CS314b - Machine Learning | Dr. Kusum Kumari Bharti

# Fighting Hate with Machine Learning
Final Project Report

## Introduction

Hate has been a political tool in recent years where majoritarianism has taken over the leaders and inclusive development remains an eye-wash. The growth of online social media services in the past decade like Facebook, Twitter, Instagram, WhatsApp etc. has provided certain elements yet another platform to spew venom in the minds of the people. Online Hate is deemed to be more personal and creates a long lasting impact given the scale and the manner through which it is spread. The targeted individual or community is also affected in a wide number of ways including trolling, rape threats, misogyny and even death threats resulting in mental torture and in extreme cases, even physical abuse.

## Methodology

It hence becomes imperative for social media platforms and young engineers to use the technology at their disposal to eradicate targeted hate from Social Media Platforms.A number of platforms use manual flagging of content to determine whether the content violates their ill-framed policies. A lot of social media users experience trauma on a daily basis being subject to racial and communal hate on these platforms that are apparently within the content guidelines. It is important that there we realise that there is a very fine demarcation between censorship and fighting hate. Criticism of opinions is not hate but attacking someone for their opinion is. A tool must hence be developed to help a user ‘set their boundaries’ and create a custom classifier for each user. This way, we would not censor the ‘hateful’ content but will hide it from the concerned user so as to create a safe space for her/him. At the same time, we would report the flagged content to the platform so that it may take action as required by law. This way, we are making social media platforms a safe space for everyone without censoring content on the platform.

## Literature

We have reviewed two papers so on the subject for our project.

1. Implementation Of Naive Bayes Classifier Algorithm On Social Media (Twitter) To The Teaching Of Indonesian Hate Speech ( 2017 International Conference on Sustainable Information Engineering and Technology (SIET) )
2. Hate Speech on Twitter - A Pragmatic Approach to Collect Hateful and Offensive Expressions and Perform Hate Speech Detection

While both of these papers are within the domain of our machine learning study, we observe that the first paper uses a simpler 'Naive Bayes Classifier' approach to machine learning rather than the second one that uses a 'Unigram' approach that requires a lot of annotation to be done before we start using the datasets for classification.

## Course of Action

Since our methodology aims to create a custom model suited to our users, we should go with the Naive Bayes Model as it is easy to calculate and store for reuse.

We plan on building a classic Naive Bayes Classifier for each of our users to start with. As the user uses our tool, she/he will flag the offensive content that will shape his classifier more to his suiting. This way each classifier will be unique and will cater to the 'personal boundaries' of each user.

Additionally, we will be working with Twitter since it is not possible to search for posts with the Facebook Graph API which is ironically very rigid as opposed to the content policy guidelines of the social media platform.

We will be using the Twitter API to focus on making Twitter a safe-space for the users.

## Our Progress

Before we ventured into making a smart classifier to address individual needs of each user, we had to understand and dive deeper into the kind of data that we're dealing with. It was more important that we focus on the needs of the Indian subcontinent given the recent spike in hate crimes under fascist regimes.

### The Data

We scraped a total of **3189 tweets** from the past year that had rather controversial hashtags through the Twitter API. We then classified these on the basis of their content into three categories.

1. Decent (Purely opinionated tweets with no hate or abuse)
2. Abusive (Tweets with abuse towards the topic using profanity but not inciting hate)
3. Hateful (Tweets with directed hate towards communities or inciting violence)

Our dataset was fairly balanced between decent and hateful classes, but the hateful class had an edge given the nature of hashtags we chose and the moral degradation of Indian Twitter trolls over the past few years. Our final numbers were as follows -

| **No.** | **Class of Tweets** | **Number of Tweets in Dataset** |
|---|---|---|
| 1 | Decent | 1121 |
| 2 | Abusive | 303 |
| 3 | Hateful | 1765 |
| **-** | **Total** | **3189** |

With this data, we were now in position to create a **base model** for our users. This model will then be subjected to their own preferences and evolve into one that understands their 'boundaries'.

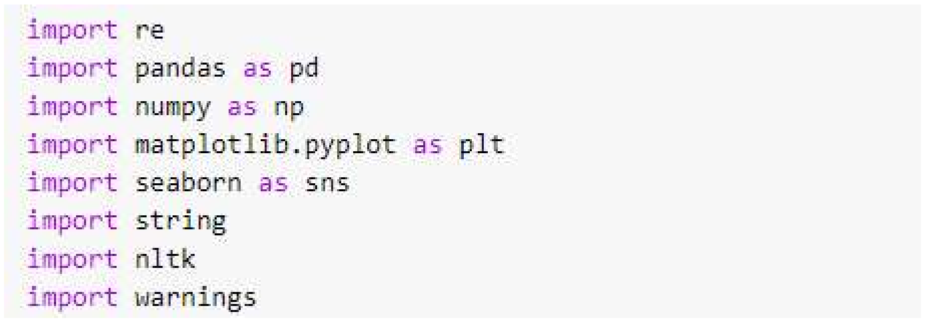
## Data Preprocessing

Like always, before we get to the actual nitty-gritty, we need to prepare our data before we can do any of the fun stuff.

We have two files, a train file and a test file. Within the train file, we have a random assortment of various tweets. There are three features, a unique id, a label, and the actual tweet text. The label is ‘0’ if the tweet is decent, the label is ‘1’ if the tweet is abusive and the label is ‘2’ if the tweet is outright hateful

### Firing up Libraries

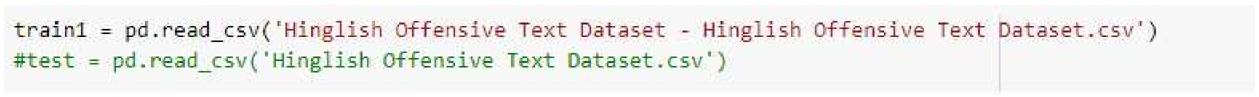
We used the following libraries to help us with our task of preparing the data.



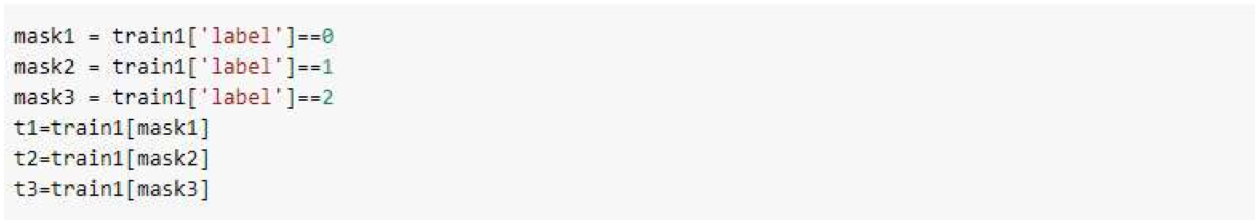
```
import re
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import string
import nltk
import warnings
```

### Create Test Train and Validation Sets

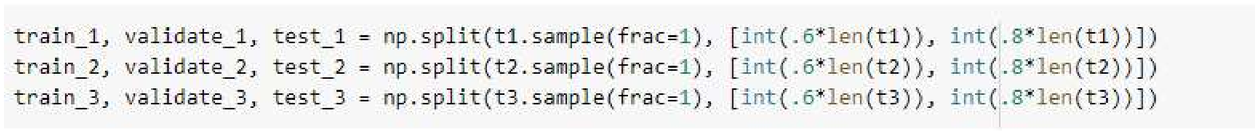
Next up, we use pandas to read our data and create test, train and validation sets



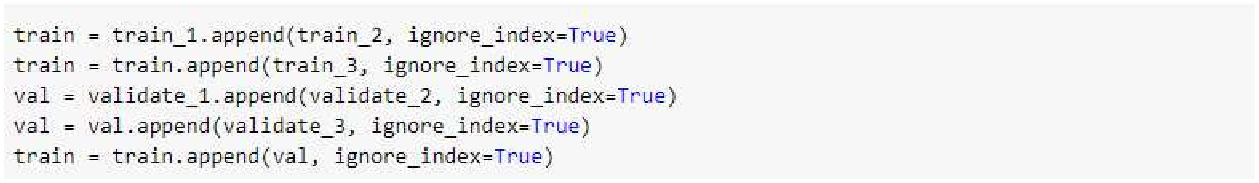
```
train1 = pd.read_csv('Hinglish Offensive Text Dataset - Hinglish Offensive Text Dataset.csv')
#test = pd.read_csv('Hinglish Offensive Text Dataset.csv')
```



```
mask1 = train1['label']==0
mask2 = train1['label']==1
mask3 = train1['label']==2
t1=train1[mask1]
t2=train1[mask2]
t3=train1[mask3]
```



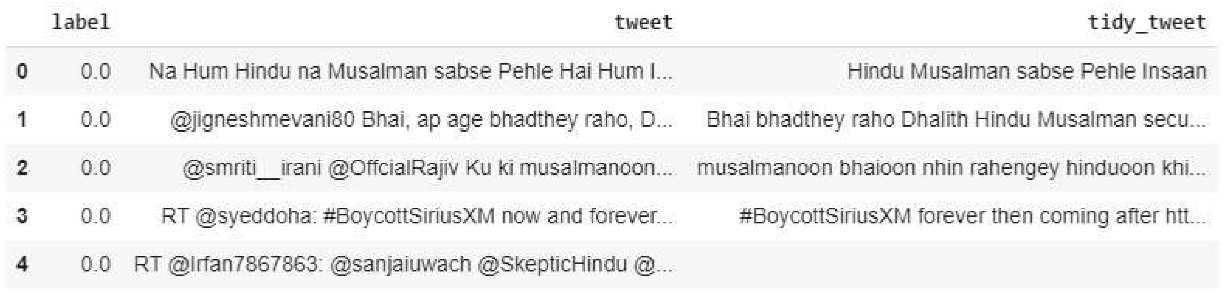
```
train_1, validate_1, test_1 = np.split(t1.sample(frac=1), [int(.6*len(t1)), int(.8*len(t1))])
train_2, validate_2, test_2 = np.split(t2.sample(frac=1), [int(.6*len(t2)), int(.8*len(t2))])
train_3, validate_3, test_3 = np.split(t3.sample(frac=1), [int(.6*len(t3)), int(.8*len(t3))])
```



```
train = train_1.append(train_2, ignore_index=True)
train = train.append(train_3, ignore_index=True)
val = validate_1.append(validate_2, ignore_index=True)
val = val.append(validate_3, ignore_index=True)
train = train.append(val, ignore_index=True)
```

## Tidying up

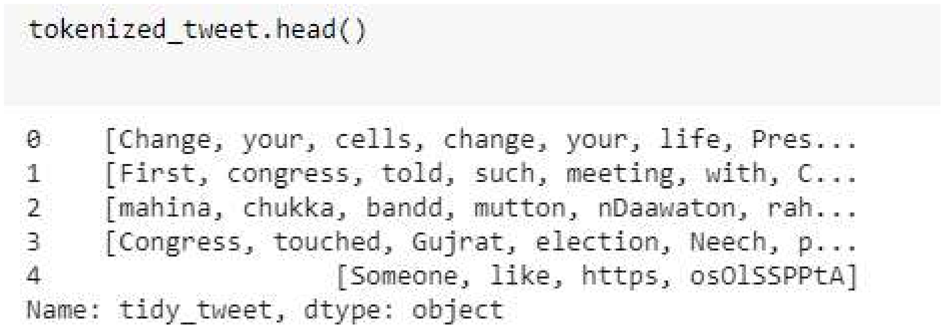
After having separate sets, we decided to tidy our data by removing unnecessary elements like user tags and small words that don't matter much.



| | label | tweet | tidy_tweet |
|---|---|---|---|
| **0** | 0.0 | Na Hum Hindu na Musalman sabse Pehle Hai Hum I... | Hindu Musalman sabse Pehle Insaan |
| **1** | 0.0 | @jigneshmevani80 Bhai, ap age bhadthey raho, D... | Bhai bhadthey raho Dhalith Hindu Musalman secu... |
| **2** | 0.0 | @smriti__irani @OffcialRajiv Ku ki musalmanoon... | musalmanoon bhaioon nhin rahengey hinduoon khi... |
| **3** | 0.0 | RT @syeddoha: #BoycottSiriusXM now and forever... | #BoycottSiriusXM forever then coming after htt... |
| **4** | 0.0 | RT @Irfan7867863: @sanjaiuwach @SkepticHindu @... | |

## Tokenizing Tweets

Then we split the tweets to head into the bag of words approach.

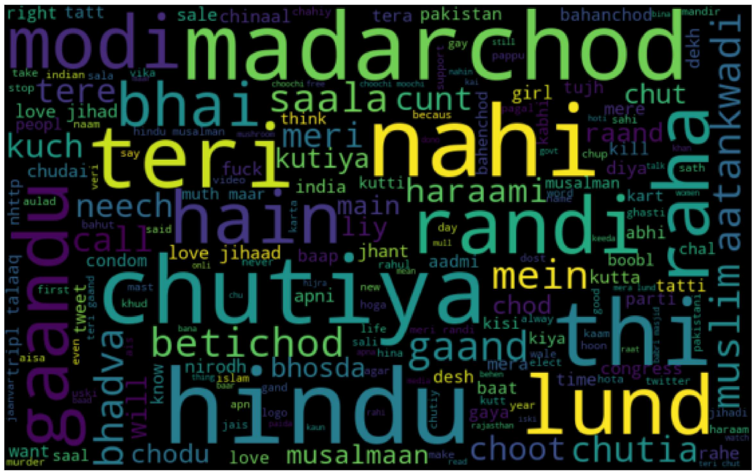


```
tokenized_tweet.head()
```

```
0    [Change, your, cells, change, your, life, Pres...
1    [First, congress, told, such, meeting, with, C...
2    [mahina, chukka, bandd, mutton, nDaawaton, rah...
3    [Congress, touched, Gujrat, election, Neech, p...
4                  [Someone, like, https, osOlSSPPtA]
Name: tidy_tweet, dtype: object
```

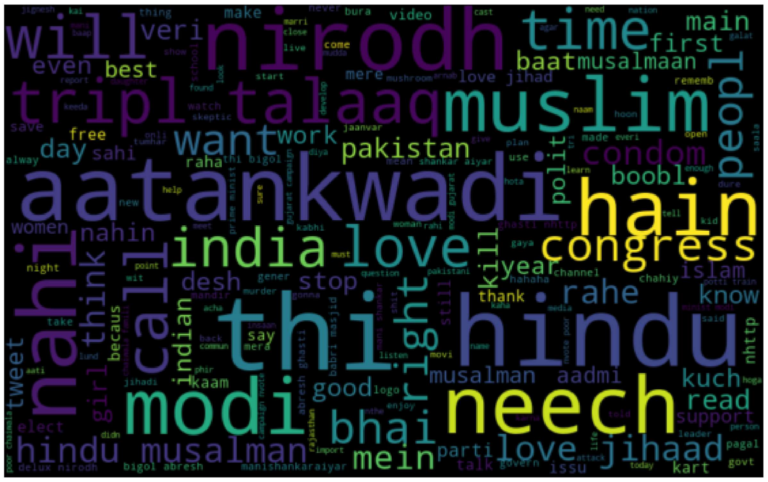
## Data Visualization

Visualization techniques help us understand the nature of our data and more importantly the basis of our classification. We will first look at the word cloud for all the processed tweets in our database.



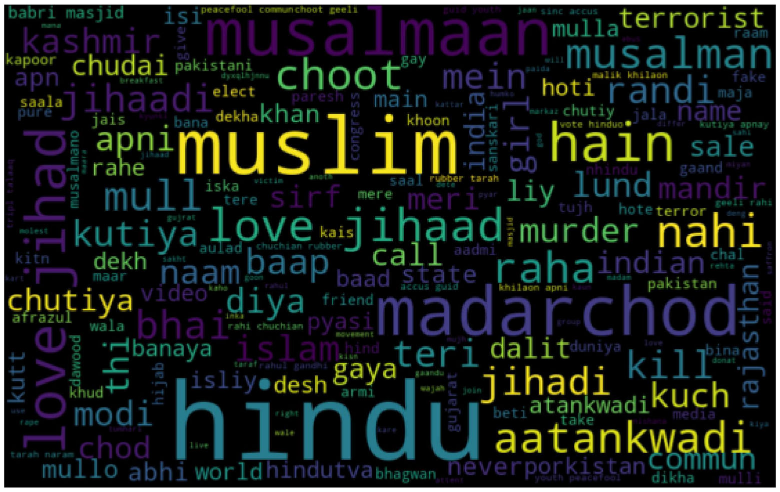
As mentioned before, clearly there is a lot of abuse and profanity owing to the edge of hateful class in the database. Perhaps a class-wise word cloud will help us better understand our data.

### The Decent Class



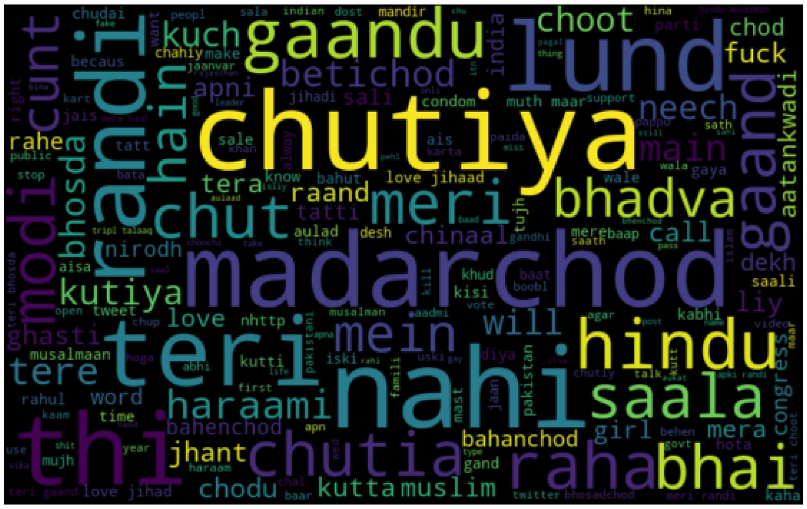
The tweets in decent class are opinionated in nature about the topic but refrain from using any abuse or profanity. These tweets can be classified as criticism in general.

## The Abusive Class



The tweets in the abusive class use profane language in expressing their views on the topic but do not necessarily incite or promote hate against the group or the ideology. These words can cause hurt and are therefore to be curtailed from sensitive audiences.

## The Hate Class



These are tweets that use extremely profane language to direct hate and violence against a community or ideology. These tweets often do not have any opinion but simply target the individual or idea by means of verbal violence.

## Building our Model

Now that we understand our data better and have a clear understanding of the classification that we aim to do, we go ahead with model building.

### The Multiclass Logistic Regression Model

We started with the basics and went ahead with a Logistic Regression Model for our data.

```
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import f1_score
from sklearn import linear_model
from sklearn import metrics
```

```
lreg = LogisticRegression()
lreg.fit(xtrain_bow, ytrain)
```

```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='auto', n_jobs=None, penalty='l2',
                   random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False)
```

We were successful in training the model but what mattered is the accuracy and thereby model evaluation. We evaluated the model against our testing set.

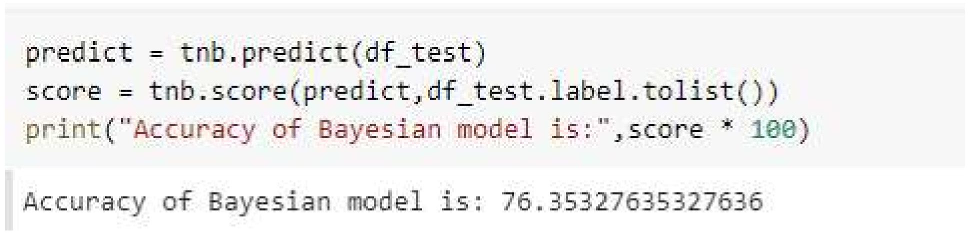
```
prediction = lreg.predict_proba(xvalid_bow)
prediction_int = prediction[:,1] >= 0.3
prediction_int = prediction_int.astype(np.int)
print("Accuracy of Logistic Regression model is:",
metrics.accuracy_score(yvalid, prediction_int)*100)
```

```
Accuracy of Logistic Regression model is: 40.130718954248366
```

Clearly, 40% accuracy is not enough and although it is a good starting point, we need a better base model for our data. Which is why we explored our next approach.

### The Naive Bayesian Model

This model returned the probability of the prediction instead of a black and white classification. Given the considerable difference in our class, this approach turned out to be better than the last one.



```
predict = tnb.predict(df_test)
score = tnb.score(predict,df_test.label.tolist())
print("Accuracy of Bayesian model is:",score * 100)
```

```
Accuracy of Bayesian model is: 76.35327635327636
```

## The Way Forward

A 76% Accuracy is a decent-fit for our base model and we would take this one forward to the customisation part that unfortunately could not be done completely due to the unforeseen circumstances that the world has seen in the past few months.

We had planned on building a Google Chrome Add-on for users to implement this Machine Learning Model and benefit real users. This project has however caught our interest and we will continue our work on the same beyond the scope of this course and evaluation towards the greater good. We have also secured access to the Twitter Developer Dashboard to further our initiative after our application for API access was accepted by Twitter.

**Team**

**Ajwad Shaikh (2017333) | Manjot Singh (2017330) | Kanishk Goyal (2017322)**