CO544: ASSIGNMENT

E/14/244

PERERA M.S.S.M.

Step1: Loading the data set

Trainset.txt has only training data

Read txt file and write data to the csv with titles

```
import csv
import pandas as pd

with open('trainset.txt', 'r') as in_file:
    stripped = []
    for line in in_file:
        sen = line.strip()
        stripped.append(sen)

    lines = (line.split("\t") for line in stripped if line)
    with open('log.csv', 'w') as out_file:
        writer = csv.writer(out_file)
        writer.writerow(('class', 'title','date','body'))
        writer.writerows(lines)
```

Read csv and create two data frames which are Y-class and X-combination of title, date and body.

```
fileName = 'log.csv'
data = pd.read_csv(fileName)
Y = data['class']
data.drop(['class'], axis=1, inplace=True)
X = data['title'] + " " + data['date'] + " " + data['body']
```

Step2: Removing features from text files

Text file is actually a vocabulary (orders). In order to run a mechanical learning algorithm, the text features of the numerical characters need to be translated into translation. We will use the bag in the Word form for our model. In short, we will calculate each text file for each text file (literally English division) and the number of times in each document, and in the end, each word will be assigned the full identification number.

we can even reduce the weightage of more common words like (the, is, an etc.) which occurs in all document.

```
from nltk.stem.porter import PorterStemmer # for stemming
from sklearn.feature_extraction.text import CountVectorizer # for extract words frequncies

porter_stemmer = PorterStemmer()
def stemming_tokenizer(str_input):
    words = re.sub(r"[^a-z]", " ", str_input).lower().split()
    words = [porter_stemmer.stem(word) for word in words]
    return words
```

In counting the number of words in each document, 1 question: Longer documents will be more weight than short texts. In order to prevent this, the frequency (TF - frequency) i.e. #count (word) / #Total words can be used in all documents.

```
import re

vec = CountVectorizer(stop_words='english', tokenizer=stemming_tokenizer)

matrix = vec.fit_transform(X)

results = pd.DataFrame(matrix.toarray(), columns=vec.get_feature_names())
```

```
6
7 results = pd.DataFrame(matrix.toarray(), columns=vec.get_feature_names())
8 print(results)
```

```
     ..  ..  ..  ...  ...  ...  ...  ...  ...  ...  ...  ...
26    0   0   0    0    0    0    0    0    0    0    0  ...
27    0   0   0    0    0    0    0    0    0    0    0  ...
28    0   0   0    0    0    0    0    0    0    0    0  ...
29    0   0   0    0    0    0    0    0    1    0    0  ...
..   ..  ..  ..  ...  ...  ...  ...  ...  ...  ...  ...  ...
170   0   0   0    0    0    0    0    0    0    0    0  ...
171   0   0   0    0    0    0    0    0    0    0    0  ...
172   0   0   0    0    0    0    0    0    0    0    0  ...
173   0   0   0    0    0    0    0    0    0    0    0  ...
174   0   0   0    0    0    0    0    0    0    0    0  ...
175   1   2   0    0    0    0    0    0    0    0    0  ...
176   0   0   0    0    0    0    0    0    0    0    0  ...
177   0   0   0    0    0    0    0    0    0    0    0  ...
178   0   0   0    0    0    0    0    0    0    0    0  ...
179   0   0   0    0    0    0    0    0    0    0    0  ...
180   0   0   0    0    0    0    0    0    0    0    0  ...
181   0   0   0    0    0    0    0    0    0    0    0  ...
182   0   1   0    0    0    0    0    0    0    0    0  ...
183   0   0   0    0    0    0    1    0    0    0    0  ...
```

Step3: Machine learning algorithem

Naïve Bayes (NB)

There are various algorithms which can be used for text classification. We will start with the simplest one 'Naive Bayes (NB)'. One over third of train data used for training and other part used for testing.

```
from sklearn.naive_bayes import MultinomialNB
from sklearn.model_selection import train_test_split, cross_val_score

nclf=MultinomialNB()
X_train , X_test , Y_train , Y_test = train_test_split(matrix, Y, test_size =0.333,random_state =0)
Y_pred = nclf.fit(X_train , Y_train ).predict(X_test)
np.mean(Y_pred == Y_test)
```

The accuracy we get is **97.015%** which is good. Then test accuracy by another classifier called Decision Tree.

Decision Tree

```
from sklearn.tree import DecisionTreeClassifier

dt_clf = DecisionTreeClassifier()
dt_Y_pred = dt_clf.fit(X_train , Y_train ).predict(X_test)
np.mean(dt_Y_pred == Y_test)*100
```

The accuracy we get is **92.537%**. It is little worse.

K-Nearest Neighbors

```
from sklearn.neighbors import KNeighborsClassifier

knn = KNeighborsClassifier()
knn_Y_pred = dt_clf.fit(X_train , Y_train ).predict(X_test)
np.mean(knn_Y_pred == Y_test)*100
```

The accuracy we get is **94.030%**. It is little better than Decision Tree.

Linear Discriminant Analysis

```
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis

lda = LinearDiscriminantAnalysis()
lda_Y_pred = dt_clf.fit(X_train , Y_train ).predict(X_test)
np.mean(lda_Y_pred == Y_test)*100
```

The accuracy we get is **92.537%**. It also same as Decision Tree.

Gaussian Naïve Bayes

```
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis

lda = LinearDiscriminantAnalysis()
lda_Y_pred = dt_clf.fit(X_train , Y_train ).predict(X_test)
np.mean(lda_Y_pred == Y_test)*100
```

The accuracy we get is **88.060%**. It is the lowest accuracy.

Support Vector Machine

```
from sklearn.svm import SVC

svm = SVC()
svm_Y_pred = dt_clf.fit(X_train , Y_train ).predict(X_test)
np.mean(svm_Y_pred == Y_test)*100
```

The accuracy we get is **91.044%**. Compare with accuracies of the above classifiers, Naïve Bayes (NB) has the best accuracy. So it is choosed as classifer for the prediction.

Confution matrix

```
from sklearn import metrics

print(metrics.confusion_matrix(Y_test, Y_pred))
```

```
[[23  0]
 [ 2 42]]
```

Classification report

```
from sklearn.metrics import classification_report

target_names =["+1", "-1"]
print(classification_report(Y_test, Y_pred,
target_names=target_names)) #Actual , Prediction
```

	precision	recall	f1-score	support
+1	0.92	1.00	0.96	23
-1	1.00	0.95	0.98	44
avg / total	0.97	0.97	0.97	67

Step4: Training(Using all data)

```
from sklearn.naive_bayes import MultinomialNB
import numpy as np

clf = MultinomialNB().fit(matrix, Y)
```

Step5: Loading testsetwithoutlabels.txt

```
with open('testsetwithoutlabels.txt', 'r') as in_file:
    stripped = (line.strip() for line in in_file)
    lines = (line.split("\t") for line in stripped if line)
    with open('test.csv', 'w') as out_file:
        writer = csv.writer(out_file)
        writer.writerow(('title','date','body'))
        writer.writerows(lines)

test_file = 'test.csv'
test_data = pd.read_csv(test_file)
X_test = test_data['title'] + " " + test_data['date'] + " " + test_data['body']
```

Read txt file, execute csv file by splitting tab. Then combine title, date and body as one column.

Step6: Prediction

```
#predict testsetwithoutlabels.txt values
k=0
for i in clf.predict(vec.transform(X_test)):
    print ("{0} {1}".format(i, X_test[k]) )
```

```
In [17]: clf.predict(vec.transform(X_test))

Out[17]: array([ 1,  1,  1, -1, -1,  1, -1, -1, -1, -1, -1, -1,  1,  1, -1,  1,  1,
                 1,  1,  1, -1, -1, -1, -1, -1, -1,  1, -1, -1, -1, -1,  1, -1,  1,
                -1,  1, -1,  1, -1, -1, -1,  1, -1, -1,  1, -1,  1, -1,  1, -1,  1,
                 1, -1, -1,  1,  1,  1,  1,  1, -1, -1, -1, -1, -1,  1, -1,  1,  1,
                 1, -1,  1, -1, -1, -1,  1, -1,  1,  1,  1, -1,  1, -1,  1,  1,  1,
                -1, -1,  1,  1,  1,  1,  1,  1,  1,  1, -1, -1,  1, -1,  1],
               dtype=int64)
```