In [70]:

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split

import nltk
from nltk.corpus import stopwords
from nltk.classify import SklearnClassifier

from wordcloud import WordCloud,STOPWORDS
import matplotlib.pyplot as plt
%matplotlib inline

from subprocess import check_output

import nltk
import re
from sklearn.feature_extraction.text import TfidfVectorizer
import string
```

In [71]:

```
#  RandomForestClassifier
from sklearn.ensemble import RandomForestClassifier


from sklearn.model_selection import KFold, cross_val_score

from sklearn.metrics import precision_recall_fscore_support as score
from sklearn.model_selection import train_test_split


###  Grid-search
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import precision_recall_fscore_support as score
from sklearn.model_selection import train_test_split

from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
```

In [72]:

```
#columns = ['sentiment', 'text']
data = pd.read_csv('TwitterHate.csv')
```

In [73]:

```
data.shape
```

Out[73]:

```
(31962, 3)
```

In [74]:

```
data.head()
```

Out[74]:

	id	label	tweet
0	1	0	@user when a father is dysfunctional and is s...
1	2	0	@user @user thanks for #lyft credit i can't us...
2	3	0	bihday your majesty
3	4	0	#model i love u take with u all the time in ...
4	5	0	factsguide: society now #motivation

In [75]:

```
data.label.value_counts()
```

Out[75]:

```
0    29720
1     2242
Name: label, dtype: int64
```

In [76]:

```
data = data[['tweet', 'label']]
```

In [77]:

```
data.shape
```

Out[77]:

```
(31962, 2)
```

In [78]:

```
data.head()
```

Out[78]:

	tweet	label
0	**@user when a father is dysfunctional and is s...**	**0**
1	**@user @user thanks for #lyft credit i can't us...**	**0**
2	**bihday your majesty**	**0**
3	**#model i love u take with u all the time in ...**	**0**
4	**factsguide: society now #motivation**	**0**

In [79]:

```
data.isnull().sum()
```

Out[79]:

```
tweet    0
label    0
dtype: int64
```

In [80]:

```
data['tweet'].head(10)
```

Out[80]:

```
0     @user when a father is dysfunctional and is s...
1    @user @user thanks for #lyft credit i can't us...
2                                  bihday your majesty
3    #model   i love u take with u all the time in ...
4               factsguide: society now    #motivation
5    [2/2] huge fan fare and big talking before the...
6     @user camping tomorrow @user @user @user @use...
7    the next school year is the year for exams.ð...
8    we won!!! love the land!!! #allin #cavs #champ...
9     @user @user welcome here !  i'm   it's so #gr...
Name: tweet, dtype: object
```

In [81]:

```
data['tweet'] = data['tweet'].str.replace('#', ' ')
data['tweet'] = data['tweet'].str.replace('amp', ' ')
data['tweet'] = data['tweet'].str.replace('rt', ' ')
```

```
data['tweet'] = data['tweet'].str.replace('http\S+', ' ')
```

In [82]:

```
stopwords = nltk.corpus.stopwords.words('english')
```

In [83]:

```
ps = nltk.PorterStemmer()
```

In [84]:

```
def clean_text(tweet):
    tweet = "".join([word.lower() for word in tweet if word not in string.punctuation])
    tokens = re.split('\W+', tweet)
    tweet = [ps.stem(word) for word in tokens if word not in stopwords]
    return tweet
```

In [85]:

```
tfidf_vect = TfidfVectorizer(analyzer=clean_text,  max_features =5000)
X_tfidf = tfidf_vect.fit_transform(data['tweet'])
```

In [86]:

```
X_tfidf
```

Out[86]:

```
<31962x5000 sparse matrix of type '<class 'numpy.float64'>'
 with 248445 stored elements in Compressed Sparse Row format>
```

In [87]:

```
X_features = pd.DataFrame(X_tfidf.toarray())
```

In [88]:

```
X_features.head()
```

Out[88]:

	0	1	2	3	4	5	6	7	8	9	...	4990	4991	4992	4993	4994	4995	4996	4997	4998	4999
0	0.073464	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.0
1	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	0.000000	0.0	0.0	0.0	0.0	0.0
2	0.112337	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.0
3	0.095789	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.797498	0.0	0.0	0.0	0.0	0.0
4	0.086500	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.0

5 rows × 5000 columns

In [89]:

```
X_features.shape
```

Out[89]:

```
(31962, 5000)
```

In [90]:

```
feauture_name = tfidf_vect.get_feature_names()
pd.DataFrame(X_tfidf.toarray(), columns = feauture_name)
```

Out[90]:

	0	01	02	03	04	05	1	10	100	...	í	ï	ï¼	ï½	ð	ð¾ð	ñ	ó¾	ø	ù

		0	01	02	03	04	05	1	10	100	...	́	̄	ï¾	ï½	ð	ð¾ð	ñ	ó¾	ø	ù
0	0.073464	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.000000	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.0
1	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.000000	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.0
2	0.112337	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.000000	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.0
3	0.095789	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.000000	0.0	0.0	0.797498	0.0	0.0	0.0	0.0	0.0
4	0.086500	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.000000	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
31957	0.047384	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.192045	0.0	0.0	0.901709	0.0	0.0	0.0	0.0	0.0
31958	0.056846	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.000000	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.0
31959	0.067943	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.000000	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.0
31960	0.055602	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.000000	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.0
31961	0.157525	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.000000	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.0

31962 rows × 5000 columns

In [91]:

```
X_train, X_test, y_train, y_test = train_test_split(X_features, data['label'], train_siz
e = 0.8, random_state = 123)
```

In [92]:

```
print(len(X_train))
print(len(X_test))
print(len(y_train))
print(len(y_test))
```

```
25569
6393
25569
6393
```

In [93]:

```
X_train.head()
```

Out[93]:

	0	1	2	3	4	5	6	7	8	9	...	4990	4991	4992	4993	4994	4995	4996	4997	4998	4999
4039	0.068867	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.0
28391	0.057648	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.0
7600	0.058711	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.0
2687	0.018865	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.314118	0.0	0.0	0.0	0.0	0.0
24346	0.076532	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.0

5 rows × 5000 columns

In [94]:

```
rf = RandomForestClassifier(n_estimators=200, n_jobs = -1)
rf_mdl = rf.fit(X_train, y_train)
```

In [95]:

```
y_pred = rf_mdl.predict(X_test)
```

In [96]:

```
y_pred_prob = rf_mdl.predict_proba(X_test)
```

In [97]:

```
y_pred_prob
```

Out[97]:

```
array([[1.   , 0.   ],
       [0.995, 0.005],
       [0.995, 0.005],
       ...,
       [1.   , 0.   ],
       [0.99 , 0.01 ],
       [0.7  , 0.3  ]])
```

In [98]:

```
print(confusion_matrix(y_test, y_pred))
```

```
[[5937   23]
 [ 199  234]]
```

In [99]:

```
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.97      1.00      0.98      5960
           1       0.91      0.54      0.68       433

    accuracy                           0.97      6393
   macro avg       0.94      0.77      0.83      6393
weighted avg       0.96      0.97      0.96      6393
```

In [100]:

```
y_pred_train = rf_mdl.predict(X_train)
```

In [101]:

```
print(confusion_matrix(y_train, y_pred_train))
print(classification_report(y_train, y_pred_train))
```

```
[[23760     0]
 [    6  1803]]
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     23760
           1       1.00      1.00      1.00      1809

    accuracy                           1.00     25569
   macro avg       1.00      1.00      1.00     25569
weighted avg       1.00      1.00      1.00     25569
```

Naive bayes

In [102]:

```
from sklearn.naive_bayes import GaussianNB
mdl = GaussianNB()
mdl_nb = mdl.fit(X_train, y_train)
y_pred = mdl_nb.predict(X_test)
```

Test data metrics

In [103]:

```
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

```
[[4564 1396]
 [ 102  331]]
              precision    recall  f1-score   support

           0       0.98      0.77      0.86      5960
           1       0.19      0.76      0.31       433

    accuracy                           0.77      6393
   macro avg       0.58      0.77      0.58      6393
weighted avg       0.92      0.77      0.82      6393
```

Train data metrics

In [104]:

```
y_pred_train = mdl_nb.predict(X_train)
print(confusion_matrix(y_train, y_pred_train))
print(classification_report(y_train, y_pred_train))
```

```
[[18082  5678]
 [    0  1809]]
              precision    recall  f1-score   support

           0       1.00      0.76      0.86     23760
           1       0.24      1.00      0.39      1809

    accuracy                           0.78     25569
   macro avg       0.62      0.88      0.63     25569
weighted avg       0.95      0.78      0.83     25569
```

In [105]:

```
from sklearn.linear_model import LogisticRegression
lr = LogisticRegression()
lr_mdl = lr.fit(X_train,y_train)
```

In [106]:

```
y_pred = lr_mdl.predict(X_test)
```

In [107]:

```
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

```
[[5948   12]
 [ 273  160]]
              precision    recall  f1-score   support

           0       0.96      1.00      0.98      5960
           1       0.93      0.37      0.53       433

    accuracy                           0.96      6393
   macro avg       0.94      0.68      0.75      6393
weighted avg       0.95      0.96      0.95      6393
```

In [108]:

```
y_pred_train = lr_mdl.predict(X_train)
print(confusion_matrix(y_train, y_pred_train))
print(classification_report(y_train, y_pred_train))
```

```
[[23718    42]
 [ 1063   746]]
              precision    recall  f1-score   support
```

```
           0       0.96      1.00      0.98     23760
           1       0.95      0.41      0.57      1809

    accuracy                           0.96     25569
   macro avg       0.95      0.71      0.78     25569
weighted avg       0.96      0.96      0.95     25569
```

In [109]:

```
from sklearn.model_selection import GridSearchCV, StratifiedKFold
clf = LogisticRegression()
param = {}
clf = GridSearchCV(clf, param, cv = 2, n_jobs = -1, verbose = 1, scoring = "recall")
#grv = GridSearchCV(estimator= lr, cv = StratifiedKFold(5), n_jobs = -1, verbose = 1, sco
ring = "recall")
grv_mdl = clf.fit(X_train,y_train)
```

```
Fitting 2 folds for each of 1 candidates, totalling 2 fits
```

```
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 2 concurrent workers.
[Parallel(n_jobs=-1)]: Done   2 out of   2 | elapsed:    9.3s remaining:    0.0s
[Parallel(n_jobs=-1)]: Done   2 out of   2 | elapsed:    9.3s finished
```

In [110]:

```
y_pred = grv_mdl.predict(X_test)
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

```
[[5948   12]
 [ 273  160]]
              precision    recall  f1-score   support

           0       0.96      1.00      0.98      5960
           1       0.93      0.37      0.53       433

    accuracy                           0.96      6393
   macro avg       0.94      0.68      0.75      6393
weighted avg       0.95      0.96      0.95      6393
```

In [111]:

```
y_pred_train = grv_mdl.predict(X_train)
print(confusion_matrix(y_train, y_pred_train))
print(classification_report(y_train, y_pred_train))
```

```
[[23718    42]
 [ 1063   746]]
              precision    recall  f1-score   support

           0       0.96      1.00      0.98     23760
           1       0.95      0.41      0.57      1809

    accuracy                           0.96     25569
   macro avg       0.95      0.71      0.78     25569
weighted avg       0.96      0.96      0.95     25569
```

In [ ]: