In [0]:
```
from google.colab import drive
drive.mount('/content/drive/','My Drive')
```

```
Go to this URL in a browser: https://accounts.google.com/o/oauth2/auth?client_i
d=947318989803-6bn6qk8qdgf4n4g3pfee6491hc0brc4i.apps.googleusercontent.com&redi
rect_uri=urn%3Aietf%3Awg%3Aoauth%3A2.0%3Aoob&scope=email%20https%3A%2F%2Fwww.go
ogleapis.com%2Fauth%2Fdocs.test%20https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fdri
ve%20https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fdrive.photos.readonly%20https%3
A%2F%2Fwww.googleapis.com%2Fauth%2Fpeopleapi.readonly&response_type=code (http
s://accounts.google.com/o/oauth2/auth?client_id=947318989803-6bn6qk8qdgf4n4g3pf
ee6491hc0brc4i.apps.googleusercontent.com&redirect_uri=urn%3Aietf%3Awg%3Aoauth%
3A2.0%3Aoob&scope=email%20https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fdocs.test%2
0https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fdrive%20https%3A%2F%2Fwww.googleapi
s.com%2Fauth%2Fdrive.photos.readonly%20https%3A%2F%2Fwww.googleapis.com%2Fauth%
2Fpeopleapi.readonly&response_type=code)

Enter your authorization code:
··········
Mounted at /content/drive/
```

1 Data downloaded from kaggle https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge (https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge)

In [0]:
```
%matplotlib inline
%matplotlib notebook
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from scipy.sparse import hstack
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import cross_val_score
from sklearn.metrics import roc_auc_score

```

In [0]:
```
data = pd.read_csv('/content/drive/My Drive/ToxicComments/data/train.csv')
```

In [0]:
```
data.head()
```

Out[4]:

	id	comment_text	toxic	severe_toxic	obscene	threat	insult	iden
0	0000997932d777bf	Explanation\nWhy the edits made under my usern...	0	0	0	0	0	
1	000103f0d9cfb60f	D'aww! He matches this background colour I'm s...	0	0	0	0	0	
2	000113f07ec002fd	Hey man, I'm really not trying to edit war. It...	0	0	0	0	0	
3	0001b41b1c6bb37e	"\nMore\nI can't make any real suggestions on ...	0	0	0	0	0	
4	0001d958c54c6e35	You, sir, are my hero. Any chance you remember...	0	0	0	0	0	

In [0]:
```
data.drop(columns=['id'], axis=1, inplace=True)
```

In [0]:
```
def objectionableOrNot(dataRow):
    if (dataRow['toxic'] == 1 or dataRow['severe_toxic'] == 1 or dataRow['o
       dataRow['insult'] == 1 or dataRow['identity_hate'] == 1) :
        retVal = 1
    else:
        retVal = 0
    return retVal
```

In [0]:
```
list(data.columns[1:7])
```

Out[7]:
```
['toxic', 'severe_toxic', 'obscene', 'threat', 'insult', 'identity_hate']
```

In [0]:
```
data['objectionable'] = data.apply(objectionableOrNot, axis=1)
data.drop(columns=list(data.columns[1:7]), axis=1, inplace= True)
data.head()
```

Out[6]:

	comment_text	objectionable
0	Explanation\nWhy the edits made under my usern...	0
1	D'aww! He matches this background colour I'm s...	0
2	Hey man, I'm really not trying to edit war. It...	0
3	"\nMore\nI can't make any real suggestions on ...	0
4	You, sir, are my hero. Any chance you remember...	0

In [0]:
```
data.objectionable = data.objectionable.astype('category')
```

In [0]:
```
data[data.objectionable == 1].shape
```

Out[48]:
```
(16225, 2)
```

In [0]:
```
data[data.objectionable == 0].shape
```

Out[49]:
```
(143346, 2)
```

In [0]:
```
train_x, test_x, train_y, test_y = train_test_split(data['comment_text'], d
print(f'Train size {train_x.shape}')
print(f'Train size {test_x.shape}')
```

```
Train size (111699,)
Train size (47872,)
```

In [0]:
```
# upsampling
# Indicies of each class' observations
i_class0 = np.array(train_y[train_y == 0].index)
i_class1 = np.array(train_y[train_y == 1].index)

# Number of observations in each class
n_class0 = len(i_class0)
n_class1 = len(i_class1)

# For every observation in class 0, randomly sample from class 1 with repla
i_class1_upsampled = np.random.choice(i_class1, size=n_class0, replace=True
```

In [0]:
```
print(n_class0)
print(n_class1)
```

```
100342
11357
```

In [0]:
```
train_x = pd.concat([train_x.loc[i_class1_upsampled], train_x.loc[i_class0]
```

In [0]:
```
train_y = pd.concat([train_y.loc[i_class1_upsampled], train_y.loc[i_class0]
```

In [0]:
```
tfidf_vectorizer_word = TfidfVectorizer(strip_accents='unicode', analyzer='
train_x_tfidf_word = tfidf_vectorizer_word.fit_transform(train_x)
test_x_tfidf_word = tfidf_vectorizer_word.transform(test_x)
```

In [0]:
```
type(train_x_tfidf_word)
```

Out[15]:
```
scipy.sparse.csr.csr_matrix
```

In [0]:
```
from scipy.sparse.csr import csr_matrix
```

In [0]:
```
tfidf_vectorizer_char = TfidfVectorizer(strip_accents='unicode', stop_words
train_x_tfidf_char = tfidf_vectorizer_char.fit_transform(train_x)
test_x_tfidf_char = tfidf_vectorizer_char.transform(test_x)
```

In [0]:
```
train_tfidf_word_char = hstack([train_x_tfidf_word, train_x_tfidf_char])
test_tfidf_word_char = hstack([test_x_tfidf_word, test_x_tfidf_char])
```

In [0]:
```
train_tfidf_word_char = train_tfidf_word_char.tocsr()
test_tfidf_word_char = test_tfidf_word_char.tocsr()
```

In [0]:
```
from sklearn.naive_bayes import MultinomialNB
```

In [0]:
```
def predict(review, clf):
  char_feature = tfidf_vectorizer_char.transform([review])
  word_feature = tfidf_vectorizer_word.transform([review])
  features = hstack([char_feature, word_feature]).tocsr()
  return clf.predict(features)
```

In [0]:
```
mnb = MultinomialNB(alpha=0.1)
mnb.fit(train_tfidf_word_char, train_y)
```

Out[91]:
```
MultinomialNB(alpha=0.1, class_prior=None, fit_prior=True)
```

In [0]:
```
print(list(np.where(test_y ==1)[0:20]))
```

```
[array([   22,    45,    51, ..., 47856, 47866, 47867])]
```

In [0]:
```
y_pred_proba = mnb.predict_proba(test_tfidf_word_char)
roc_auc_score(test_y, y_pred_proba[0:, 1], average='micro')
```

Out[92]:
```
0.9476282928946549
```

In [0]:
```
for r in np.where(test_y ==1)[0][0:20]:
  print(predict(test_x.iloc[r], mnb))
```

```
[0]
[1]
[1]
[1]
[1]
[0]
[1]
[1]
[1]
[1]
[0]
[1]
[1]
[1]
[1]
[1]
[1]
[1]
[1]
[1]
```

In [0]:
```
import pickle
```

In [0]:
```
pickle.dump(tfidf_vectorizer_word, open('/content/drive/My Drive/ToxicComme
pickle.dump(tfidf_vectorizer_char, open('/content/drive/My Drive/ToxicComme
pickle.dump(mnb, open('/content/drive/My Drive/ToxicComments/lgr.p','wb'))
```