作業三

1.資料前處理 data preprocessing

a. 讀取csv前10000筆，保留text與score

In [1]:
```
import pandas as pd
```

In [2]:
```
df = pd.read_csv(r'Reviews.csv')
```

In [3]:
```
df.head()
```

Out[3]:

	Id	ProductId	UserId	ProfileName	HelpfulnessNumerator	HelpfulnessDenominator	Score	Time	Summary	Text
0	1	B001E4KFG0	A3SGXH7AUHU8GW	delmartian	1	1	5	1303862400	Good Quality Dog Food	I have bought several of the Vitality canned d...
1	2	B00813GRG4	A1D87F6ZCVE5NK	dll pa	0	0	1	1346976000	Not as Advertised	Product arrived labeled as Jumbo Salted Peanut...
2	3	B000LQOCH0	ABXLMWJIXXAIN	Natalia Corres "Natalia Corres"	1	1	4	1219017600	"Delight" says it all	This is a confection that has been around a fe...
3	4	B000UA0QIQ	A395BORC6FGVXV	Karl	3	3	2	1307923200	Cough Medicine	If you are looking for the secret ingredient i...
4	5	B006K2ZZ7K	A1UQRSCLF8GW1T	Michael D. Bigham "M. Wassir"	0	0	5	1350777600	Great taffy	Great taffy at a great price. There was a wid...

In [4]:
```
sample_data = df[['Text','Score']][:10000]
len(sample_data)
```

Out[4]:
```
10000
```

In [5]:
```
sample_data.head()
```

Out[5]:

	Text	Score
0	I have bought several of the Vitality canned d...	5
1	Product arrived labeled as Jumbo Salted Peanut...	1
2	This is a confection that has been around a fe...	4
3	If you are looking for the secret ingredient i...	2
4	Great taffy at a great price. There was a wid...	5

將 "Score" 欄位內值大於等於4的轉成1(positive)，其餘轉成0(negative)

In [6]:
```
sample_data['Score'] = sample_data['Score'].map(lambda x: 1 if x>=4 else 0) #此 map是pandas map不是一般python map
```

In [7]:
```
sample_data.head()
```

Out[7]:

	Text	Score
0	I have bought several of the Vitality canned d...	1
1	Product arrived labeled as Jumbo Salted Peanut...	0
2	This is a confection that has been around a fe...	1
3	If you are looking for the secret ingredient i...	0
4	Great taffy at a great price. There was a wid...	1

將text欄位內的文字利用分割符號切割

In [8]:
```
#sample_data['Text'] = sample_data['Text'].map(lambda x: x.split())
#sample_data['Text'][0]
```

b. 去除停頓詞stop words

In [9]:
```
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn.feature_extraction.text import TfidfVectorizer
```

In [10]:
```
sample_data['Text'][0]
```

Out[10]:
```
'I have bought several of the Vitality canned dog food products and have found them all to be of good quality. The product looks more like a stew than a processed meat and it smells better. My Labrador is finicky and she appreciates this product better than  most.'
```

In [11]:
```
vectorizer = CountVectorizer(stop_words='english')
count_vector = vectorizer.fit_transform(sample_data['Text'])
```

In [12]:
```
transformer = TfidfTransformer()
tfidf_vector = transformer.fit_transform(count_vector)
```

TfidfTransformer(CountVectorizer(input)) == TfidfVectorizer(input)

In [13]:
```
tfidf = pd.DataFrame(tfidf_vector.toarray(),columns=vectorizer.get_feature_names())
tfidf['product'].head(20) #隨便找一個詞來看一下
```

Out[13]:
```
0     0.202600
1     0.168716
2     0.000000
3     0.000000
4     0.000000
5     0.000000
6     0.000000
7     0.000000
8     0.000000
9     0.000000
10    0.000000
11    0.068892
12    0.000000
13    0.000000
14    0.000000
15    0.000000
16    0.000000
17    0.000000
18    0.040345
19    0.000000
Name: product, dtype: float64
```

In [14]:
```
df_tfidf = pd.concat([sample_data,tfidf],join='inner',axis=1)
```

In [15]:
```
df_tfidf.head()
```

Out[15]:

	Text	Score	00	000	0003	000kwh	002	008	0100	0174	...	zon	zoo	zoom	zotz	zucchini	zuke	zukes	zupas	zuppa	ît
0	I have bought several of the Vitality canned d...	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
1	Product arrived labeled as Jumbo Salted Peanut...	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	This is a confection that has been around a fe...	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
3	If you are looking for the secret ingredient i...	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	Great taffy at a great price. There was a wid...	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

5 rows × 18499 columns

w2v實作

In [16]:
```
from gensim.parsing.preprocessing import remove_stopwords
```

In [17]:
```
print(sample_data['Text'][0])
print(remove_stopwords(sample_data['Text'][0]))
```

```
I have bought several of the Vitality canned dog food products and have found them all to be of good quality. The product looks more like a stew than a processed meat and it smells better. My Labrador is finicky and she appreciates this product better than  most.
I bought Vitality canned dog food products good quality. The product looks like stew processed meat smells better. My Labrador finicky appreciates prod
uct better most.
```

In [18]:
```
w2v_words = []
for Texts in sample_data['Text']:
    w2v_words.append(remove_stopwords(Texts))
w2v_words[0]
```

Out[18]:
```
'I bought Vitality canned dog food products good quality. The product looks like stew processed meat smells better. My Labrador finicky appreciates pro
duct better most.'
```

用空白分割句子

In [19]:
```
w2v_words_list = []
for i in range(len(w2v_words)):
    w2v_words_list.append(w2v_words[i].split())
w2v_words_list[0]
```

Out[19]:
```
['I',
 'bought',
 'Vitality',
 'canned',
 'dog',
 'food',
 'products',
 'good',
 'quality.',
 'The',
 'product',
 'looks',
 'like',
 'stew',
 'processed',
 'meat',
 'smells',
 'better.',
 'My',
 'Labrador',
 'finicky',
 'appreciates',
 'product',
 'better',
 'most.']
```

In [20]:
```
from gensim.models import Word2Vec
model = Word2Vec(w2v_words_list,vector_size=300,min_count=10,window=5)
#vector size決定dimension數量(多少個feature)
```

```
#min count 表示詞出現多少次以上才計入
#window Maximum distance between the current and predicted word within a sentence.
```

In [21]:
```
model.wv['product'] #詞所代表的向量
```

Out[21]:
```
array([ 1.72675233e-02,  2.88014889e-01, -6.02449030e-02, -1.15243951e-03,
       -1.29582703e-01, -2.50166565e-01,  1.45383939e-01,  4.64902043e-01,
        2.05339164e-01, -2.28777662e-01,  1.63434520e-01, -4.36167479e-01,
        1.48335367e-01, -1.16442516e-01, -2.19491974e-01,  7.38900900e-02,
        1.50169447e-01, -5.29956110e-02, -1.39451802e-01, -1.86075494e-01,
       -3.92811328e-01, -5.65155260e-02,  1.34550547e-02,  1.21119961e-01,
        1.66424036e-01,  2.30078056e-01, -2.38333046e-01, -5.31360321e-02,
       -1.36958897e-01, -4.73605573e-01, -1.70814840e-03, -1.24676391e-01,
        8.22380781e-02, -8.07563215e-03, -2.22525626e-01,  2.78399587e-01,
        2.64357209e-01, -1.58445701e-01,  1.25250476e-03,  5.38070463e-02,
       -3.11060071e-01,  1.19556747e-01, -4.96183395e-01, -4.54284668e-01,
        3.88917387e-01,  5.13749151e-03,  1.42344937e-01,  7.15091750e-02,
       -1.00792544e-02,  1.31137982e-01, -1.26689374e-01, -5.70178255e-02,
       -4.97880839e-02, -1.30253866e-01,  1.28553912e-01, -2.99032301e-01,
        2.07884178e-01,  1.85118929e-01, -1.00753903e-01,  2.24178314e-01,
        1.46198019e-01, -2.20670879e-01, -1.03361011e-01, -2.94376671e-01,
       -3.30530673e-01,  7.25551397e-02,  4.64257114e-02,  4.67755347e-02,
        7.93063045e-02, -5.37790805e-02,  1.37907743e-01,  5.08176722e-02,
        1.82098731e-01, -5.71260080e-02,  1.53924391e-01,  2.70281672e-01,
       -2.04148926e-02, -9.48284566e-03,  1.71065122e-01,  9.09102485e-02,
       -3.28223407e-01,  2.28484534e-02,  4.21905756e-01,  1.46063492e-01,
       -5.37360720e-02,  2.68982146e-02, -5.90400659e-02,  3.23740244e-01,
        3.31517845e-01,  3.33459139e-01,  6.53560013e-02,  3.01071089e-02,
        2.27496624e-01,  1.04460649e-01,  4.48949546e-01,  2.88127244e-01,
        3.17814387e-02,  1.05344251e-01,  7.31516257e-02,  4.06004727e-01,
       -1.69944033e-01, -1.35403007e-01,  1.91699341e-01,  3.56269270e-01,
       -2.56456107e-01, -2.14181498e-01,  1.37040392e-01, -1.66316645e-03,
       -2.90697038e-01, -2.16859151e-02, -2.78982997e-01, -6.29168469e-03,
       -1.64783254e-01,  2.09121138e-01, -1.32053643e-01,  8.61715749e-02,
        1.19369313e-01,  4.97556359e-01,  4.91384536e-01,  3.77006792e-02,
        1.19253792e-01,  1.36402369e-01, -1.24984182e-01,  2.30125971e-02,
       -1.56490579e-01,  2.05061778e-01,  8.68756995e-02,  3.30510251e-02,
       -9.13038105e-02,  3.05268824e-01,  2.58363396e-01,  8.04558471e-02,
        1.66492179e-01, -1.66900784e-01, -1.29889548e-01, -1.52289716e-03,
       -7.59490058e-02, -2.76377022e-01,  2.75788978e-02, -1.40176833e-01,
        2.19453186e-01, -5.43479621e-01, -2.71690607e-01,  2.62676954e-01,
        5.25891781e-01, -1.80754378e-01, -1.48382246e-01, -2.33019441e-01,
        2.38226354e-02, -1.92602709e-01,  1.10792510e-01, -4.67664510e-01,
       -1.87556639e-01, -2.36596033e-01, -1.19223595e-01,  2.81480849e-02,
       -2.46166080e-01,  1.70834109e-01, -7.55715370e-02,  6.26763999e-02,
       -8.25664178e-02,  6.56677363e-03, -1.51016623e-01,  1.30853638e-01,
        5.16906455e-02,  7.00924024e-02,  1.91141278e-01, -3.87137264e-01,
        7.51783997e-02,  6.40377879e-01,  8.67986977e-02, -4.60270643e-02,
        5.15053794e-02, -2.10269317e-02,  7.97984973e-02, -1.08251926e-02,
       -3.50379735e-01, -1.62946507e-01,  6.18928634e-02, -1.73706487e-02,
       -1.40661195e-01, -1.15009218e-01, -2.95991868e-01, -1.79221675e-01,
       -1.01729386e-01, -3.07885736e-01,  3.53033334e-01,  7.99987391e-02,
       -1.39489090e-02, -1.85217738e-01, -1.45979181e-01,  1.12274796e-01,
       -4.78662491e-01, -1.37943998e-01,  4.56422381e-02, -2.94087380e-01,
        1.72240511e-01,  4.76164185e-02,  1.15251809e-01,  1.40963197e-01,
       -1.94781780e-01, -4.28164527e-02,  8.61747488e-02, -8.56574774e-02,
       -1.35192737e-01,  1.71804070e-01, -1.45677865e-01,  5.20886332e-02,
        2.48694956e-01,  1.24672391e-02, -1.32878214e-01, -3.80855143e-01,
        1.20306283e-01,  9.22107846e-02,  4.59538460e-01, -3.07571262e-01,
       -2.04210747e-02, -3.36357206e-01, -1.67807072e-01, -1.55375823e-01,
        2.11325407e-01, -1.99537203e-01, -4.73675460e-01, -3.35757844e-02,
        6.44944683e-02, -2.43812755e-01,  4.63027894e-01, -2.14741319e-01,
       -1.48171410e-01, -1.08107254e-02,  2.19359055e-01,  1.02271721e-01,
       -4.12839800e-01,  1.63868368e-02,  6.81552067e-02,  1.05806418e-01,
       -3.24247144e-02,  1.34973124e-01,  1.31712958e-01, -1.69510171e-01,
        3.19839984e-01, -1.88570172e-01, -5.08780181e-01,  2.88388729e-01,
        1.49669036e-01,  6.24753302e-04, -1.57403842e-01,  6.55291378e-02,
       -1.02679737e-01,  2.94921517e-01,  1.16797224e-01,  2.17940375e-01,
        1.49810284e-01, -1.49288774e-02, -2.07973301e-01, -9.73565802e-02,
        1.34697720e-01,  1.43017769e-01, -8.15868139e-01, -3.72902513e-01,
       -1.51483864e-01, -3.61158885e-02,  1.21239409e-01, -2.23338336e-01,
        2.71554571e-02, -7.81802833e-02,  4.36930299e-01,  4.42611635e-01,
       -2.66008496e-01,  2.78032243e-01, -1.13256484e-01, -2.17262730e-01,
       -1.34678602e-01,  3.46141197e-02,  8.79722610e-02,  2.27204427e-01,
        1.24367610e-01,  1.04633734e-01, -2.31918976e-01, -3.00580170e-02,
       -8.97282735e-02, -1.80901676e-01,  1.40921194e-02, -4.16245535e-02,
       -3.44065607e-01,  1.60970092e-01, -4.63774621e-01,  8.58251527e-02,
        2.49275133e-01,  1.53937161e-01,  2.30612531e-02,  2.24792175e-02,
        1.21892974e-01, -3.41442287e-01,  1.07280023e-01,  1.52180150e-01,
        3.40344594e-03, -3.46540868e-01, -6.88906237e-02, -4.58777472e-02],
      dtype=float32)
```

In [22]:
```
print(f'總共取{len(model.wv)}個字,每個字有{len(model.wv[0])}個維度')
```

總共取5142個字,每個字有300個維度

In [23]:
```
len((model.wv.index_to_key)) #index為詞，column為feature(300)
```

Out[23]: 5142

再來要計算每句話的平均Feature，先找出哪些詞有出現，再根據有出現的詞做平均

In [24]:
```
import numpy as np
```

In [25]:
```
feature_avg = np.zeros((len(w2v_words_list),len(model.wv[0]))) #總表
sentence_index = 0
w2v_index = model.wv.index_to_key                              # index為詞,col_name為300個 feature
for words in w2v_words_list:                                   # iter 每個 comment
    word_count = 0                                             # 計算重要的 word在所有 comment裡的次數
    feature_vec = np.zeros(len(model.wv[0]),dtype = 'float32') # 該 comment的feature
    for word in words:                                         # iter每個 comment中的每個 word
        if word in model.wv.index_to_key:                      # 若該 word有出現在 wv裡(表示相對重要)
            word_count += 1
            word_vector = model.wv[word]                       # 取出該 word在 w2v裡的300個 feature
            feature_vec = feature_vec + word_vector            # 記得這裡是(1,300)的運算
    avg = feature_vec / word_count                             # 計算該 comment裡，total feature/ 重要 word數量 = avg
    feature_avg[sentence_index] = avg                          # 將 avg儲存在總表裡
```

```
    sentence_index += 1                                    # 移動到下一個 comment
df_w2v = pd.concat([sample_data,pd.DataFrame(feature_avg)],join='inner',axis=1)
df_w2v.head()
```

Out[25]:

	Text	Score	0	1	2	3	4	5	6	7	...	290	291	292	293	294	295	296
0	I have bought several of the Vitality canned d...	1	-0.027959	0.245138	-0.043531	0.119486	-0.024511	-0.281336	0.070776	0.354023	...	-0.009458	0.216101	0.287688	-0.201719	0.197139	0.221891	-0.049641
1	Product arrived labeled as Jumbo Salted Peanut...	0	0.000311	0.226615	-0.049388	0.086323	-0.039253	-0.260148	0.064660	0.342760	...	-0.016318	0.140172	0.179877	-0.155793	0.109781	0.186728	-0.037544
2	This is a confection that has been around a fe...	1	-0.012225	0.234607	-0.052463	0.112305	-0.035047	-0.265333	0.059659	0.357397	...	0.031916	0.149118	0.209803	-0.132396	0.098234	0.220340	0.035993
3	If you are looking for the secret ingredient i...	0	-0.023246	0.241019	-0.006543	0.088321	-0.037592	-0.323466	0.075459	0.457880	...	-0.093818	0.167152	0.293918	-0.106076	0.206972	0.290518	0.124106
4	Great taffy at a great price. There was a wid...	1	0.010424	0.189464	-0.062288	0.092318	-0.033870	-0.217970	0.058472	0.263572	...	0.023523	0.115695	0.144575	-0.100909	0.070387	0.150014	0.018097

5 rows × 302 columns

In [26]:
```
from sklearn import ensemble, preprocessing, metrics
```

In [27]:
```
def k_fold_crossvalidation(k:int ,data): #k = 切幾分

    #切割資料
    subset_size = int(len(data)/k)       #每個子集的長度
    avg_acc = 0                          #等等算平均用的
    total_acc = 0
    for i in range(k):
        start = subset_size * i
        end = start + subset_size
        valid = data[start:end]

        valid_x = valid.drop(['Score','Text'], axis = 1)
        valid_y = valid[['Score']]

        if i == 0:
            train = data[end:]
        else:
            train = pd.concat([data[:start],data[end:]],axis=0,join ='inner',ignore_index=True)
        train_x = train.drop(['Score','Text'], axis = 1)
        train_y = train[['Score']]

        #開始種樹長森林
        forest = ensemble.RandomForestClassifier(n_estimators=100) #n_estimators決定有幾棵樹在森林
        forest_fit = forest.fit(train_x,train_y)
        valid_y_predicted = forest_fit.predict(valid_x)
        sub_acc = metrics.accuracy_score(valid_y,valid_y_predicted)
        print(f'valid為:{start},{end}')
        print(f'此subset_acc為{sub_acc}')

        total_acc += sub_acc
    avg_acc = total_acc / k
    return avg_acc
```

In [28]:
```
k_fold_crossvalidation(4,df_tfidf)
```

```
<ipython-input-27-ef38defc6100>:24: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to
(n_samples,), for example using ravel().
  forest_fit = forest.fit(train_x,train_y)
valid為:0,2500
此subset_acc為0.8124
<ipython-input-27-ef38defc6100>:24: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to
(n_samples,), for example using ravel().
  forest_fit = forest.fit(train_x,train_y)
valid為:2500,5000
此subset_acc為0.8048
<ipython-input-27-ef38defc6100>:24: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to
(n_samples,), for example using ravel().
  forest_fit = forest.fit(train_x,train_y)
valid為:5000,7500
此subset_acc為0.7948
<ipython-input-27-ef38defc6100>:24: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to
(n_samples,), for example using ravel().
  forest_fit = forest.fit(train_x,train_y)
valid為:7500,10000
此subset_acc為0.802
```

Out[28]:
```
0.8035
```

In [29]:
```
k_fold_crossvalidation(4,df_w2v)
```

```
<ipython-input-27-ef38defc6100>:24: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to
(n_samples,), for example using ravel().
  forest_fit = forest.fit(train_x,train_y)
valid為:0,2500
此subset_acc為0.7716
<ipython-input-27-ef38defc6100>:24: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to
(n_samples,), for example using ravel().
  forest_fit = forest.fit(train_x,train_y)
valid為:2500,5000
此subset_acc為0.766
<ipython-input-27-ef38defc6100>:24: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to
(n_samples,), for example using ravel().
  forest_fit = forest.fit(train_x,train_y)
valid為:5000,7500
此subset_acc為0.746
<ipython-input-27-ef38defc6100>:24: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to
(n_samples,), for example using ravel().
  forest_fit = forest.fit(train_x,train_y)
valid為:7500,10000
此subset_acc為0.7592
```

Out[29]:
```
0.7606999999999999
```

In [ ]:
```

```