2 Sentiment Analysis

2.2 Movie Review Data

Let us first start by looking at the data provided with the exercise. We have positive and negative movie reviews labeled by human readers, all positive and negative reviews are in the 'pos' and 'neg' folders respectively. If you look in- side a sample file, you will see that these review messages have been 'tokenized', where all words are separated from punctuations. There are approximately 1000 files in each category with files names starting with cv000, cv001, cv002 and so on. You will split the dataset into training set and testing set.

1. Write some code to load the data from text files.

Load data from text files

```
1. ใช้ glob เพื่ออ่านชื่อไฟล์ทั้งหมดใน directory
2. ใช้ with open ชื่อไฟล์นั้นๆ เพื่ออ่าน text files ที่อยู่ข้างใน แล้ว append ไปเก็บไว้ใน raw_data
```

In [4]:
```
#Import Libraries
import pandas as pd
import numpy as np
import glob
```

In [5]:
```
neg_txt = glob.glob( r'C:\Users\dell\Desktop\module8&9\AI\Lab02\neg\*.txt') #read file name in folder' neg
pos_txt = glob.glob( r'C:\Users\dell\Desktop\module8&9\AI\Lab02\pos\*.txt') #read file name in folder' pos
```

In [6]:
```
# read data in txt.file and append it to list
raw_data = []
for i in range(len(pos_txt)):
    with open(pos_txt[i],'r') as F:
        raw_data.append(F.read())

for i in range(len(neg_txt)):
    with open(neg_txt[i],'r') as F:
        raw_data.append(F.read())
```

2.3 TF-IDF

From a raw text review, you want to create a vector, whose elements indicate the number of each word in each document. The frequency of all words within the documents are the 'features' of this machine learning problem.

A popular method for transforming a text to a vector is called tf-idf, short for term frequencyinverse document frequency.

1. Conduct a research about tf-idf and explain how it works.
2. Scikit-learn provides a module for calculating this, this is called TfidfVec- torizer. You can study how this function is used here:

`http://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html`

Write code to transform your text to tf-idf vector.

TF-idf

```
จะเป็นการหา frequency ของ word ต่างๆที่อยู่ใน raw_data ทั้งหมด แล้ว return ออกมาเป็น column ซึ่งขนาดของ column จะเท่ากับ จำนวน word ที่ไม่ซ้ำกันทั้งหมด
```

โดยในแต่ละ column ค่าออกมาอยู่ในช่วง 0-1

```
1. โดยถ้าเข้าใกล้ 1 หมายถึง มี word นั้นอยู่ใน text.file นั้นเยอะมาก
2. ถ้าเข้าใกล้ 0 หมายถึง มี word นั้นอยู่ใน text.file น้อยมาก
```

In [7]:
```
from sklearn.feature_extraction.text import TfidfVectorizer
Tfid1 = TfidfVectorizer()
feature1 = Tfid1.fit_transform(raw_data)
```

In [8]:
```
data = pd.DataFrame(feature1.toarray())
```

In [17]:
```
data.head()
```

In [17]:

```
data.head()
```

Out[17]:

	0	1	2	3	4	5	6	7	8	9	...	39649	39650	39651	39652	39653	39654	39655	39656	39657	39658
0	0.052257	0.000000	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	0.000000	0.000000	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	0.000000	0.000000	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	0.000000	0.020881	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	0.000000	0.000000	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 × 39659 columns

In [5]:

```
# assign true class
y = np.hstack((np.ones(1000),np.zeros(1000))) #1 = pos , 0 = neg
```

2.4 Classification

Use 4 different models to classify each movie into positive or negative category.

1. K-Nearestneighbormodel,using module `sklearn.neighbors.KNeighborsClassifier`
2. RandomForest, using module `sklearn.ensemble.RandomForestClassifier`
3. SVM, using module `sklearn.svm.SVC`
4. Neural network, using `sklearn.neural_network.MLPClassifier`

You may pick other models you would like to try. Just present results for at least 4 models. Please provide your code for model fitting and cross validation. Calculate your classification accuracy, precision, and recall.

In [6]:

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.neural_network import MLPClassifier
from sklearn.model_selection import cross_val_predict
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report
```

In [7]:

```
KN1 = KNeighborsClassifier()
yhat_KN1 = cross_val_predict(KN1,feature1,y)
scoreKN1 = accuracy_score(y,yhat_KN1)
report = classification_report(y, yhat_KN1)
print('-------------------------------KN1 with feature1----------------------------------\n')
print(report)
print('accuracy = ' + str(scoreKN1))
```

```
-------------------------------KN1 with feature1----------------------------------

             precision    recall  f1-score   support

        0.0       0.80      0.19      0.31      1000
        1.0       0.54      0.95      0.69      1000

avg / total       0.67      0.57      0.50      2000

accuracy = 0.572
```

In [96]:

```
RF1 = RandomForestClassifier()
yhat_RF1 = cross_val_predict(RF1,feature1,y)
scoreRF1 = accuracy_score(y,yhat_RF1)
report = classification_report(y, yhat_RF1)
print('-------------------------------RF1 with feature1----------------------------------\n')
print(report)
print('accuracy = ' + str(scoreRF1))
```

```
-------------------------------RF1 with feature1----------------------------------

             precision    recall  f1-score   support

        0.0       0.62      0.77      0.69      1000
        1.0       0.70      0.53      0.60      1000

avg / total       0.66      0.65      0.65      2000

accuracy = 0.6515
```

In [151]:
```
SVM1 = SVC()
yhat_SVM1 = cross_val_predict(SVM1,feature1,y)
scoreSVM1 = accuracy_score(y,yhat_SVM1)
report = classification_report(y, yhat_SVM1)
print('-------------------------------SVM1 with feature1----------------------------------\n')
print(report)
print('accuracy = ' + str(scoreSVM1))
```

```
-------------------------------SVM1 with feature1----------------------------------

             precision    recall  f1-score   support

        0.0       0.80      0.70      0.74      1000
        1.0       0.73      0.82      0.77      1000

avg / total       0.76      0.76      0.76      2000

accuracy = 0.7585
```

In [10]:
```
MLP1 = MLPClassifier()
yhat_MLP1 = cross_val_predict(MLP1,feature1,y)
scoreMLP1 = accuracy_score(y,yhat_MLP1)
report = classification_report(y, yhat_MLP1)
print('-------------------------------MLP1 with feature1----------------------------------\n')
print(report)
print('accuracy = ' + str(scoreMLP1))
```

```
-------------------------------MLP1 with feature1----------------------------------

             precision    recall  f1-score   support

        0.0       0.84      0.82      0.83      1000
        1.0       0.82      0.84      0.83      1000

avg / total       0.83      0.83      0.83      2000

accuracy = 0.8275
```

2.5 Model Tuning

Can you try to beat the simple model you created above? Here are some things you may try:

- When creating TfidfVectorizer object, you may tweak sublinear_tf parameter which use the tf with logarithmic scale instead of the usual tf.
- You may also exclude words that are too frequent or too rare, by adjusting max_df and min_df.
- Adjusting parameters available in the model, like neural network structure or number of trees in the forest.

Design at least 3 experiments using these techniques. Show your experimental results.

KNeighbors

กำหนด max_df = 0.85 min_df = 0.1

1. `accuracy: 0.676 > 0.572`
2. `precission: 0.68 > 0.67`
3. `recall: 0.68 > 0.57`

เปลี่ยน hyper parameter ค่า k = 300 จาก Default = 5 โดย max_df = 0.85 min_df = 0.1

1. `accuracy: 0.727`
2. `precission: 0.74`
3. `recall: 0.73`

```
In [58]: Tfid2 = TfidfVectorizer(max_df = 0.85,min_df = 0.1)
         feature2 = Tfid2.fit_transform(raw_data)
         yhat_KN2 = cross_val_predict(KN1,feature2,y)
         scoreKN2 = accuracy_score(y,yhat_KN2)
         report = classification_report(y, yhat_KN2)
         print('-------------------------------KN1 with feature2-----------------------------------\n')
         print(report)
         print('accuracy = ' + str(scoreKN2))
```

```
-------------------------------KN1 with feature2-----------------------------------

             precision    recall  f1-score   support

        0.0       0.71      0.60      0.65      1000
        1.0       0.65      0.75      0.70      1000

avg / total       0.68      0.68      0.67      2000

accuracy = 0.676
```

```
In [82]: KN2 = KNeighborsClassifier(n_neighbors = 300)
         yhat_KN2 = cross_val_predict(KN2,feature2,y)
         scoreKN2 = accuracy_score(y,yhat_KN2)
         report = classification_report(y, yhat_KN2)
         print('-------------------------------KN2 with feature2-----------------------------------\n')
         print(report)
         print('accuracy = ' + str(scoreKN2))
```

```
-------------------------------KN1 with feature2-----------------------------------

             precision    recall  f1-score   support

        0.0       0.69      0.84      0.75      1000
        1.0       0.79      0.61      0.69      1000

avg / total       0.74      0.73      0.72      2000

accuracy = 0.727
```

RandomForest

กำหนด max_df = 0.85 min_df = 0.1

```
1. accuracy: 0.6775 > 0.6515
2. precission: 0.68 > 0.66
3. recall: 0.68 > 0.65
```

เปลี่ยน hyper parameter ค่า n_estimators = 300 จาก Default = 10 โดย max_df = 0.85 min_df = 0.1

```
1. accuracy: 0.7855
2. precission: 0.79
3. recall: 0.79
```

```
In [145]: Tfid2 = TfidfVectorizer(max_df = 0.85,min_df = 0.1)
          feature2 = Tfid2.fit_transform(raw_data)
          yhat_RF2 = cross_val_predict(RF1,feature2,y)
          scoreRF2 = accuracy_score(y,yhat_RF2)
          report = classification_report(y, yhat_RF2)
          print('-------------------------------RF1 with feature2-----------------------------------\n')
          print(report)
          print('accuracy = ' + str(scoreRF2))
```

```
-------------------------------RF1 with feature2-----------------------------------

             precision    recall  f1-score   support

        0.0       0.65      0.77      0.71      1000
        1.0       0.72      0.58      0.64      1000

avg / total       0.68      0.68      0.67      2000

accuracy = 0.6775
```

```
In [149]: RF2 = RandomForestClassifier(n_estimators = 1000)
          yhat_RF2 = cross_val_predict(RF2,feature2,y)
          scoreRF2 = accuracy_score(y,yhat_RF2)
          report = classification_report(y, yhat_RF2)
          print('-------------------------------RF2 with feature2----------------------------------\n')
          print(report)
          print('accuracy = ' + str(scoreRF2))
```

```
-------------------------------RF1 with feature2----------------------------------

             precision    recall  f1-score   support

        0.0       0.78      0.80      0.79      1000
        1.0       0.79      0.77      0.78      1000

avg / total       0.79      0.79      0.79      2000

accuracy = 0.7855
```

SVM

กำหนด max_df = 0.85 min_df = 0.1

```
1. accuracy: 0.7285 < 0.7585
2. precission: 0.73 < 0.76
3. recall: 0.73 < 0.76
**max_df และ min_df ทำให้ performance ลดลง
```

เปลี่ยน hyper parameter ค่า kernel = 'sigmoid' จาก Default = 'rbf' โดย max_df,min_df (default)

```
1. accuracy: 0.7585
2. precission: 0.76
3. recall: 0.76
**ผลลัพธ์ไม่ต่าง
```

เปลี่ยน hyper parameter ค่า kernel = 'linear' จาก Default = 'rbf' โดย max_df,min_df (default)

```
1. accuracy: 0.845
2. precission: 0.85
3. recall: 0.84
**ผลลัพธ์ดีขึ้น
```

```
In [159]: Tfid2 = TfidfVectorizer(max_df = 0.85 ,min_df = 0.1 )
          feature2 = Tfid2.fit_transform(raw_data)
          yhat_SVM2 = cross_val_predict(SVM1,feature2,y)
          scoreSVM2 = accuracy_score(y,yhat_SVM2)
          report = classification_report(y, yhat_SVM2)
          print('-------------------------------SVM1 with feature2----------------------------------\n')
          print(report)
          print('accuracy = ' + str(scoreSVM2))
```

```
-------------------------------SVM1 with feature2----------------------------------

             precision    recall  f1-score   support

        0.0       0.77      0.66      0.71      1000
        1.0       0.70      0.80      0.75      1000

avg / total       0.73      0.73      0.73      2000

accuracy = 0.7285
```

```
In [160]: SVM2 = SVC(kernel = 'sigmoid')
          yhat_SVM2 = cross_val_predict(SVM2,feature1,y)
          scoreSVM2 = accuracy_score(y,yhat_SVM2)
          report = classification_report(y, yhat_SVM2)
          print('-------------------------------SVM2 with feature1----------------------------------\n')
          print(report)
          print('accuracy = ' + str(scoreSVM2))
```

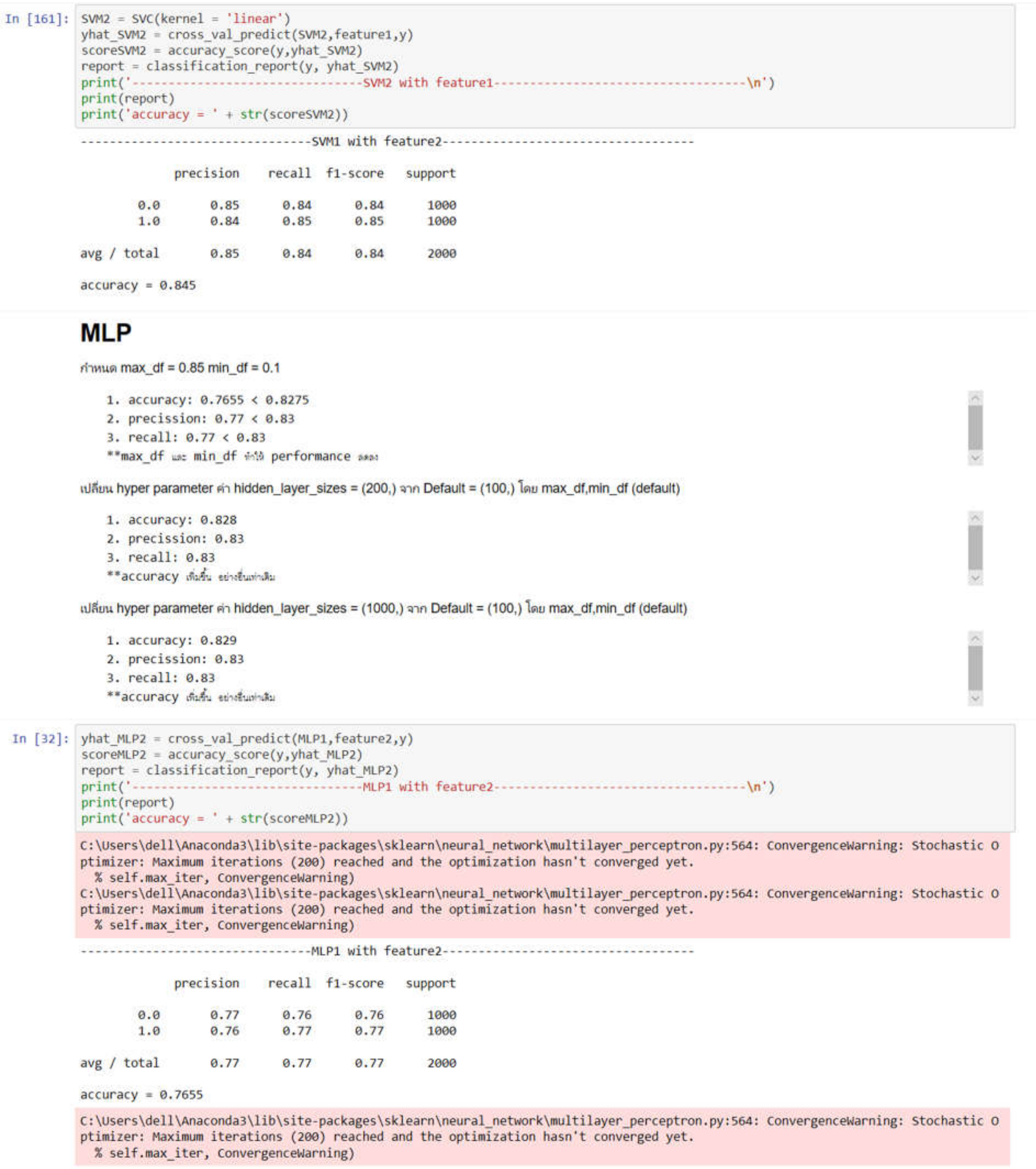
```
-------------------------------SVM1 with feature2----------------------------------

             precision    recall  f1-score   support

        0.0       0.80      0.70      0.74      1000
        1.0       0.73      0.82      0.77      1000

avg / total       0.76      0.76      0.76      2000

accuracy = 0.7585
```



```
In [161]: SVM2 = SVC(kernel = 'linear')
          yhat_SVM2 = cross_val_predict(SVM2,feature1,y)
          scoreSVM2 = accuracy_score(y,yhat_SVM2)
          report = classification_report(y, yhat_SVM2)
          print('-------------------------------SVM2 with feature1----------------------------------\n')
          print(report)
          print('accuracy = ' + str(scoreSVM2))
```

```
-------------------------------SVM1 with feature2----------------------------------

             precision    recall  f1-score   support

        0.0       0.85      0.84      0.84      1000
        1.0       0.84      0.85      0.85      1000

avg / total       0.85      0.84      0.84      2000

accuracy = 0.845
```

MLP

กำหนด max_df = 0.85 min_df = 0.1

```
1. accuracy: 0.7655 < 0.8275
2. precission: 0.77 < 0.83
3. recall: 0.77 < 0.83
**max_df และ min_df ทำให้ performance ลดลง
```

เปลี่ยน hyper parameter ค่า hidden_layer_sizes = (200,) จาก Default = (100,) โดย max_df,min_df (default)

```
1. accuracy: 0.828
2. precission: 0.83
3. recall: 0.83
**accuracy เพิ่มขึ้น อย่างอื่นเท่าเดิม
```

เปลี่ยน hyper parameter ค่า hidden_layer_sizes = (1000,) จาก Default = (100,) โดย max_df,min_df (default)

```
1. accuracy: 0.829
2. precission: 0.83
3. recall: 0.83
**accuracy เพิ่มขึ้น อย่างอื่นเท่าเดิม
```

```
In [32]: yhat_MLP2 = cross_val_predict(MLP1,feature2,y)
         scoreMLP2 = accuracy_score(y,yhat_MLP2)
         report = classification_report(y, yhat_MLP2)
         print('-------------------------------MLP1 with feature2----------------------------------\n')
         print(report)
         print('accuracy = ' + str(scoreMLP2))
```

```
C:\Users\dell\Anaconda3\lib\site-packages\sklearn\neural_network\multilayer_perceptron.py:564: ConvergenceWarning: Stochastic O
ptimizer: Maximum iterations (200) reached and the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
C:\Users\dell\Anaconda3\lib\site-packages\sklearn\neural_network\multilayer_perceptron.py:564: ConvergenceWarning: Stochastic O
ptimizer: Maximum iterations (200) reached and the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
```

```
-------------------------------MLP1 with feature2----------------------------------

             precision    recall  f1-score   support

        0.0       0.77      0.76      0.76      1000
        1.0       0.76      0.77      0.77      1000

avg / total       0.77      0.77      0.77      2000

accuracy = 0.7655
```

```
C:\Users\dell\Anaconda3\lib\site-packages\sklearn\neural_network\multilayer_perceptron.py:564: ConvergenceWarning: Stochastic O
ptimizer: Maximum iterations (200) reached and the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
```

In [168]:

```
MLP2 = MLPClassifier(hidden_layer_sizes= (200,))
yhat_MLP2 = cross_val_predict(MLP2,feature1,y)
scoreMLP2 = accuracy_score(y,yhat_MLP2)
report = classification_report(y, yhat_MLP2)
print('------------------------------MLP2 with feature1----------------------------------\n')
print(report)
print('accuracy = ' + str(scoreMLP2))
```

```
-------------------------------MLP1 with feature2----------------------------------

             precision    recall  f1-score   support

        0.0       0.84      0.81      0.83      1000
        1.0       0.82      0.84      0.83      1000

avg / total       0.83      0.83      0.83      2000

accuracy = 0.828
```

In [169]:

```
MLP2 = MLPClassifier(hidden_layer_sizes= (1000,))
yhat_MLP2 = cross_val_predict(MLP2,feature1,y)
scoreMLP2 = accuracy_score(y,yhat_MLP2)
report = classification_report(y, yhat_MLP2)
print('------------------------------MLP2 with feature1----------------------------------\n')
print(report)
print('accuracy = ' + str(scoreMLP2))
```

```
-------------------------------MLP2 with feature1----------------------------------

             precision    recall  f1-score   support

        0.0       0.84      0.81      0.83      1000
        1.0       0.82      0.84      0.83      1000

avg / total       0.83      0.83      0.83      2000

accuracy = 0.829
```

3 Text Clustering

We have heard about Google News clustering. In this exercise, we are going to implement it with Python.

3.1 Data Preprocessing

Let's switch up and use another dataset called 20newsgroup data, which is a collection of approximately 20,000 newsgroup documents, partitioned (nearly) evenly across 20 different newsgroups. The data is collected from a university's mailing list, where students exchange opinions in everything from motorcycles to middle east politics.

1. Import data using sklearn.datasets.fetch_20newsgroups
2. Transform data to vector with TfidfVectorizer

In [171]:

```
from sklearn.datasets import fetch_20newsgroups
```

In [172]:

```
raw_data = fetch_20newsgroups(subset = 'train') #select data in subset 'train'
```

In [200]:

```
# transform data
Tfidf_google = TfidfVectorizer(max_df = 0.8 , min_df = 0.05)
x = Tfidf_google.fit_transform(raw_data)
```

3.2 Clustering

We are going to use the simplest clustering model, k-means clustering, to do this task. Our hope is that this simple algorithm will result in meaningful news categories, without using labels.

1. Fit K-Means clustering model to the text vector. What is the value of K you should pick? Why?
2. Use Silhouette score to evaluate your clusters. Try to evaluate the model for different values of k to see which k fits best for the dataset.

KMeans

```
select k = 20 เพราะ data ที่ import เข้ามามี 20 groups
```

Result

```
1. K = 20 Silhoutte score =  0.00307
2. K = 15 Silhoutte score =  0.00484
3. K = 10 Silhoutte score = -0.00039
4. K = 5  Silhoutte score =  0.00411
5. K = 35 Silhoutte score =  0.00525
6. K = 50 Silhoutte score =  0.00913
```

จากผลลัพธ์ K = 50 ดีที่สุด แต่ผมคิดว่ามันไม่ควรจะแบ่งได้ 50 cluster จึงเลือก K = 15 เพราะใน data จริงๆ จาก 20 group มันสามารถจัดกลุ่มความคล้ายกันทำให้ group น้อยลงได้

```
In [201]: from sklearn.cluster import KMeans
          from sklearn.metrics import silhouette_score
```

```
In [202]: kmean = KMeans(n_clusters = 20)
          kmean.fit(x)
          sil_score = silhouette_score(x,kmean.labels_ )
          print('----------------------------K = 20--------------------------------')
          print('silhouette score = '+ str(sil_score))

          ----------------------------K = 20--------------------------------
          silhouette score = 0.003071184841869602
```

```
In [203]: kmean1 = KMeans(n_clusters = 15)
          kmean1.fit(x)
          sil_score = silhouette_score(x,kmean1.labels_ )
          print('----------------------------K = 15--------------------------------')
          print('silhouette score = '+ str(sil_score))

          ----------------------------K = 15--------------------------------
          silhouette score = 0.004844898597086147
```

```
In [204]: kmean2 = KMeans(n_clusters = 10)
          kmean2.fit(x)
          sil_score = silhouette_score(x,kmean2.labels_ )
          print('----------------------------K = 10--------------------------------')
          print('silhouette score = '+ str(sil_score))

          ----------------------------K = 10--------------------------------
          silhouette score = -0.0003900338875452058
```

```
In [205]: kmean3 = KMeans(n_clusters = 5)
          kmean3.fit(x)
          sil_score = silhouette_score(x,kmean3.labels_ )
          print('----------------------------K = 5--------------------------------')
          print('silhouette score = '+ str(sil_score))

          ----------------------------K = 5--------------------------------
          silhouette score = 0.004115487009679443
```

```
In [206]: kmean4 = KMeans(n_clusters = 35)
          kmean4.fit(x)
          sil_score = silhouette_score(x,kmean4.labels_ )
          print('----------------------------K = 35--------------------------------')
          print('silhouette score = '+ str(sil_score))

          ----------------------------K = 35--------------------------------
          silhouette score = 0.005255762323989831
```

```
In [207]: kmean5 = KMeans(n_clusters = 50)
          kmean5.fit(x)
          sil_score = silhouette_score(x,kmean5.labels_ )
          print('----------------------------K = 50--------------------------------')
          print('silhouette score = '+ str(sil_score))

          ----------------------------K = 50--------------------------------
          silhouette score = 0.009132529670051202
```

3.3 Topic Terms

We want to explore each cluster to understand what news articles are in the cluster, what terms are associated with the cluster. This will require a bit of hacking.

1. Use TfidfVectorizer.get feature names to extract words associated with each dimension of the text vector.
2. Extract cluster's centroids using kmeans.cluster centers .
3. For each centroid, print the top 15 words that have the highest frequency.

Kmeans clustering K = 15

```
TOP 15 word in each clusters
```

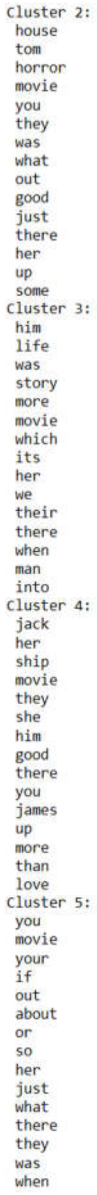
1. Cluster 0: her she movie you was they when their like about there up so out which
2. Cluster 1: movie was you they what just there horror me my like so about or good
3. Cluster 2: house tom horror movie you they was what out good just there her up some
4. Cluster 3: him life was story more movie which its her we their there when man into
5. Cluster 4: jack her ship movie they she him good there you james up more than love
6. Cluster 5: you movie your if out about or so her just what there they was when
7. Cluster 6: alien ship they movie was which her earth more so its than dr there when
8. Cluster 7: they movie there their up joe you out which like comedy funny when can was
9. Cluster 8: mr robin which was her no characters there would out while more films character series
10. Cluster 9: action movie van you was some like plot so there they good out bad scenes
11. Cluster 10: war private men they movie their battle was british american world most mission action about
12. Cluster 11: killer you horror murder movie more her was its about what up they too which
13. Cluster 12: star effects planet science special fiction movie was space they series so like some we
14. Cluster 13: they you we so was about like their there movie what or just up out
15. Cluster 14: 10 you movie was me also about little some plot my well did see out

```
In [208]: feat = Tfidf_google.get_feature_names()
```

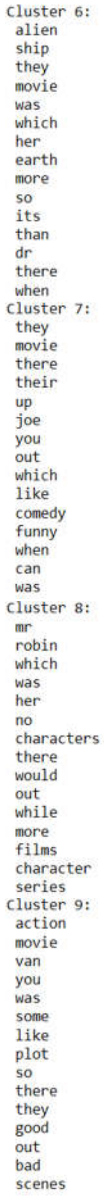
```
In [214]: cluster_centers = kmean1.cluster_centers_
```

```
In [215]: order_centroids = cluster_centers.argsort()[:, ::-1]
          for i in range(len(cluster_centers)):
              print("Cluster %d:" % i),
              for ind in order_centroids[i, :15]:
                  print(' %s' % feat[ind])
```

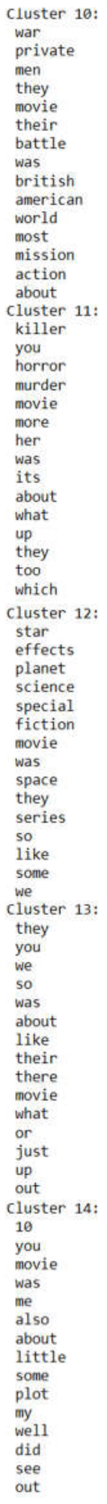
```
Cluster 0:
 her
 she
 movie
 you
 was
 they
 when
 their
 like
 about
 there
 up
 so
 out
 which
Cluster 1:
 movie
 was
 you
 they
 what
 just
 there
 horror
 me
 my
 like
 so
 about
 or
 good
```



```
Cluster 2:
 house
 tom
 horror
 movie
 you
 they
 was
 what
 out
 good
 just
 there
 her
 up
 some
Cluster 3:
 him
 life
 was
 story
 more
 movie
 which
 its
 her
 we
 their
 there
 when
 man
 into
Cluster 4:
 jack
 her
 ship
 movie
 they
 she
 him
 good
 there
 you
 james
 up
 more
 than
 love
Cluster 5:
 you
 movie
 your
 if
 out
 about
 or
 so
 her
 just
 what
 there
 they
 was
 when
```



```
Cluster 6:
 alien
 ship
 they
 movie
 was
 which
 her
 earth
 more
 so
 its
 than
 dr
 there
 when
Cluster 7:
 they
 movie
 there
 their
 up
 joe
 you
 out
 which
 like
 comedy
 funny
 when
 can
 was
Cluster 8:
 mr
 robin
 which
 was
 her
 no
 characters
 there
 would
 out
 while
 more
 films
 character
 series
Cluster 9:
 action
 movie
 van
 you
 was
 some
 like
 plot
 so
 there
 they
 good
 out
 bad
 scenes
```



```
Cluster 10:
 war
 private
 men
 they
 movie
 their
 battle
 was
 british
 american
 world
 most
 mission
 action
 about
Cluster 11:
 killer
 you
 horror
 murder
 movie
 more
 her
 was
 its
 about
 what
 up
 they
 too
 which
Cluster 12:
 star
 effects
 planet
 science
 special
 fiction
 movie
 was
 space
 they
 series
 so
 like
 some
 we
Cluster 13:
 they
 you
 we
 so
 was
 about
 like
 their
 there
 movie
 what
 or
 just
 up
 out
Cluster 14:
 10
 you
 movie
 was
 me
 also
 about
 little
 some
 plot
 my
 well
 did
 see
 out
```