In [6]:

```
import pandas as pd
```

In [7]:

```
df=pd.read_csv('Total_data.csv')
df.shape
```

Out[7]:

```
(14000, 9)
```

In [8]:

```
import re
pattern = re.compile(u'\s|\n|<[^>]*>|&.*;|\\(.*?\\)', re.S)
for i in range(df.shape[0]):
    df.content[i] = pattern.sub('', df.content[i])
    df.content[i] = df.content[i].replace('”“','')
df.category.unique()
df=df[['category','content','src','time','title']]

import jieba
for i in range(df.shape[0]):
    df.content[i]=jieba.lcut(df.content[i])
#df.to_csv('df_fencihou.csv')
```

```
/Users/apple/anaconda3/lib/python3.6/site-packages/ipykernel_launcher.
py:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (h
ttp://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#re
turning-a-view-versus-a-copy)
  after removing the cwd from sys.path.
/Users/apple/anaconda3/lib/python3.6/site-packages/ipykernel_launcher.
py:5: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (h
ttp://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#re
turning-a-view-versus-a-copy)
  """
Building prefix dict from the default dictionary ...
Loading model from cache /var/folders/y1/hn1g_j8d3dld698c0g4g97qm0000g
n/T/jieba.cache
Loading model cost 0.756 seconds.
Prefix dict has been built succesfully.
```

In [9]:

```
#df=pd.read_csv('df_fencihou.csv')
df_content=df[['category','content']]

stopwords=pd.read_csv("stopwords.txt",index_col=False,sep="\t",quoting=3,names=['sto
def drop_stopwords(contents,stopwords):
    contents_clean = []
    all_words = []
    for line in contents:
        line_clean = []
        for word in line:
            if word in stopwords:
                continue
            line_clean.append(word)
            all_words.append(str(word))
        contents_clean.append(line_clean)
    return contents_clean,all_words

contents = df_content.content.values.tolist()
stopwords = stopwords.stopword.values.tolist()
contents_clean,all_words = drop_stopwords(contents,stopwords)
df_content=pd.DataFrame({'contents_clean':contents_clean,'label':df_content.category

#df_content_pre = df_ex1[df_ex1.isnull().T.any().T]
```

In [11]:

```
df_content=df_content[df_content.label!='video']
```

In [12]:

```
df_content_use = df_content.dropna(axis=0,how='any')

from sklearn import preprocessing
le = preprocessing.LabelEncoder()
le.fit(df_content_use.label.tolist())
df_content.use_label=le.transform(df_content_use.label)

df_content_use_new=pd.DataFrame({'content':df_content_use.contents_clean,'label':df_

#from sklearn.model_selection import train_test_split

#x_train, x_test, y_train, y_test = train_test_split(df_content_use_new['content'].

words_content = []
for line_index in range(df_content_use_new.shape[0]):
    try:
        #x_train[line_index][word_index] = str(x_train[line_index][word_index])
        words_content.append(' '.join(df_content_use_new['content'].values[line_inde
    except:
        print (line_index)

#test_words_content = []
#for line_index in range(len(x_test)):
    #try:
        #x_train[line_index][word_index] = str(x_train[line_index][word_index])
        #test_words_content.append(' '.join(x_test[line_index]))
    #except:
         #print (line_index,word_index)


from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer = TfidfVectorizer(analyzer='word', max_features=4000,  lowercase = False
vectorizer.fit(words_content)

#from sklearn.naive_bayes import MultinomialNB
#classifier = MultinomialNB()
#classifier.fit(vectorizer.transform(words_content), y_train)


#classifier.score(vectorizer.transform(words), y_train)

#classifier.score(vectorizer.transform(test_words_content), y_test)

vectorizer.transform(words_content)
X_content = vectorizer.fit_transform(words_content).toarray()
d_content =pd.DataFrame(X_content)
```

```
/Users/apple/anaconda3/lib/python3.6/site-packages/ipykernel_launcher.
py:6: UserWarning: Pandas doesn't allow columns to be created via a ne
w attribute name - see https://pandas.pydata.org/pandas-docs/stable/in
dexing.html#attribute-access (https://pandas.pydata.org/pandas-docs/st
able/indexing.html#attribute-access)
```

In [*]:

```
#ytrain=pd.DataFrame(df_content_use_new.label)
y=pd.DataFrame(df_content_use_new.label)
y=y.reset_index(drop=True)
d_content_s=pd.concat([d_content,y],axis=1,ignore_index=True)
#d_content_s=d_content_s.drop([-2])
d_content_s
d_content_s.to_csv('s.csv')
```

In [14]:

```
d_content_s.shape
```

Out[14]:

```
(13277, 4001)
```

In [86]:

```
df=pd.read_csv('Total_data.csv')
import re
pattern = re.compile(u'\s|\n|<[^>]*>|&.*;|\\(.*?\\)', re.S)
for i in range(df.shape[0]):
    df.title[i] = pattern.sub('', df.title[i])
    df.title[i] = df.title[i].replace('”“','')
#df.category.unique()
df=df[['category','content','src','time','title']]

import jieba
for i in range(df.shape[0]):
    df.title[i]=jieba.lcut(df.title[i])
#df.to_csv('df_fencihou.csv')
df_ex=df[['category','title']]

stopwords=pd.read_csv("stopwords.txt",index_col=False,sep="\t",quoting=3,names=['sto
def drop_stopwords(contents,stopwords):
    contents_clean = []
    all_words = []
    for line in contents:
        line_clean = []
        for word in line:
            if word in stopwords:
                continue
            line_clean.append(word)
            all_words.append(str(word))
        contents_clean.append(line_clean)
    return contents_clean,all_words

contents = df_ex.title.values.tolist()
stopwords = stopwords.stopword.values.tolist()
contents_clean,all_words = drop_stopwords(contents,stopwords)
df_ex1=pd.DataFrame({'title_clean':contents_clean,'label':df_ex.category})

#df_ex1_pre = df_ex1[df_ex1.isnull().T.any().T]
df_ex1_use = df_ex1.dropna(axis=0,how='any')

from sklearn import preprocessing
le = preprocessing.LabelEncoder()
le.fit(df_ex1_use.label.tolist())
df_ex1.use_label=le.transform(df_ex1_use.label)

df_ex1_use_new=pd.DataFrame({'title':df_ex1_use.title_clean,'label':df_ex1.use_label

#from sklearn.model_selection import train_test_split

#x_train, x_test, y_train, y_test = train_test_split(df_ex1_use_new['title'].values,

words = []
for line_index in range(df_ex1_use_new.shape[0]):
    try:
        #x_train[line_index][word_index] = str(x_train[line_index][word_index])
        words.append(' '.join(df_ex1_use_new['title'].values[line_index]))
    except:
        print (line_index,word_index)

#test_words = []
#for line_index in range(len(x_test)):
    #try:
        #x_train[line_index][word_index] = str(x_train[line_index][word_index])
        #test_words.append(' '.join(x_test[line_index]))
    #except:
         #print (line_index,word_index)

from sklearn.feature_extraction.text import TfidfVectorizer
```

```
/Users/apple/anaconda3/lib/python3.6/site-packages/ipykernel_launcher.
py:5: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (h
ttp://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#re
turning-a-view-versus-a-copy)
  """
/Users/apple/anaconda3/lib/python3.6/site-packages/ipykernel_launcher.
py:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (h
ttp://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#re
turning-a-view-versus-a-copy)

/Users/apple/anaconda3/lib/python3.6/site-packages/ipykernel_launcher.
py:41: UserWarning: Pandas doesn't allow columns to be created via a n
ew attribute name - see https://pandas.pydata.org/pandas-docs/stable/i
ndexing.html#attribute-access (https://pandas.pydata.org/pandas-docs/s
table/indexing.html#attribute-access)
```

In [87]:

```
vectorizer = TfidfVectorizer(analyzer='word',max_features=4000,lowercase = False)
vectorizer.fit(words)

#from sklearn.naive_bayes import MultinomialNB
#classifier = MultinomialNB()
#classifier.fit(vectorizer.transform(words), y_train)


#classifier.score(vectorizer.transform(words), y_train)

#classifier.score(vectorizer.transform(test_words), y_test)
```

Out[87]:

```
TfidfVectorizer(analyzer='word', binary=False, decode_error='strict',
        dtype=<class 'numpy.float64'>, encoding='utf-8', input='conten
t',
        lowercase=False, max_df=1.0, max_features=4000, min_df=1,
        ngram_range=(1, 1), norm='l2', preprocessor=None, smooth_idf=T
rue,
        stop_words=None, strip_accents=None, sublinear_tf=False,
        token_pattern='(?u)\\b\\w\\w+\\b', tokenizer=None, use_idf=Tru
e,
        vocabulary=None)
```

In [89]:

```
vectorizer.transform(words)
X_title = vectorizer.fit_transform(words).toarray()
d_title=pd.DataFrame(X_title)

y=pd.DataFrame(df_ex1_use_new.label)
y=y.reset_index(drop=True)
d_title_s=pd.concat([d_title,y],axis=1,ignore_index=True)
#d_title_s
d_title_s.to_csv('s_title.csv')
```

In [39]:

```
from sklearn.neural_network import MLPRegressor
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import f1_score
import lightgbm as lgb
import numpy as np
from xgboost import XGBClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.naive_bayes import MultinomialNB
from sklearn.neural_network import MLPClassifier
```

In [48]:

```
def choose_models(nstop, X_trntst, Y_trntst):
    rsq_trn = pd.DataFrame(np.zeros((nstop, 4)), columns =['bayes','NN','GB','RF'])
    rsq_tst = pd.DataFrame(np.zeros((nstop, 4)), columns =['bayes','NN','GB','RF'])
    rsq_validate = pd.DataFrame(np.zeros((1, 4)), columns =['bayes','NN','GB','RF']

    for n in range(nstop):
        X_trn, X_tst, Y_trn, Y_tst = train_test_split(X_trntst, Y_trntst, test_size
        X_trn_save = X_trn.copy()
        X_tst_save = X_tst.copy()
        Y_trn_save = Y_trn.copy()
        Y_tst_save = Y_tst.copy()


    #Bayes

        bys = MultinomialNB()
        bys.fit(X_trn,Y_trn)
        predict_bys_trn = bys.predict(X_trn)
        predict_bys_tst = bys.predict(X_tst)
        rsq_trn.loc[n,'bayes'] = accuracy_score(Y_trn,predict_bys_trn)
        rsq_tst.loc[n,'bayes'] = accuracy_score(Y_tst,predict_bys_tst)

        X_trn = X_trn_save
        X_tst = X_tst_save
        Y_trn = Y_trn_save
        Y_tst = Y_tst_save

        #print('b')


    # xgboost
        #xgbt=XGBClassifier(learning_rate=0.2,n_estimators=200,max_depth=8,random_s
        #xgbt.fit(X_trn,Y_trn)
        #predict_xgbt_trn = xgbt.predict(X_trn)
        #predict_xgbt_tst = xgbt.predict(X_tst)
        #rsq_trn.loc[n,'xgbt'] = accuracy_score(Y_trn,predict_xgbt_trn)
        #rsq_tst.loc[n,'xgbt'] = accuracy_score(Y_tst,predict_xgbt_tst)

        #X_trn = X_trn_save
        #X_tst = X_tst_save
        #Y_trn = Y_trn_save
        #Y_tst = Y_tst_save



    # Neural net
        NN = MLPClassifier(hidden_layer_sizes=(4,),activation='relu',solver='adam',
                           learning_rate='adaptive',max_iter=10000,learning_rate_init
        NN.fit(X_trn,Y_trn)
        predict_NN_trn = NN.predict(X_trn)
        predict_NN_tst = NN.predict(X_tst)
        rsq_trn.loc[n,'NN'] = accuracy_score(Y_trn,predict_NN_trn)
        rsq_tst.loc[n,'NN'] = accuracy_score(Y_tst,predict_NN_tst)

        X_trn = X_trn_save
        X_tst = X_tst_save
        Y_trn = Y_trn_save
        Y_trn = Y_trn_save
        #print('n')


    # Gradient boosting tree
        params = {'n_estimators':100, 'max_depth':5, 'min_samples_split':20, 'learn
        GB = GradientBoostingClassifier(**params)
        GB.fit(X_trn,Y_trn)
        predict_GB_trn = GB.predict(X_trn)
        predict_GB_tst = GB.predict(X_tst)
        rsq_trn.loc[n,'GB'] = accuracy_score(Y_trn,predict_GB_trn)
        rsq_tst.loc[n,'GB'] = accuracy_score(Y_tst,predict_GB_tst)

        X_trn = X_trn_save
        X_tst = X_tst_save
        Y_trn = Y_trn_save
        Y_tst = Y_tst_save

        #print('gbt')


    # Random forest
        RF = RandomForestClassifier(n_estimators=200, max_depth=6)
        RF.fit(X_trn,Y_trn)
        predict_RF_trn = RF.predict(X_trn)
        predict_RF_tst = RF.predict(X_tst)
        rsq_trn.loc[n,'RF'] = accuracy_score(Y_trn,predict_RF_trn)
        rsq_tst.loc[n,'RF'] = accuracy_score(Y_tst,predict_RF_tst)

        print(n)

    return rsq_trn,rsq_tst
```

In [5]:

```
d_title_s=pd.read_csv('s_title.csv')
```

In [21]:

```
x=d_title_s.iloc[:,1:4001]
y=d_title_s.iloc[:,4001]
```

In [49]:

```
a,b=choose_models(10, x, y)
```

```
0
1
2
3
4
5
6
7
8
9
```

In [50]:

```
a
```

Out[50]:

	bayes	NN	GB	RF
0	0.867872	0.983723	0.979362	0.354043
1	0.867128	0.982979	0.978936	0.373617
2	0.863191	0.982872	0.978723	0.363617
3	0.866596	0.982340	0.979574	0.326596
4	0.861915	0.982872	0.978085	0.317872
5	0.867234	0.981596	0.979362	0.317234
6	0.870000	0.982872	0.976809	0.334043
7	0.868830	0.984362	0.980000	0.354894
8	0.868191	0.983085	0.979894	0.316383
9	0.868404	0.983085	0.981064	0.334574

In [51]:

```
b
```

Out[51]:

	bayes	NN	GB	RF
0	0.820055	0.798461	0.804418	0.336312
1	0.823033	0.793249	0.803177	0.366344
2	0.823778	0.800199	0.800447	0.350211
3	0.827997	0.796724	0.798709	0.322909
4	0.832961	0.808637	0.811368	0.323405
5	0.820799	0.796972	0.804666	0.314966
6	0.811368	0.803673	0.801440	0.328369
7	0.810375	0.791263	0.793001	0.339787
8	0.818565	0.796972	0.797468	0.309506
9	0.824770	0.801688	0.800447	0.321420

In [53]:

```
print('Bayes:', a['bayes'].mean(), b['bayes'].mean())
print('NN:', a['NN'].mean(), b['NN'].mean())
print('GB:', a['GB'].mean(), b['GB'].mean())
print('RF:', a['RF'].mean(), b['RF'].mean())
```

```
Bayes: 0.8669361702127659 0.8213700670141474
NN: 0.9829787234042552 0.7987838173243981
GB: 0.9791808510638298 0.8015140233308514
RF: 0.3392872340425532 0.3313229089103996
```

In [54]:

```
le.classes_
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call
 last)
<ipython-input-54-373909f5a4bd> in <module>
----> 1 le.classes_

NameError: name 'le' is not defined
```

In [61]:

```
d=d_title_s.iloc[:,4001].tolist()
```

In [64]:

```
dic={}
for k in d:
    if k not in dic:
        dic[k]=1
    else:
        dic[k]+=1
```

In [67]:

```
dic
```

Out[67]:

```
{14: 2830,
 17: 2141,
 5: 1005,
 7: 1641,
 18: 941,
 10: 92,
 20: 152,
 3: 976,
 1: 653,
 12: 590,
 16: 2,
 19: 1,
 11: 2,
 2: 2,
 0: 58,
 13: 1440,
 9: 286,
 4: 135,
 21: 4,
 15: 467,
 8: 9,
 6: 2}
```

In [ ]:

```

```