In [1]:
```
import os
import csv
import pandas as pd
import numpy as np
from IPython.display import display

from bs4 import BeautifulSoup
import re
import nltk
import nltk.data
from nltk.corpus import stopwords
from nltk.stem.porter import PorterStemmer
from nltk.stem import WordNetLemmatizer

from sklearn.feature_extraction.text import CountVectorizer
from sklearn.cross_validation import cross_val_score
from sklearn.ensemble import RandomForestClassifier
import xgboost as xgb
```

In [91]:
```
nltk.download()
```

```
showing info http://www.nltk.org/nltk_data/
```

Out[91]:
```
True
```

In [105]:
```
train = pd.read_csv('C:/Users/SIDDHARTH/Desktop/2.document_set/document_sets.csv',quoting=2, encoding='utf-8')

#train = pd.read_csv('C:/Users/SIDDHARTH/Desktop/3.training_data/training_data.csv', header=0, quoting=3)
```

In [106]:
```
print('Dimension of Labeled Training Data: {}.'.format(train.shape))
print('There are {0} samples and {1} variables in the training data.'.format(train.shape[0], train.shape[1]))
```

```
Dimension of Labeled Training Data: (148665, 2).
There are 148665 samples and 2 variables in the training data.
```

In [107]:
```
train.columns.values
```

Out[107]:
```
array(['Document_Id', 'Text'], dtype=object)
```

In [108]:
```
display(train.head())
```

	Document_Id	Text
0	Document_0	Personal view of author on goals and content...
1	Document_1	The phenomenon of quantum number fractionali...
2	Document_2	Selected problems in heavy quark physics are...
3	Document_3	Prospects on electroweak physics at a future...
4	Document_4	To appear in Encyclopedia of Mathematical Ph...

In [110]:
```
print(train.Text[0])
```

In [111]:
```
train['text_bs'] = train['Text'].apply(lambda x: BeautifulSoup(x, 'html.parse
r'))
```

In [112]:
```
train.text_bs[0].get_text()
```

Out[112]:
```
'  Personal view of author on goals and content of Mathematical Physics.\r\r
\n'
```

In [113]:
```
train['text_letters_only'] = train['text_bs'].apply(lambda x: re.sub(r'[^a-zA-
Z]', ' ', x.get_text()))
```

In [114]:
```
train['text_letters_only'][0]
```

Out[114]:
```
'  Personal view of author on goals and content of Mathematical Physics    '
```

In [115]:
```
train['text_words'] = train['text_letters_only'].apply(lambda x: x.lower().spl
it())
```

In [116]:
```
train['text_words'][0]
```

Out[116]:
```
['personal',
 'view',
 'of',
 'author',
 'on',
 'goals',
 'and',
 'content',
 'of',
 'mathematical',
 'physics']
```

In [117]:
```
set_of_stopwords = set(stopwords.words("english"))
train['text_meaningful_words'] = train['text_words'].apply(lambda x: [w for w
in x if not w in set_of_stopwords])
```

In [118]:
```
num_removed = len(train['text_words'][0]) - len(train['text_meaningful_words'][0])
print('For the first review entry, the number of stop words removed is {0}.'.format(num_removed))
```

```
For the first review entry, the number of stop words removed is 4.
```

In [119]:
```
train['text_cleaned'] = train['text_meaningful_words'].apply(lambda x: ' '.join(x)) # comment if using stemming
```

In [120]:
```
train.drop(['Text', 'text_bs', 'text_letters_only', 'text_words', 'text_meaningful_words'],
           axis=1, inplace=True)
display(train.head())
```

	Document_Id	text_cleaned
0	Document_0	personal view author goals content mathematica...
1	Document_1	phenomenon quantum number fractionalization ex...
2	Document_2	selected problems heavy quark physics discusse...
3	Document_3	prospects electroweak physics future internati...
4	Document_4	appear encyclopedia mathematical physics publi...

In [144]:
```
print(train['text_cleaned'][0])
```

```
personal view author goals content mathematical physics
```

In [122]:
```
vectorizer = CountVectorizer(analyzer="word", preprocessor=None, tokenizer=None, stop_words=None, max_features=500)
```

In [123]:
```
train_data_features = vectorizer.fit_transform(list(train['text_cleaned'].values))
```

In [124]:
```
train_data_features[0]
print('The dimension of train_data_features is
{}.'.format(train_data_features.shape))
```

```
The dimension of train_data_features is (148665, 500).
```

In [129]:
```
# Numpy arrays are easy to work with, so convert the result to an array
train_data_features = train_data_features.toarray()
```

In [130]:
```
def clean_reviews(text, remove_stopwords=False, stem=False):
    """
    to clean review strings
    review: a list of review strings
    remove_stop_words: whether to remove stop words
    output: a list of clean reviews
    """
    # 1. Remove HTML
    reviews_text = list(map(lambda x: BeautifulSoup(x,
'html.parser').get_text(),text))
    #
    # 2. Remove non-letters
    reviews_text = list(map(lambda x: re.sub("[^a-zA-Z]"," ", x),
reviews_text))
    #
    # 3. Convert words to lower case and split them
    words = list(map(lambda x: x.lower().split(), reviews_text))
    #
    # 4. Optionally remove stop words (false by default)
    if remove_stopwords:
        set_of_stopwords = set(stopwords.words("english"))
        meaningful_words = list(map(lambda x: [w for w in x if not w in set_of
_stopwords], words))

    # 5. Optionally stem the words
    if stem:
        porter_stemmer = PorterStemmer()
        wordnet_lemmatizer = WordNetLemmatizer()
        stemmed_words = list(map(lambda x: [porter_stemmer.stem(w) for w in
x], meaningful_words))
        stemmed_words = list(map(lambda x:[wordnet_lemmatizer.lemmatize(w) for
 w in x], stemmed_words))

        # 6. Join the words to a single string
        clean_review = map(lambda x: ' '.join(x), stemmed_words)
    else:
        clean_review = list(map(lambda x: ' '.join(x), meaningful_words))

    return clean_review
```

In [131]:
```
# Read the test data
test = pd.read_csv('C:/Users/SIDDHARTH/Desktop/4.test_data/test_data.csv', hea
der=0, quoting=3)

# Verify that there are 25,000 rows and 2 columns
print('The dimension of test data is {}.'.format(test.shape))

# Get a bag of words for the test set, and convert to a numpy array
clean_test_reviews = clean_reviews(list(test['document_id'].values), remove_st
opwords=True)
test_data_features = vectorizer.transform(clean_test_reviews)
test_data_features = test_data_features.toarray()
```

```
The dimension of test data is (29733, 2).
```

In [136]:
```
dataset_train = pd.read_csv('C:/Users/SIDDHARTH/Desktop/3.training_data/traini
ng_data.csv', header=0, quoting=3)

train_data_features.shape
# Numpy arrays are easy to work with, so convert the result to an array
Y=train_data_features[0]
Y
```

Out[136]:
```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 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, 0, 0, 0, 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, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], dtype=int64)
```

In [ ]:
```
# Initialize a Random Forest classifier with 100 trees
rf_clf = RandomForestClassifier(n_estimators=100, n_jobs=-1, random_state=42)

# Use cross validation to evaluate the performance of Random Forest
#rf_clf_error = 1 - cross_val_score(rf_clf, train_data_features, dataset_train
['document_id'],   cv=5, scoring='accuracy', n_jobs=-1).mean()
#print('Random Forest training error: {:.4}'.format(rf_clf_error))
rf_clf.fit(dataset_train['category'], train_data_features)
```

In [138]:
```
rf_clf.fit(train_data_features, dataset_train['category'])

# Use the random forest to make categoryt label predictions
result =rf_clf.predict(test_data_features)

# Copy the results to a pandas dataframe with an "docunment_id" column an a "c
ategory" column
output = pd.DataFrame(data={"Document_id":test["document_id"], "category":resu
lt})

# Use pandas to write the comma-separated output file
output.to_csv("submissions.csv", index=False, quoting=3)
```