IJCSE International Journal of Computer Sciences and Engineering Open Access
Research Paper Vol. 8, Issue.5, May 2020 E-ISSN: 2347-2693

Detection of Cyberbullying using Voting Classifier

R. Kaur[1*], M.S. Sagar[2]

[1,2]Dept. of Computer Science and Engineering, Mahatma Gandhi Institute of Technology, Hyderabad, India

[*]Corresponding Author: rashikaur6@gmail.com, Tel.: +91-95505-82250

DOI: https://doi.org/10.26438/ijcse/v8i5.5360 | Available online at: www.ijcseonline.org

Received: 03/May/2020, Accepted: 18/May/2020, Published: 31/May/2020

***Abstract* —** The advent of social media has changed the ways of human communication. It has brought people around the world closer to each other. Despite its innumerable benefits, social media is considered to be one of the harmful elements of society. Cyberbullying and online harassment are the most common negative effects of social media. Cyberbullying is a way of bullying someone with the use of technology and it can take place through many forms such as SMS, Apps, online gaming, social networking sites online forums, etc. The project aims at detecting cyberbullying content based on textual features. The system detects various language patterns often used by bullies. This is accomplished using machine learning. The proposed system uses voting classifier to classify the input text as 'Bullying' or 'Non-Bullying'. It also compares the accuracies of various classifiers and introduces a framework of supervised machine learning to detect cyberbullying in textual data. It is observed that a voting classifier i.e. a combination of the Logistic Regression, Random Forest, Support Vector Machine, SGD classifier gives the highest accuracy and precision i.e. 74% and 77% respectively. This trained model is deployed on a webpage which makes the system user intuitive and user-friendly.

***Keywords*—** Cyberbullying, Machine Learning, Classification, Voting classifier, Social Media

I. INTRODUCTION

Social networking websites play a key role in communication these days. It provides an easy and reliable way to connect with our friends, family and peers. The rise of the internet and usage of social media has led to the emergence of a new form of bullying that doesn't occur in the classroom, home or neighborhood, but takes place online and is carried out on the internet. This modern form of bullying is known as cyberbullying. It is the use of any form of technology like SMS, online chat groups, online gaming, social networking, etc, to intentionally threaten or domineer someone. Cyberbullying is widely increasing in India. According to the study, 'Online Study and Internet Addiction', which released in 2020, 22.4% of people, aged between 13-18 years, who used social media for more than three hours a day, were at risk of being a victim of cyberbullying. Online harassment and cyberbullying have become a serious social threat in our society. To curb cyberbullying, we need to detect instances of cyberbullying by creating a speech model based on historic data available.

The proposed system introduces a simple and user-friendly website to detect whether a post contains cyberbullying data or not. Machine learning is used to predict the label of a given text. The system uses supervised learning algorithms to predict the class label of the text i.e. 'Bullying' or 'Non-Bullying'. In supervised learning, the data used to train the algorithm is already labeled with correct answers. Initially, various simple classifiers like SVM, Naïve Bayes algorithm, Random Forest classifier etc, are applied to the given dataset to predict the class labels of the text. Then, the accuracy of voting classifiers is checked for the given dataset. The voting classifier is a machine learning algorithm that trains on an ensemble of many models and predicts an output class based on the highest probability of chosen class as the output. It is observed that the voting classifier gives the highest accuracy in detecting cyberbullying.

This paper also intends to identify the most informative features in texts containing cyberbullying. This is achieved by using the Bag of Words (BoW) concept along with the Naïve Bayes classifier.

The rest of the paper is organized as follows, Section II contains the problem statement of the paper, Section III discusses the literature survey of cyberbullying detection, Section IV contains the data collection information, Section V contains the procedure followed for data preprocessing, Section VI explains steps followed in developing the models, Section VII describes the results obtained, Section VIII concludes the paper and Section IX describes its future scope.

II. PROBLEM STATEMENT

Cyberbullying is one of the major issues faced by our society today. Many people nowadays say things online which they wouldn't say to a person directly. The Internet provides a false sense of security to people, allowing them to feel as though they can say anything without any repercussions. Anonymity online gives users the ability to say whatever they want without considering its consequences. Many individuals suffer psychological problems such as depression, sleeplessness, lowered self-esteem and lack of motivation to live due to cyberbullying. The false world of the web makes it difficult to detect and stop cyberbullying. This project aims to detect cyberbullying in textual data using various supervised machine learning algorithms. It also detects the most common features or words used in texts containing bullying.

III. LITERATURE SURVEY

Table 1: Literature Survey for cyber bullying detection using machine learning

S.No	Author and Year	Methodology
[1]	Cheng et al. 2019	Proposed XBully, a framework for detecting cyberbullying, that initially re articulates multi-modal data from social network and then targets to train node-embedding illustrations upon it.
[2]	Rafiq et al 2018	Developed a cyberbullying detection system for media-based social networks, consisting of a dynamic priority scheduler, a novel incremental classifier, and an initial predictor.
[3]	Zhao et al 2016	A learning method was proposed for detection of cyberbullying by concatenating bullying, latent semantic and BoW features together.
[4]	Al-garadi et al. 2016	Suggested a feature-based classifier for detecting cyberbullying using supervised machine learning in the Twitter media.
[5]	Mangaonkar et al. 2015	Proposed collaborative paradigm that used different machine learning techniques for classification of bully or non-bully data.
[6]	Nahar et al. 2014	Proposed semi-supervised learning in the session-based framework that incorporates an ensemble of one-class classifiers.
[7]	Reynolds et al. 2011	Supervised machine learning approach in conjunction with labelled data was used to learn the system to identify bullying content.
[8]	Dinakar et al. 2011	Used supervised machine learning approach in which binary & multiclass Classifiers classify bullying sensitive topics.

IV. DATA COLLECTION

The datasets used are downloaded from Kaggle. Various datasets are combined to improve the accuracy of the model. The major part of the dataset contains tweets from the social networking site 'Twitter'. 'Twitter' is an American microblogging and social networking service on which users post and interact with messages known as "tweets". The dataset consists of two attributes i.e. 'Tweet' and 'Text Label'. The attribute 'Text Label' takes two values i.e. 'Bullying' and 'Non-Bullying'. Only 28% of the data is labeled as 'Bullying'. To improve the precision and accuracy of the model, another dataset consisting of bad or toxic words is combined with the previous dataset. The final dataset consists of 10,344 tweets. 33% of the tweets in the final dataset are labeled as 'Bullying'.

	A	B
1	Tweet	Text Label
2	yeah I got 2 backups for all that. I just hate when that happen. I been strugglin for a week now...handle that tho	Non-Bullying
3	I hate using my BB but love my iPhone. Haven't tried the new BB. My BB is provided by my corp. I don't get to pick which model	Non-Bullying
4	Get fucking real dude.	Bullying
5	She is as dirty as they come and that crook Rengel the Dems are so fucking corrupt it's a joke. Make Republicans look like ...	Bullying
6	why did you fuck it up. I could do it all day too. Let's do it when you have an hour. Ping me later to sched writing a book here.	Bullying
7	Dude they dont finish enclosing the fucking showers. I hate half assed jobs. Whats the reasononing behind it? Makes no sense.	Bullying
8	WTF are you talking about Men? No men thats not a menage that's just gay.	Bullying
9	Ill save you the trouble sister. Here comes a big ol fuck France block coming your way here on the twitter.	Bullying
10	Im dead serious.Real athletes never cheat don't even have the appearance of at his level. Fuck him dude seriously I think he did	Bullying
11	wow lol sounds like a lot of piss then hehehe	Non-Bullying
12	not a damn thang..the typical rap beef. one person worrying about what the next is doing and the other respondin etc etc	Non-Bullying

Figure 1: Snippet of the final dataset

V. DATA PREPROCESSING

The data that was collected for solving the problem must be transformed into a format suitable for machine learning. We need to make sure that the data is free of inconsistencies and all the data points are presented using the same logic. This improves the model performance and the quality of received insights from the data.

Textual data is a form of unstructured data. This could reduce the accuracy of the classification algorithms used. So, before applying machine learning algorithms to the textual data, we clean the text. The raw textual data is cleaned using the following steps:

A. Removing Unwanted Characters

Unwanted characters constitute of characters that might not be a part of a language. Data taken from HTML/XML sources may contain various unwanted characters like HTML tags, entities, and attributes. The unwanted textual data can be cleaned using regular expressions.

B. Tokenization and Capitalization/ De-capitalization

The process of breaking down a given sentence into words is called tokenization. The textual data must be completely capitalized or de-capitalized to avoid changes in the result due to different case types.

C. Removing Stopwords

The words used habitually in a language are known as stopwords. These words occur time and again in the texts, which makes them lose their semantic meaning. They are usually connecting words like 'of', 'are', 'it', etc.

D. Lemmatizing/ Stemming

In any language, the 'root' word is a part of the word that provides the basic meaning of the word. Stemming or lemmatizing is the process of converting the words into their 'root' forms.

VI. DEVELOPING THE MODEL

A. *Approaches*

Bag of Words (BoW)

To extract various features from textual data, the Bag of Words approach can be used for modelling the machine learning algorithm. The Bag of Words model is concerned with the vocabulary of the words used and their frequencies. This model doesn't focus on the structure or order of the words in a sentence. It focuses on the various words that occur in the textual data. It is based on the thought that similar documents contain similar content. First, all the unique words in the data are extracted. Using this list of words, document vectors are created. The words in each document are scored. Generally, if a word is present in the document, it is marked as 1. If it is absent, it is marked as 0. When a new input document is given, it is scored using the same process as above. This score is used to classify the data.

In this project, this approach is followed to retrieve the most informative features in the dataset to detect cyberbullying.

Count Vectorizer

Count Vectorizer tokenizes the documents and builds a vocabulary of known words. Once a new document is given, it counts the frequency of the tokens that appear in the document.

Example sentence: "The weather was wonderful today and I went outside to enjoy the beautiful and sunny weather." You can tell from the output below that the words "the", "weather", "and "and" appeared twice while other words appeared once. That is what Count Vectorization accomplishes. This project follows the count vectorizer approach to predict the class labels of the new input text given.

B. *Algorithms*

The project compares the performances of various algorithms. The algorithms used are:

Logistic regression (LR) uses a sigmoid function to predict the class labels of the given data. It performs classification based on the probability that a data point belongs to a particular class. The logistic regression classifier aims at maximizing the likelihood function of the model.

Random Forests (RF): This classifier uses multiple decision trees that work together as an ensemble classifier. Each decision tree predicts a class label for the given input. The class label predicted by the majority of the decision trees is considered as the final result.

Support Vector Machines (SVM): It uses a hyperplane in an N-dimensional space to classify the various data points. A kernel function is used to decide the shape of the hyperplane. Support vector machines can solve problems that can't be solved using linear boundaries.

Stochastic Gradient Descent (SGD): It works by optimizing a specific objective function by using the iterative method. It is based on the Gradient Descent optimization technique which is a convex function.

Naive Bayes: It is a classification algorithm based on the Bayes theorem. It is called "naïve" as it assumes that each feature of the dataset is conditionally independent of each other. This assumption is made to simplify the calculation of the probabilities.

Decision Tree: A decision tree is one of the simplest yet powerful classification algorithms. Each internal node represents an attribute of the dataset, and the leaf nodes represent the final outcomes.

AdaBoost: The Adaboost classifier is based on the boosting method. AdaBoost initially fits a classifier on the given dataset. Multiple copies of the same classifier are then fit on the same dataset to adjust the weights of incorrectly classified instances.

Ensemble/ Voting Classifier: Ensemble learning combines various machine learning models to improve the final accuracy of the model. A vote is taken from the various classifiers used.

C. *Model Performance*

The machine learning model's performance is evaluated by the following measures i.e. confusion matrix, precision, recall, f1-score, support and accuracy.

The performance of the various models tested is shown below:

Logistic Regression

```
[[1781  314]
 [ 601  365]]
              precision    recall  f1-score   support

           0       0.75      0.85      0.80      2095
           1       0.54      0.38      0.44       966

    accuracy                           0.70      3061
   macro avg       0.64      0.61      0.62      3061
weighted avg       0.68      0.70      0.68      3061

0.701078079059131
```

Figure 2: Performance of Logistic Regression

Random Forest Classifier

```
[[1951  144]
 [ 660  306]]
              precision    recall  f1-score   support

           0       0.75      0.93      0.83      2095
           1       0.68      0.32      0.43       966

    accuracy                           0.74      3061
   macro avg       0.71      0.62      0.63      3061
weighted avg       0.73      0.74      0.70      3061

0.7373407383208102
```

Figure 3: Performance of Random Forest Classifier

Stochastic Gradient Descent Classifier

```
[[1714  381]
 [ 579  387]]
              precision    recall  f1-score   support

           0       0.75      0.82      0.78      2095
           1       0.50      0.40      0.45       966

    accuracy                           0.69      3061
   macro avg       0.63      0.61      0.61      3061
weighted avg       0.67      0.69      0.68      3061

0.6863770009800719
```

Figure 4: Performance of SGD Classifier

Naïve Bayes Classifier

```
[[1944  151]
 [ 835  131]]
              precision    recall  f1-score   support

           0       0.70      0.93      0.80      2095
           1       0.46      0.14      0.21       966

    accuracy                           0.68      3061
   macro avg       0.58      0.53      0.50      3061
weighted avg       0.63      0.68      0.61      3061

0.6778830447566155
```

Figure 5: Performance of Naïve Bayes classifier

Decision Tree

```
[[1516  579]
 [ 528  438]]
              precision    recall  f1-score   support

           0       0.74      0.72      0.73      2095
           1       0.43      0.45      0.44       966

    accuracy                           0.64      3061
   macro avg       0.59      0.59      0.59      3061
weighted avg       0.64      0.64      0.64      3061

0.6383534792551454
```

Figure 6: Performance of Decision Tree Classifier

AdaBoost Classifier

```
[[1850  245]
 [ 639  327]]
              precision    recall  f1-score   support

           0       0.74      0.88      0.81      2095
           1       0.57      0.34      0.43       966

    accuracy                           0.71      3061
   macro avg       0.66      0.61      0.62      3061
weighted avg       0.69      0.71      0.69      3061

0.7112054884024829
```

Figure 7: Performance of Adaboost Classifier

KNN

```
[[1059 1036]
 [ 332  634]]
              precision    recall  f1-score   support

           0       0.76      0.51      0.61      2095
           1       0.38      0.66      0.48       966

    accuracy                           0.55      3061
   macro avg       0.57      0.58      0.54      3061
weighted avg       0.64      0.55      0.57      3061

0.5530872263966025
```

Figure 8: Performance of KNN Classifier

Voting Classifier 1 (Logistic Regression + Random Forest + Support Vector Machine)

```
[[2016   79]
 [ 720  246]]
              precision    recall  f1-score   support

           0       0.74      0.96      0.83      2095
           1       0.76      0.25      0.38       966

    accuracy                           0.74      3061
   macro avg       0.75      0.61      0.61      3061
weighted avg       0.74      0.74      0.69      3061

0.7389741914407056
```

Figure 9: Performance of Voting Classifier 1

Voting Classifier 2 (Logistic Regression + Random Forest + Support Vector Machine + SGD)

```
[[2024   71]
 [ 725  241]]
              precision    recall  f1-score   support

           0       0.74      0.97      0.84      2095
           1       0.77      0.25      0.38       966

    accuracy                           0.74      3061
   macro avg       0.75      0.61      0.61      3061
weighted avg       0.75      0.74      0.69      3061

0.7399542633126429
```

Figure 10: Performance of Voting Classifier 2

Voting Classifier 3 (Logistic Regression + Random Forest + SGD Classifier)

```
[[1829  266]
 [ 616  350]]
              precision    recall  f1-score   support

           0       0.75      0.87      0.81      2095
           1       0.57      0.36      0.44       966

    accuracy                           0.71      3061
   macro avg       0.66      0.62      0.62      3061
weighted avg       0.69      0.71      0.69      3061

0.711858869650441
```

Figure 11: Performance of Voting Classifier 3

Voting Classifier 4 (Logistic Regression + Decision Tree + Support Vector Machine)

```
[[1951  144]
 [ 705  261]]
              precision    recall  f1-score   support

           0       0.73      0.93      0.82      2095
           1       0.64      0.27      0.38       966

    accuracy                           0.72      3061
   macro avg       0.69      0.60      0.60      3061
weighted avg       0.71      0.72      0.68      3061

0.7226396602417511
```

Figure 12: Performance of Voting Classifier 4

Voting Classifier 5 (Logistic Regression + Random Forest + Decision Tree + SVC)

```
[[2036   59]
 [ 747  219]]
              precision    recall  f1-score   support

           0       0.73      0.97      0.83      2095
           1       0.79      0.23      0.35       966

    accuracy                           0.74      3061
   macro avg       0.76      0.60      0.59      3061
weighted avg       0.75      0.74      0.68      3061

0.736687357072852
```

Figure 13: Performance of Voting Classifier 5

Voting Classifier 6 (Logistic Regression + Random Forest + Decision Tree + SVC + SGD)

```
[[1972  123]
 [ 693  273]]
              precision    recall  f1-score   support

           0       0.74      0.94      0.83      2095
           1       0.69      0.28      0.40       966

    accuracy                           0.73      3061
   macro avg       0.71      0.61      0.61      3061
weighted avg       0.72      0.73      0.69      3061

0.7334204508330611
```

Figure 14: Performance of Voting Classifier 6

Voting Classifier 7(Logistic Regression + Random Forest + Decision Tree + SVC + SGD + AdaBoost Classifier)

```
[[2003   92]
 [ 710  256]]
              precision    recall  f1-score   support

           0       0.74      0.96      0.83      2095
           1       0.74      0.27      0.39       966

    accuracy                           0.74      3061
   macro avg       0.74      0.61      0.61      3061
weighted avg       0.74      0.74      0.69      3061

0.7379941195687684
```

Figure 15: Performance of Voting Classifier 7

Voting Classifier 8 (Logistic Regression + Random Forest + Decision Tree + SVC + SGD + AdaBoost + KNN)

```
[[1916  179]
 [ 665  301]]
              precision    recall  f1-score   support

           0       0.74      0.91      0.82      2095
           1       0.63      0.31      0.42       966

    accuracy                           0.72      3061
   macro avg       0.68      0.61      0.62      3061
weighted avg       0.71      0.72      0.69      3061

0.7242731133616466
```

Figure 16: Performance of Voting Classifier 8

D. Comparison of various models

The following bar graph depicts the accuracy of the various models used. Most of the models give an accuracy between 65% - 75%. The SGD Algorithm gives the least accuracy.

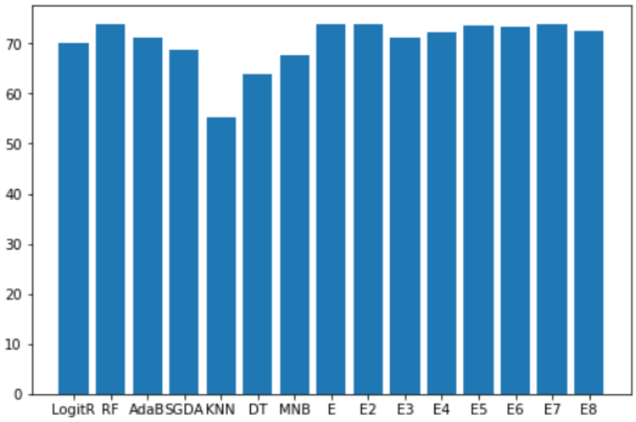


Figure 17: Bar Graph comparing the accuracies of the various classifiers.

Table 2: Comparison of accuracies of various algorithms

S.NO	ALGORITHM	ACCURACY	BULLYING PRECISION
1.	Logistic Regression	70%	54%
2.	Random Forest	73.7%	68%
3.	AdaBoost	71%	57%
4.	SGD Classifier	68.6%	50%
5.	KNN	55.3%	38%
6.	Decision Tree	63.8%	43%
7.	Multinomial Naïve Bayes	67.7%	46%

8.	Logistic Regression + Random Forest + Support Vector Machine Classifier	74%	76%
9.	Logistic Regression + Random Forest Classifier + SVC + SGD Classifier	74%	77%
10.	Logistic Regression + Random Forest + SGD Classifier	71.1%	57%
11.	Logistic Regression + Decision Tree + SVM Classifier	72.2%	64%
12.	Logistic Regression + Random Forest + Decision Tree + SVC Classifier	73.6%	74%
13.	Logistic Regression + Random Forest + Decision Tree + SVC+ SGD Classifier	73.3%	69%
14.	Logistic Regression + Random Forest + Decision Tree + SVC+ SGD + AdaBoost Classifier	73.7%	74%
15.	Logistic Regression + Random Forest + Decision Tree + SVC+ SGD + AdaBoost + KNN Classifier	72.4%	63%

The Random Forest classifier, Ensemble 1 and Ensemble 2 classifiers give the highest accuracy. Even though the accuracy of these three classifiers is similar, we select Ensemble 2 classifier since it gave a higher precision and F1 score.

VII. Results

A. Predicting class labels for input text

Since the voting classifier (ensemble classifier) 2 gives better results, it is deployed onto the webpage using Flask. This machine learning model is used to predict the label of any input text entered by the user. It gives an accuracy of 74%.

The user provides the input text in the textbox area. This new unlabelled data is then sent to the trained machine learning model. The model classifies the given text as 'Bullying' or 'Non-Bullying' and returns the result to the user.

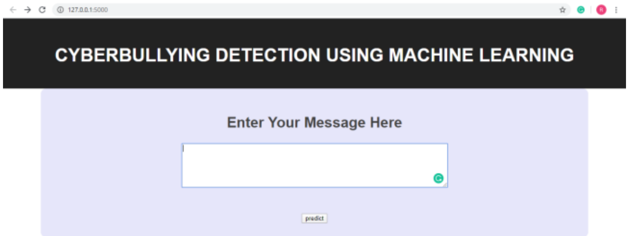


Figure 18: Front-end interface of the web app

In the below figure, the user enters the text "Hi! How are you?? My name is Rashi". To obtain the results, the user must click on the predict button.

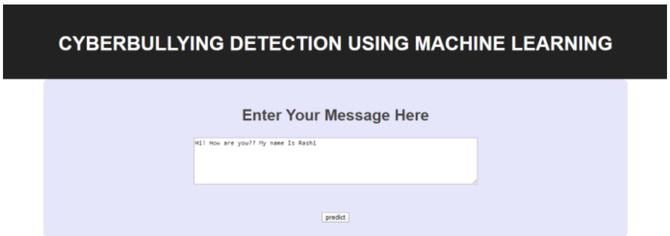


Figure 19: User entering a new text as input in the web app

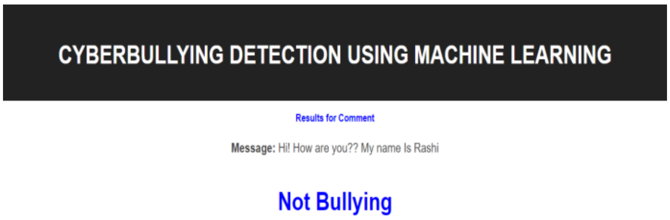


Figure 20: Result interface for the text entered by the user

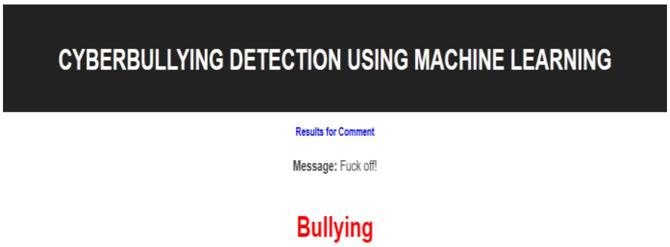


Figure 21: Result shown for the input text "Fuck off!!"

The trained machine learning model doesn't differentiate between texts having different capitalization. This is shown in the result below.

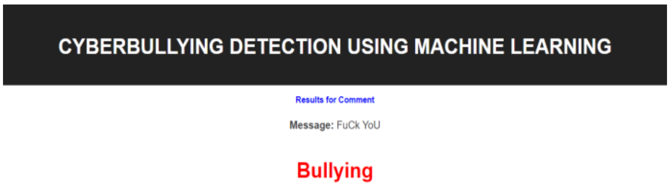


Figure 22: No change in result for various captilization of the same text

The machine learning model can detect 'Bullying' in cases such as 'Racism' and 'Political Hate Speech' as shown below.

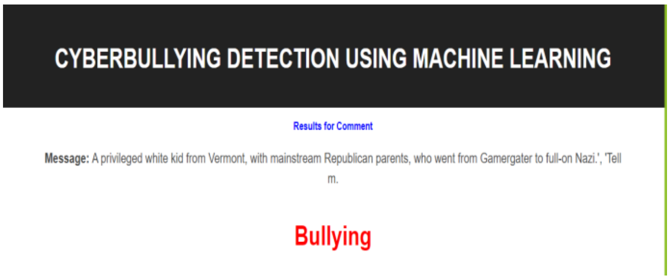


Figure 23: Bullying detected on a tweet containing racist remarks

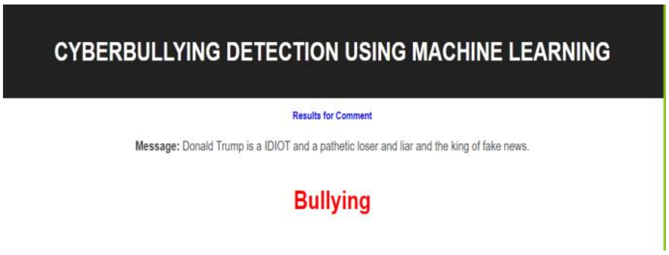


Figure 24: Bullying detected in a tweet containing political hate speech.

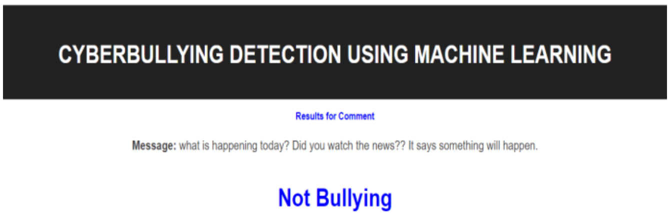
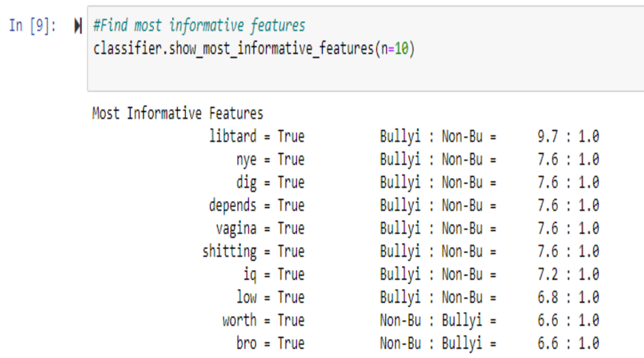


Figure 25: Example of 'Not Bullying' detected

B. Obtaining the most informative features of the data

Using the Bag of Words (BoW) approach and the Naïve Bayes classifier, we can obtain the most informative features in the dataset. In textual data, the features are the various words used. The most informative features represent the most common combinations of words that are used in a sentence that is labelled as 'Bullying', 'Toxic', or 'Offensive'. These sets of words are given below.

The application of the naïve bayes algorithm on unigrams gives the single words/features that are the most abusive. These obtained features are shown below.

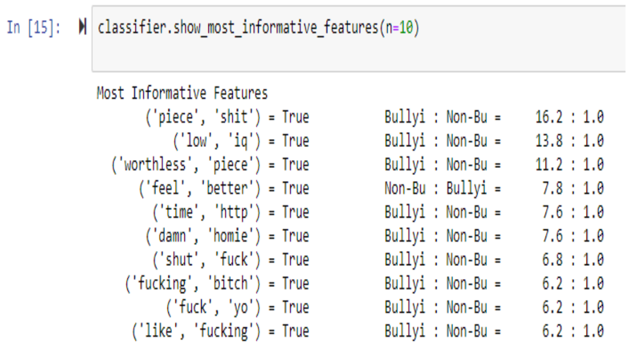


```
In [9]:  #Find most informative features
         classifier.show_most_informative_features(n=10)

Most Informative Features
                 libtard = True           Bullyi : Non-Bu =      9.7 : 1.0
                     nye = True           Bullyi : Non-Bu =      7.6 : 1.0
                     dig = True           Bullyi : Non-Bu =      7.6 : 1.0
                 depends = True           Bullyi : Non-Bu =      7.6 : 1.0
                  vagina = True           Bullyi : Non-Bu =      7.6 : 1.0
                shitting = True           Bullyi : Non-Bu =      7.6 : 1.0
                      iq = True           Bullyi : Non-Bu =      7.2 : 1.0
                     low = True           Bullyi : Non-Bu =      6.8 : 1.0
                   worth = True           Non-Bu : Bullyi =      6.6 : 1.0
                     bro = True           Non-Bu : Bullyi =      6.6 : 1.0
```

Figure 25: The most important features obtained in Unigrams

The application of the naïve bayes algorithm on Bi-grams gives a set of two words/ features that are the most abusive. These obtained features are shown below.

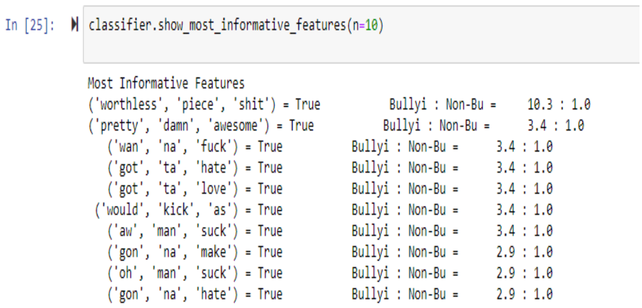


```
In [15]:  classifier.show_most_informative_features(n=10)

Most Informative Features
          ('piece', 'shit') = True           Bullyi : Non-Bu =     16.2 : 1.0
              ('low', 'iq') = True           Bullyi : Non-Bu =     13.8 : 1.0
     ('worthless', 'piece') = True           Bullyi : Non-Bu =     11.2 : 1.0
          ('feel', 'better') = True          Non-Bu : Bullyi =      7.8 : 1.0
            ('time', 'http') = True          Bullyi : Non-Bu =      7.6 : 1.0
           ('damn', 'homie') = True          Bullyi : Non-Bu =      7.6 : 1.0
            ('shut', 'fuck') = True          Bullyi : Non-Bu =      6.8 : 1.0
        ('fucking', 'bitch') = True          Bullyi : Non-Bu =      6.2 : 1.0
              ('fuck', 'yo') = True          Bullyi : Non-Bu =      6.2 : 1.0
         ('like', 'fucking') = True          Bullyi : Non-Bu =      6.2 : 1.0
```

Figure 26: The most important features obtained in Bi-grams

The application of the naïve bayes algorithm on n-grams returns all possible combinations of words/ features that are the most abusive. These obtained features are shown below.



```
In [25]:  classifier.show_most_informative_features(n=10)

Most Informative Features
('worthless', 'piece', 'shit') = True          Bullyi : Non-Bu =     10.3 : 1.0
('pretty', 'damn', 'awesome') = True          Bullyi : Non-Bu =      3.4 : 1.0
   ('wan', 'na', 'fuck') = True          Bullyi : Non-Bu =      3.4 : 1.0
   ('got', 'ta', 'hate') = True          Bullyi : Non-Bu =      3.4 : 1.0
   ('got', 'ta', 'love') = True          Bullyi : Non-Bu =      3.4 : 1.0
  ('would', 'kick', 'as') = True          Bullyi : Non-Bu =      3.4 : 1.0
   ('aw', 'man', 'suck') = True          Bullyi : Non-Bu =      3.4 : 1.0
   ('gon', 'na', 'make') = True          Bullyi : Non-Bu =      2.9 : 1.0
   ('oh', 'man', 'suck') = True          Bullyi : Non-Bu =      2.9 : 1.0
   ('gon', 'na', 'hate') = True          Bullyi : Non-Bu =      2.9 : 1.0
```

Figure 27: The most important features obtained in N-grams

VIII. CONCLUSION

Due to the increase in the usage and popularity of social media, new ways of oppression have surfaced. Meaningful engagement has transformed into a detrimental avenue where individuals are often vulnerable targets to online ridiculing. Predictive models detect this cyberbullying in online content are imperative and this research proffered a prototype model for the same.

The proposed system uses the count vectorizer approach along with a voting classifier to detect cyberbullying in textual data. The voting classifier does a decent job by correctly classifying 74% of the texts while giving a precision of 77%.

IX. FUTURE SCOPE

The limitations of the model arise from the characteristics of real-time social data which are inherently "high-dimensional", "imbalanced or skewed", "heterogeneous", and "cross-lingual". The growing use of micro-text (wordplay, creative spellings, slangs) and emblematic markers (punctuations and emoticons) further increase the complexity of real-time cyberbullying detection. In the future, these problems can be resolved. The project can also be extended to detect cyberbullying in other forms of media such as audio, images, videos. The developed model can also be added as an extension in web browsers such as Google Chrome.

REFERENCES

[1] L. Cheng, J. Li, Y. N. Silva, D. L. Hall, and H. Liu, "XBully: Cyberbullying Detection within a Multi-Modal Context.", WSDM 2019, pp. **339-347, 2019.**

[2] R. I. Rafiq, H. Hosseinmardi, R. Han, Q. Lv, and S. Mishra, "Scalable and timely detection of cyberbullying in online social networks", In Proceedings of the 33rd Annual ACM Symposium on Applied Computing, ACM, pp. **1738-1747**, **2018**.

[3] R. Zhao, and K. Mao, "Cyberbullying detection based on semantic-enhanced marginalized denoising auto-encoder.", IEEE Transactions on Affective Computing , Vol.**8**, Issue.**3**, pp. **328-339**, **2016**.

[4] M. A. Al-garadi, K. D. Varathan, and S. D. Ravana, "Cybercrime detection in online communications: The experimental case of cyberbullying detection in the Twitter network.", Computers in Human Behavior, Vol.**63**, pp. **433-443**, **2016**.

[5] A. Mangaonkar, A. Hayrapetian, and R. Raje, "Collaborative detection of cyberbullying behavior in Twitter data.", 2015 IEEE International Conference on Electro/Information Technology (EIT), pp. **611-616, 2015**.

[6] V. Nahar, X. Li, H. L. Zhang, and C. Pang, "Detecting cyberbullying in social networks using multi-agent system." Web Intelligence and Agent Systems: An International Journal, Vol.**12**, Issue.**4**, pp. **375-388, 2014**.

[7] K. Reynolds, A. Kontostathis, and L. Edwards, "Using machine learning to detect cyberbullying.", In 2011 10th International Conference on Machine learning and applications and workshops, IEEE, Vol.**2**, pp. **241-244**, **2011**.

[8] K. Dinakar, R. Reichart, and H. Lieberman, "Modeling the detection of textual cyberbullying.", In fifth international AAAI conference on weblogs and social media, **2011**.

Authors Profile

Ms. Rashi Kaur is pursuing her Bachelor of Technology in the field of Computer Science & Engineering from Mahatma Gandhi Institute of Technology, Hyderabad. She has worked as an intern at Quikr India and ParallelDots Pvt. Ltd. She has developed projects in the fields of Machine Learning, IoT, JavaScript, and Python. The research areas she is interested in include Data Mining, Data Analytics, Machine Learning and IoT.

Mr. M Srikanth Sagar is an Assistant Professor in the Department of Computer Science and Engineering at Mahatma Gandhi Institute of Technology, Hyderabad. He has work experience of over 7.5 years. He has worked with TCS and Cognizant software companies and he is in academics. His areas of interest in research include Data Sciences, Machine learning.

