w2v_result
```

Out[18]:

	0	1	2	3	4	5	6	7	8	9	...	290	291	292
0	22.849730	-11.819430	-13.759816	-10.115563	-15.462795	-1.655134	16.597793	5.961505	-4.713338	-7.129731	...	-10.586883	-2.558048	5.674491
1	23.383793	-17.178526	-19.510348	-15.796448	-19.515365	-11.670665	25.542749	3.294629	-6.343784	-12.837782	...	-17.975540	-4.588933	5.330428
2	4.337944	-1.941838	-1.671447	-1.418125	-2.582594	0.494836	3.048609	1.197139	-0.252668	-0.760151	...	-0.585443	-0.471107	1.237356
3	7.516824	-8.022857	-3.554385	-2.184254	-7.605596	-1.925897	7.496315	-1.957974	-1.887747	-6.946986	...	-8.561158	-4.495390	-0.843666
4	6.730976	-5.565188	-4.674677	-2.166968	-6.054194	-1.609753	7.217770	-0.254702	-2.098248	-4.437130	...	-6.276295	-1.832129	0.103551
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
9995	21.387717	-9.345866	-10.169018	-10.399602	-15.073903	-1.455237	18.655507	7.000310	-0.484499	-3.154963	...	-4.942187	-2.276072	8.501824
9996	16.535634	-12.763435	-9.783446	-9.050506	-12.176563	-4.963615	9.187864	2.557699	-4.595005	-12.167576	...	-15.399836	-1.037234	-1.018911
9997	29.308147	-23.134808	-16.114958	-20.583455	-30.383555	-7.980202	24.090117	5.436814	-4.426679	-17.760033	...	-22.434543	-5.864711	4.368801
9998	10.705599	-13.253772	-9.575576	-6.608626	-12.559526	-7.399858	15.786564	-3.535893	-5.553967	-12.644842	...	-16.221467	-7.257468	-0.875007
9999	16.918098	-7.970540	-6.178939	-5.532056	-11.294372	0.695492	11.620410	4.938631	-1.007407	-3.722135	...	-5.658702	-1.204397	4.634797

10000 rows × 300 columns

- [21]：同樣使用隨機森林和 k_fold 函數進行預測，得到平均 accuracy 約為 0.73。使用 word2vec 比使用 tfidf 方法的 acc 高約 0.03 左右。



```
In [21]: del model
         model = RandomForestClassifier(n_estimators=350, max_depth=25)


         def k_fold(k, data, term):
             size = int(len(data)/k)
             acc = 0
             for i in range(0,k):
                 # 訓練特徵
                 train_word = pd.concat([term[:i*size],term[(i+1)*size:]])
                 # 訓練答案
                 train_star = pd.concat([data[:i*size],data[(i+1)*size:]])['stars']
                 # 測試資料
                 test_word = term[i*size:(i+1)*size]
                 # 測試答案
                 test_star = data[i*size:(i+1)*size]['stars']
                 print('{}, {}'.format(len(train_word),len(test_word)))
                 #訓練模型
                 model.fit(train_word, train_star)
                 predict_ans = model.predict(test_word)
                 acc += metrics.accuracy_score(test_star, predict_ans)
                 print(metrics.accuracy_score(test_star, predict_ans))
             return acc/k


         print(len(data))
         # del vec
         k_fold(4,data,w2v_result)

         10000
         7500, 2500
         0.728
         7500, 2500
         0.7372
         7500, 2500
         0.7284
         7500, 2500
         0.724

Out[21]: 0.7294
```