Bag of Words Meets Bags of Popcorn

In this competition we have data with IMDB movie reviews: the texts of the reviews and the marks (whether the review is poritive or negative). The goal is to predict the marks for reviews in test dataset.

The metric to calculate the accuracy of predictions is AUC. One characteristic of the AUC is that it is independent of the fraction of the test population which is class 0 or class 1: this makes the AUC useful for evaluating the performance of classifiers on unbalanced data sets.

In fact I simply take the texts of the reviews, drop stop words (common words, which have no impact), extract word-features and make prediction based on them.

In [ ]:
```
import pandas as pd
from bs4 import BeautifulSoup
import re
import nltk
from nltk.corpus import stopwords
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.ensemble import RandomForestClassifier
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import roc_auc_score
from sklearn.model_selection import train_test_split
```

In [4]:
```
#This downloads data for nltk analysis. Use if necessary.
#nltk.download()
```

In [3]:
```
train = pd.read_csv('../input/labeledTrainData.tsv', header=0, delimiter='\t
test = pd.read_csv('../input/testData.tsv', header=0, delimiter='\t', quotir
```

In [4]:
```
def text_to_words(text):
    """
    Extract words from text.
    """
    text = BeautifulSoup(text, 'lxml').get_text()
    letters = re.sub('[^a-zA-Z]', ' ', text)
    words = letters.lower().split()
    stops = set(stopwords.words('english'))
    meaningful_words = [w for w in words if not w in stops]
    return (' '.join(meaningful_words))
```

In [5]:
```
#Check that it works
print(text_to_words(train['review'][0]))
```

```
stuff going moment mj started listening music watching odd documentary watch
ed wiz watched moonwalker maybe want get certain insight guy thought really
cool eighties maybe make mind whether guilty innocent moonwalker part biogra
phy part feature film remember going see cinema originally released subtle m
essages mj feeling towards press also obvious message drugs bad kay visually
impressive course michael jackson unless remotely like mj anyway going hate
find boring may call mj egotist consenting making movie mj fans would say ma
de fans true really nice actual feature film bit finally starts minutes excl
uding smooth criminal sequence joe pesci convincing psychopathic powerful dr
ug lord wants mj dead bad beyond mj overheard plans nah joe pesci character
ranted wanted people know supplying drugs etc dunno maybe hates mj music lot
s cool things like mj turning car robot whole speed demon sequence also dire
ctor must patience saint came filming kiddy bad sequence usually directors h
ate working one kid let alone whole bunch performing complex dance scene bot
tom line movie people like mj one level another think people stay away try g
ive wholesome message ironically mj bestest buddy movie girl michael jackson
truly one talented people ever grace planet guilty well attention gave subje
ct hmmm well know people different behind closed doors know fact either extr
emely nice stupid guy one sickest liars hope latter
```

In [6]:
```
def clean(a):
    """
    Cleaning data.
    """
    for i in range(0, a.size):
        yield text_to_words(a[i])
```

In [7]:
```
vectorizer = CountVectorizer(analyzer = 'word',
                             tokenizer = None,
                             preprocessor = None,
                             stop_words = None,
                             max_df = 0.5,
                             max_features = 10000)
```

In [8]:
```
train_reviews = list(clean(train['review']))
train_data_features = vectorizer.fit_transform(train_reviews)
test_reviews = list(clean(test['review']))
test_data_features = vectorizer.transform(test_reviews)
```

In [9]:
```
Xtrain, Xtest, ytrain, ytest = train_test_split(train_data_features, train['
```

I tried several models and MultinomialNB proved to be better than most of them.

In [10]:
```
mnb = MultinomialNB(alpha=0.0001)
y_val_m = mnb.fit(Xtrain, ytrain).predict_proba(Xtest)[:,1]
y_pred_m = mnb.fit(train_data_features, train['sentiment']).predict_proba(te

#Accuracy of prediction on validation set
roc_auc_score(ytest, y_val_m)
```

Out[10]:
```
0.9238410855634166
```

In [11]:
```
#Random Forest is even better
forest = RandomForestClassifier(n_estimators=300, criterion = 'gini')
```

```
y_val_f = forest.fit(Xtrain, ytrain).predict_proba(Xtest)[:,1]
y_pred_f = forest.fit(train_data_features, train['sentiment']).predict_proba
roc_auc_score(ytest, y_val_f)
```

Out[11]: 0.92840773397372978

In [12]:
```
#Ensemble of models seems to be the best.
roc_auc_score(ytest, y_val_m + y_val_f)
```

Out[12]: 0.93958275032204619

In [13]:
```
output = pd.DataFrame(data={'id':test['id'], 'sentiment':y_pred_m + y_pred_f

output.to_csv('Bag_of_Words_model.csv', index=False, quoting=3)
```

This competition has already ended, but people still can submit their solutions and see their scores. First two places have score ~0.99, third has ~0.97.

My MultinomialNB model got a score of ~0.9, RandomForestClassifier - 0.93. Ensemble got a score of 0.93366.