TOXIC COMMENTS CLASSIFICATION

Problem statement : Here we are given the dataset which contains the comments in text form which are classified into toxic, severe toxic, hate, obscene, insult and identity hate. And we need to classify the comments into these classes for given test data.

So the given train dataset contains the comments made by people as comment_text with their ids and all the six classes with values 0 and 1.

"0" means it doesn't belong to that class while "1" means it belongs to that class.

Contents of report:

- Visualization
- Preparing the data
- Pre-processing the text
- Training and Modeling

Error metric used is ROC as instructed in the challenge.

Two models have been used : one is Naïve bayes and another is XGBoost.

Naïve bayes results were not looking satisfactory so have also used XGBoost.

Results of both the models are included.

I have also tried solving by logistic regression and random forest but was not able to run on my machine due to configuration issues.

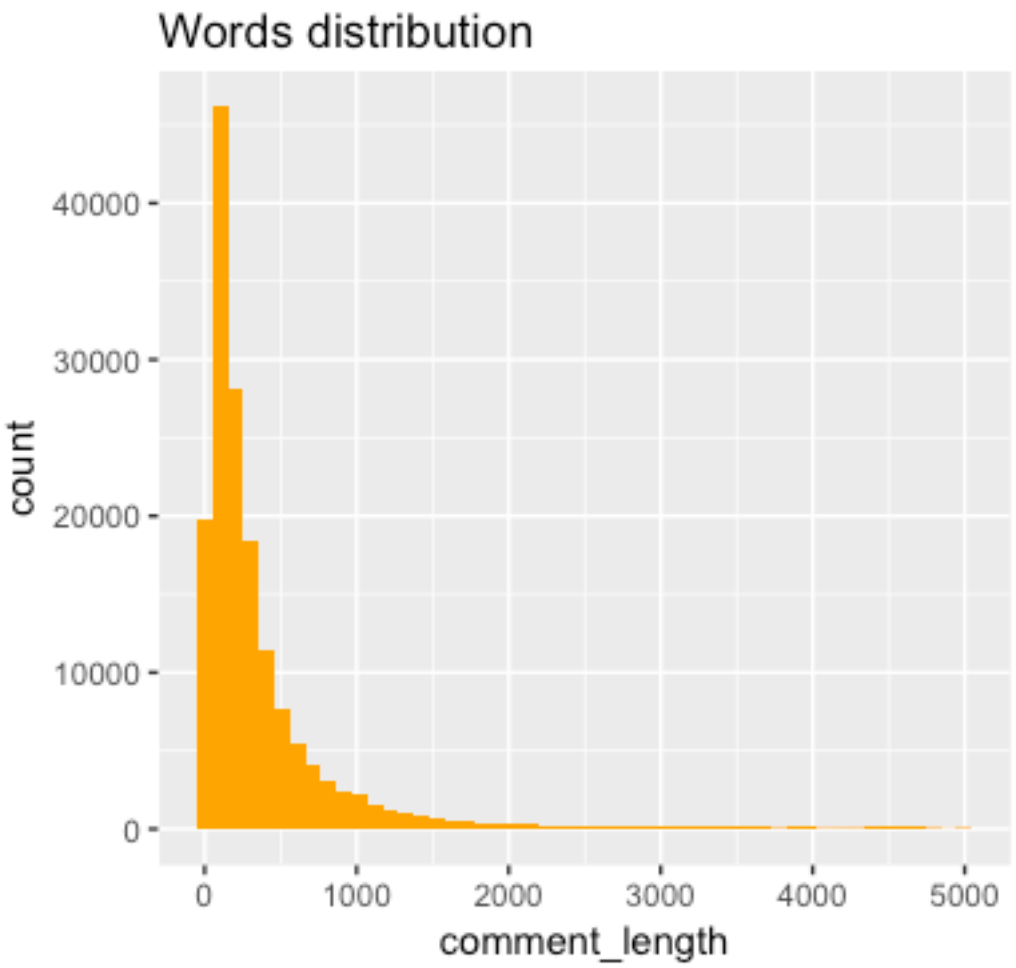
Everytime I run logistic regression or random forest my machine used to get crash and never respond.

Same was the reason I chose XGBoost for my 1st project because it is very fast.

Visualization

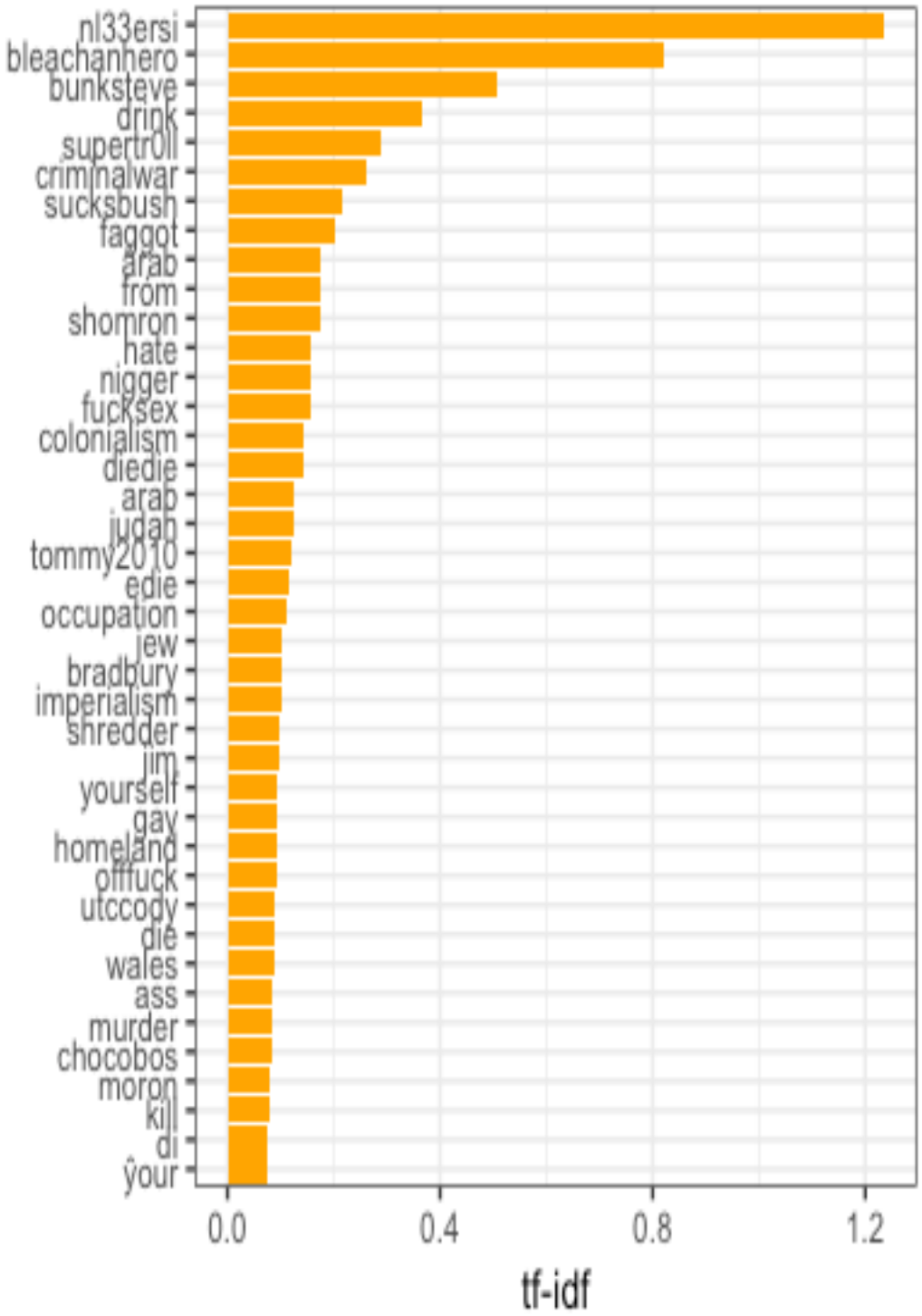
- First we load the datasets given that is test , train and sample submission.
- Then we format the comments in the train dataset like removing special characters , removing extra spaces .
- Then we add an extra variable to the test and train data set as comment length which tells ous the number of strings in a comment.

Word distribution graph :

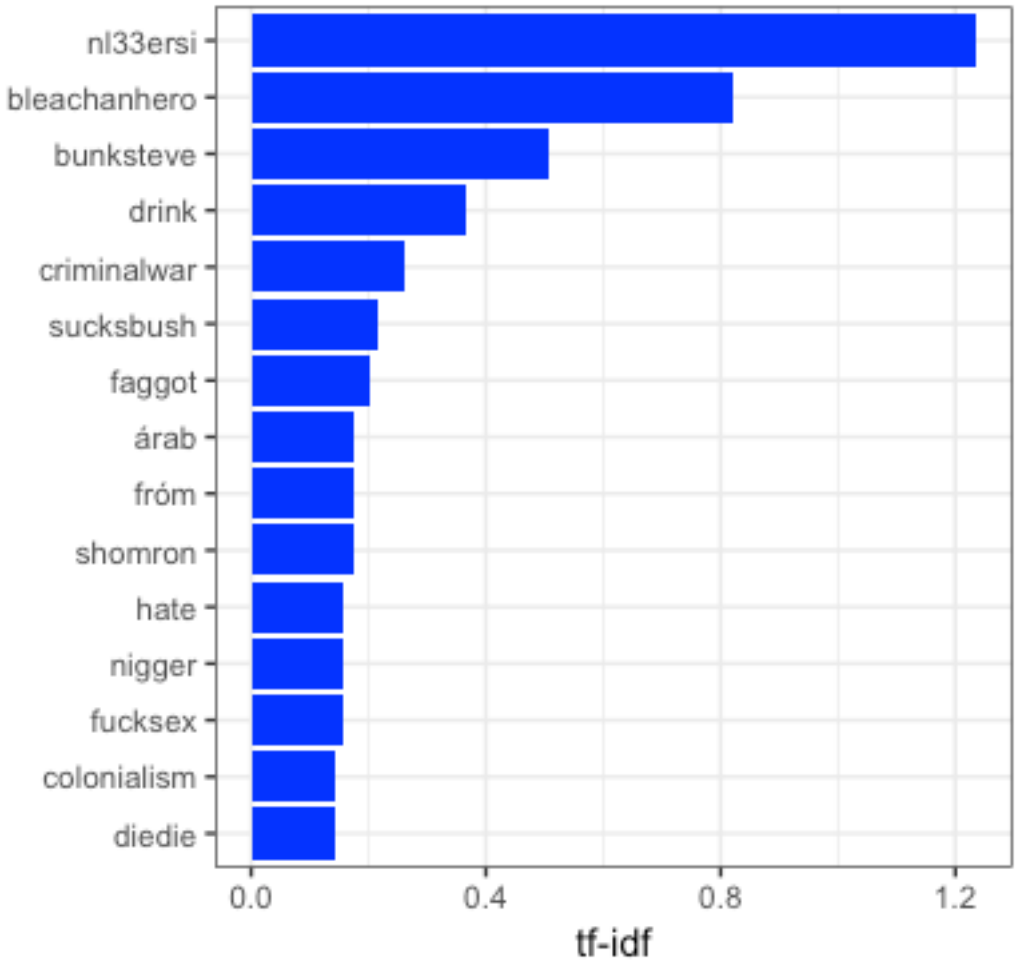


Word cloud of top 100 words which will show up according to the tf-idf i.e higher the tf-idf score the bigger it appears.

Plotting the top 40 words according to tf-idf.



Top 15 words in toxic category according to tf-idf



same way we can plot the top 15 words of other categories by just replacing the "category_name = 1" in the code. For e.g severe_toxic = 1

Preparing data:

- After formatting the comments we will extract the words from comments and counting it in each category by using unnest_tokens function.

- Now we find the total number of words in each class by grouping them using groupby().
- After counting the total words we find that there are total 41 observations so we have assigned number 1 to 41 as different categories and storing them in object category.
- Now we apply the left join on extracted words and total number of words table. And storing it in object words_train.
- Now we bind the tf-idf of word_train using bind_tf_idf(), which takes words, category and n as an argument.
- Where words means column containing terms as string or symbol category means column containing the document ids as string or symbol , n means the columns containing document term counts as string or symbols.
- Now arranging the same with descending order of tf-idf .

Pre-processing the text:

- Here we replace all the alphanumeric characters by a space .
- Later we build a corpus of the train dataset by converting thr words into lower case , removing numbers , removing punctuations , striping whitespaces , removing stopwords , stemming the document.
- Now we build the document term matrix using the corpus asnd removing the sparse terms.
- Convert that document term matrix into a dataframe.
- We did all the above steps on test data also.
- Later we make the columns same of test and train using intersect() function.
- Splitting the data into test and train.

Naïve Bayes modeling:

- Now we train the naïve bayes model using the train dataset by using single class at a time. That means first we will train for toxic then severe_toxic and so on for all the 6 classes.
- Then we predict the classes one by one and add every class to the dataframe submission that we loaded earlier at the start of code.

- This is how whole submission file will get ready and then we saved it as a csv file.
- But by looking the submission file it wasn't looking very much convincing.
- So we will try to classify using XGBoost algorithm.

XGBoost modeling:

- We will do it in the same way by predicting only one class at a time.
- In train control we are setting classprobs as true because we want prediction as probabilities. And using summary as twoclasssummary which will calculate specificity , sensitivity etc.
- Setting up the grid by passing the arguments like eta which is the learning rate, nrounds which says the number of rounds we require., maxdepth which is used for tree pruning etc.
- Later we prepare the submission file by predicting each class one by one for test data. And save it as csv file.