1. **INTRODUCTION**

Today, youth are actively involving the major count over usage of social media because it has such an influence nowadays in every aspect. Social media became an easy medium to be connected with our friends and families from all over the world. But as the coin has both sides, social media is being misused by people in such a way that, the lives of many people are at stake. Online abusing had been raising, especially the adults in their 20’s finding means of harassing online upon each other, which is impacting the victim mentally raising depression and suicidal ideations. On the other side, the so-called dark web, where the hackers make trials 24/7 times even started targeting the people on social media by harassing and abusing using electronic means.

* 1. **EXISTING SYSTEM:**

As we can see on various social media platforms, there is a separate comment section for every post. Though there are privacy policies to be maintained, it is giving space for bullying people. Even sometimes, people unknowingly use the bully words that might take into the wrong way. Thus, the existing system is lagging in automatic detection of the bullying comments while the user is typing such that it could indicate about it and warn the user to modify or stop such comments.

In a recent study on cyberbullying detection, gender-specific features were used and users are categorized into male and female groups. It is limited only to gender features. In another study, NUM and NORM features were devised by assigning a severity level to the bad words list (nosewaring.com). NUM is a count and NORM is a normalization of the bad words respectively. The dataset consisted of 3,915 posted messages crawled from the Web Site, Formspring. It showed only 58.5% accuracy, which is very less accurate.

In using a bag-of-words approach examined a baseline text mining system and improved by including sentiment and contextual features. Even with those models, a vector machine learner produces a recall level of 61.9%.

* 1. **PROPOSED SYSTEM:**

As we had been seeing how the technologies are being misused by creating these cyber attacks troubling the people; it is even weird to say that the same technology is used as an antidote for solving the issues relatively. Data Science has been creating a great impact in addressing real-world problems. Thus, it’s an efficient application to involve ML as it became a crucial technology to be implemented in terms of cybersecurity.

Generally, most existing systems are focusing on effects after cyberbullying incidents and there is no system for online cyberbullying detection. Intelligence techniques are also not used in cyberbully detection. The proposed system is to detect the cyberbullying activities and classify them as Bullying or not and even specifying the toxicity of comment, which helps to prevent the cyberbullying victims from facing effects of cyberbullying and take necessary actions like blocking, law enforcement, or taking corresponding legal actions accordingly.

The proposed system involves the application of ML models with the support of NLTK (Natural Language Tool Kit) which runs in the backend and classifies the comments and the bullying probability rate. We applied different algorithms over our model like Linear Regression, Decision Tree Classifier, Naïve Bayes, KNN, Random Forest and analyzed the best accuracy of about 94.36% with the Random Forest Algorithm. Thus, we got good accuracy in our case when compared to the existing model. This helps in classifying the comments in real-time with a correct prediction.

The proposed solution can be easily used in initiating the creation of a plugin for the social media platforms that can automatically warn the user based on the toxicity of the comment while he/she is typing.

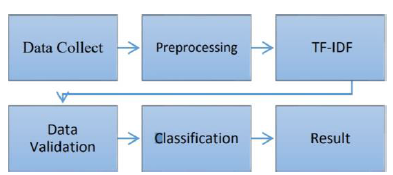
1. **LITERATURE SURVEY**
   1. **RELATED WORK:**

As Cyberbullying became the major availing problem in different social media platforms, there had been researched papers where it has been proposed the trained models over different datasets applying algorithms like Naïve Bayes which is commonly used regarding text-related classifications. One of the research papers depicted the approach of the classifier for the detection of bullying comments based on the Naïve Bayes algorithm. It is not just limited to this particular algorithm because many papers proposed even by fitting the model over logistic regression as it gives the output prediction in either 1 or 0 specifying bullying or not. But when compared it led only up to 75% of score specificity.

Exploring and finding the accuracies with different other algorithms is a must in research where one paper focused on comparison among different types of comments using Random Forest where con 5 attributes were considered on a whole for prediction. The data set was considered as structured data, and the model was able to result in 91.6% accuracy after testing successfully.

* 1. **SYSTEM STUDY:**

The most common approach followed for attaining the solutions for these kinds of problems is portrayed as the below form.



**Fig. 2.2.1: Flow of Approach**

As shown above, the data collected is preprocessed and TF-IDF is done for weighting the frequencies and validating respectively. Then the classifiers are applied for sake of classification. This is the most common machine-applicable approach followed in many research papers regarding text classification into classes.

It has explored 3 various optimization models by applying optimization techniques relatively to find effective ideal values for the variables involved. It has been compared with the SVM, Naïve Bayes, TAN, decision tree over the vivid real-world datasets with binary labels. The results of these numerical experiments were portrayed in the paper. It has been found that though the accuracies obtained were more when compared to the traditional classifier, it was most similar to the structure of NB and other decision classifiers.

As told earlier, we cannot limit over one specific traditional classifier in terms of text, so we will be implementing different classifiers and comparing the scores over the dataset.

1. **DESIGN**
   1. **REQUIREMENT SPECIFICATION:**

**SOFTWARE:**

**Coding Platform used:** Google Colab

**IDE:** PyCharm 2020.3.3

**Python Interpreter:** Python 3.9

**Libraries & versions:**

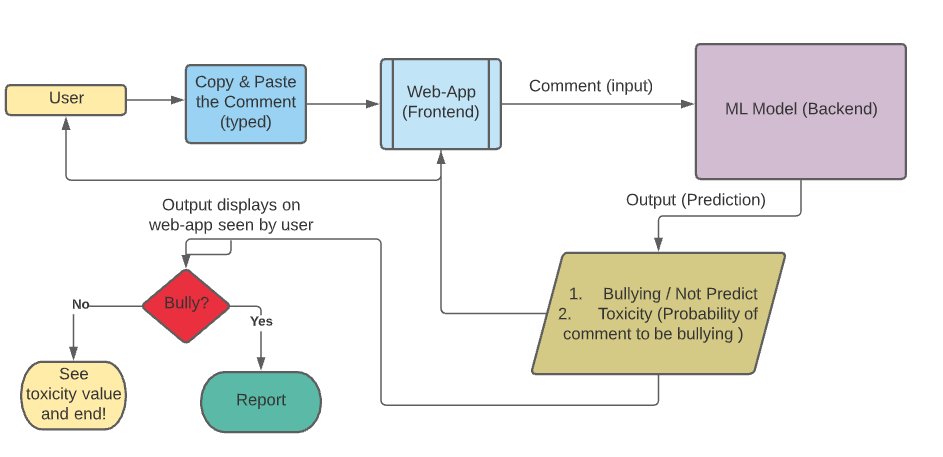
numpy==1.18.5  
pandas==0.25.3  
python-dateutil==2.8.1  
pytz==2020.1  
scikit-learn==0.23.1  
scipy==1.4.1

NLTK Libraries:

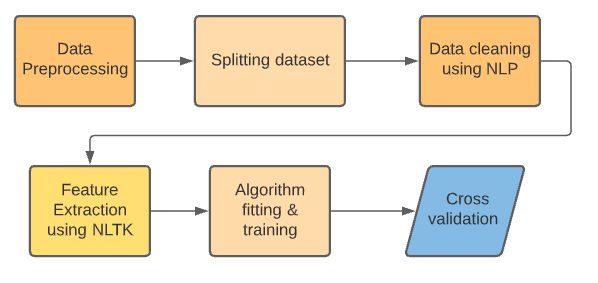
Word Tokenizer, WordNetLemmatizer, CountVectorizer, TfidfTranformer

**To build Web Application:** Flask App ( Flask==1.1.2)

* 1. **DATA FLOW DIAGRAM:**



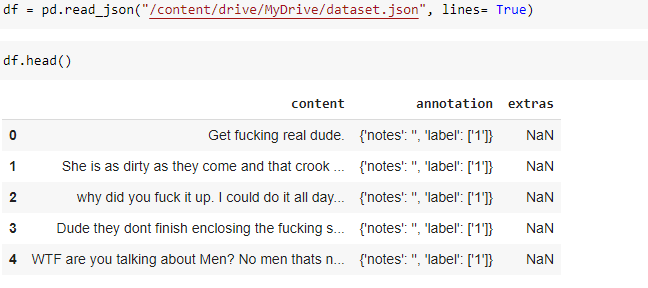
**Fig. 3.2.1: Flow of the proposed solution**



**Fig. 3.2.2: ML Model Backend Flow**

1. **IMPLEMENTATION**
   1. **MODULES:**
2. **Data Set:**

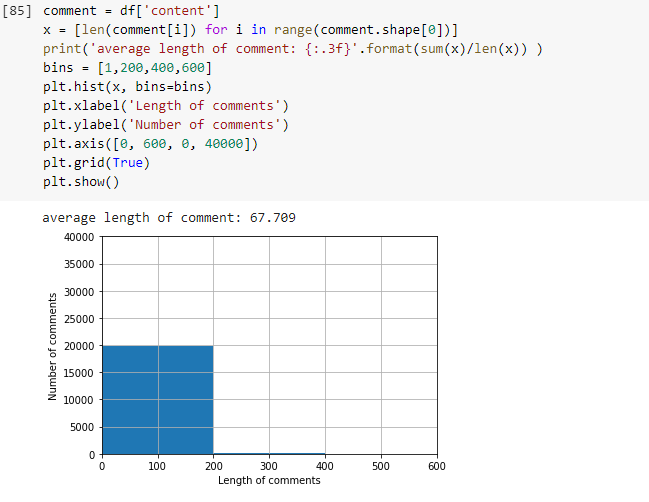
The specific dataset used was the online Wikipedia corpus dataset which has tweets labeled as a bully or not for the prevention of trolls over social media like Twitter. It was hosted on Kaggle in JSON format.The shape of the dataset is (20001,3)

The dataset is read by importing pandas library as shown in the below figure.

**Fig. 4.1.A.i: Dataset used**



**Fig. 4.1.A.ii: Visualization of frequent words in comments (dataset column:content)**



**Fig. 4.1.A.iii: Average length of comments in dataset**

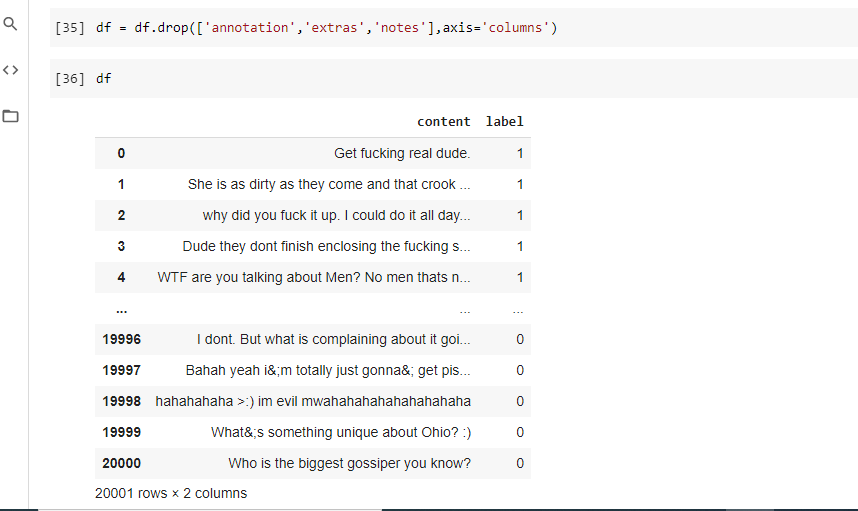
1. **Data Preprocessing:**

It is observed that there is no separate column for the label corresponding to the content. The label is embedded in the annotation column for every content. Thus, it’s better to create a new column and list the labels from annotation respectively.



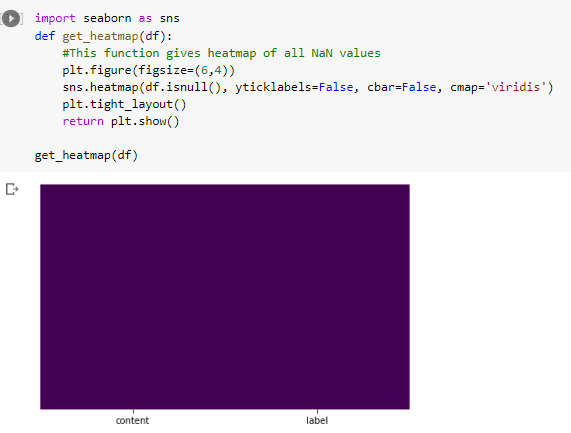
**Fig. 4.1.B.i: New column label is created**

As Data cleaning deals with changing the collected data by removing unwanted ones as per the required use, considering the features necessary for the input and output column, we had dropped all other columns except content (input) and label (output).



**Fig. 4.1.B.ii: Simplified dataset**

Let us now check whether there are any NaN values in any of the fields.



**Fig. 4.1.B.iii: Depicts no NaN values in the dataset (since no heatmap displayed)**

1. **Splitting the data (Training and testing set):**

Generally, we follow the steps of Cross-validation before feeding the dataset directly to the model.

Cross-validation is very important to find out the performance of our model fitted with a specific algorithm.

Thus, according to it, the dataset is split into a Training dataset and Testing dataset with the help of Train-Test Split.

**Train-Test spli**t is a technique that is usually used for cross-validation for evaluating our model via the performance of the ML algorithm applied.

Training Dataset: used to fit the ML model (for training model)

Test Dataset: used to validate the fitted ML model

**Import from sklearn library:**

from sklearn.model\_selection import train\_test\_split



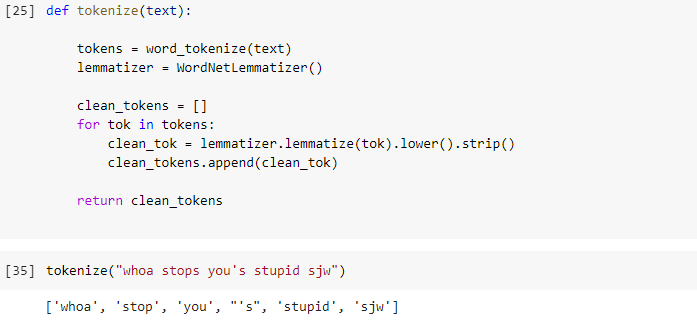
**Fig. 4.1.C.i: Splitting the dataset**

1. **Steps for Data cleaning using NLP:**

This involves:

* Removing all non-alphanumeric characters (irrelevant ones)
* Bringing all the characters to its lowercase
* Performing Tokenization (split the whole text into individual words)
* Perform Lemmatization

Lemmatization is a process of converting a word into its base form considering the context. Thus, the obtained one would be in a meaningful form. This is the reason why we did not perform Stemming here. Because stemming just removes the last few characters of the word and in most cases leads to spelling errors and incorrect meaning of the word.

[**Used - Wordnet Lemmatizer**](https://wordnet.princeton.edu/): It is the freely available database for the English language that establishes structured semantic relations between words. It is one of the most commonly used lemmatizers.

**Fig. 4.1.D.i: Demo of Lemmatize**

1. **Feature Extraction Techniques from the text:**

We know that the machines cannot understand the high-level language (English) that we use. As our approach is computer-oriented, we should first convert them into machine-understandable.

Thus to attain this we use NLTK (Natural Language Tool Kit).

Vectorization is a technique of converting the text into a machine-readable form where all the words are represented in form of vectors.

1. **Word Count with help of CountVectorizer:**

Steps:

* Create an instance of CountVectorizer class
* With the help of instance call fit() to learn vocabulary from one or more comments
* With the help of the same instance, call the transform() on one or more comments to convert it as a vector.

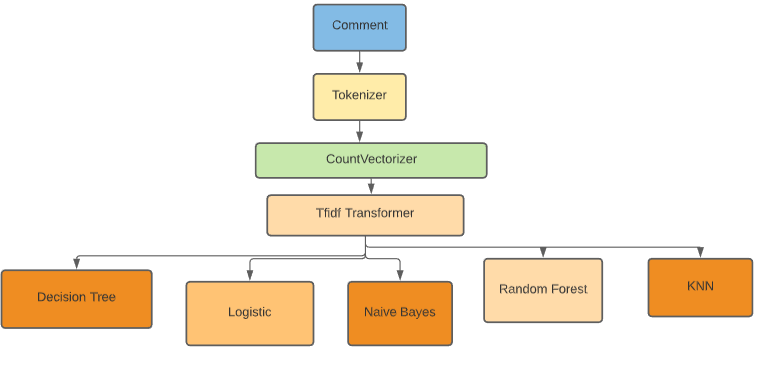
1. **Word frequency with help of TfidfVectorizer:**

**Term Frequency**: retrieves the frequency of the word in the current comment.

**Inverse Document Frequency**: retrieves how rare the word is across all comments.

Here, we want to highlight the words which are frequent in one particular comment but not across all comments. Thus, we use TfidfVectorizer here.

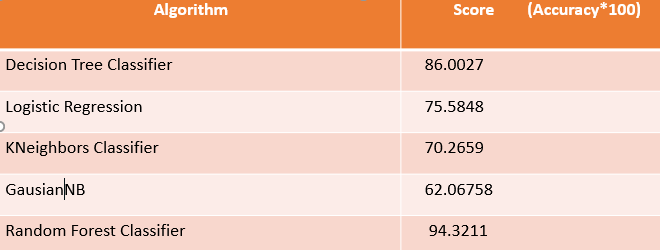
If a word present in all comments, its idf value will be low and vice versa.



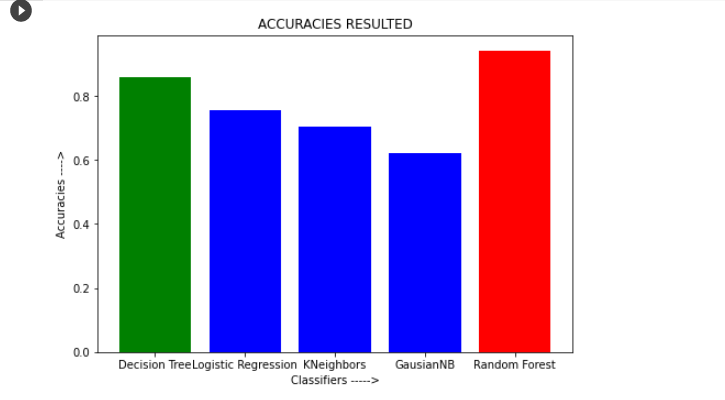
**Fig. 4.1.E.i: Flow Process**

1. **Algorithms:**

The scores depicted in the below figure are obtained when the validation dataset (testing) is given to the model after successful training concerning a specific algorithm.

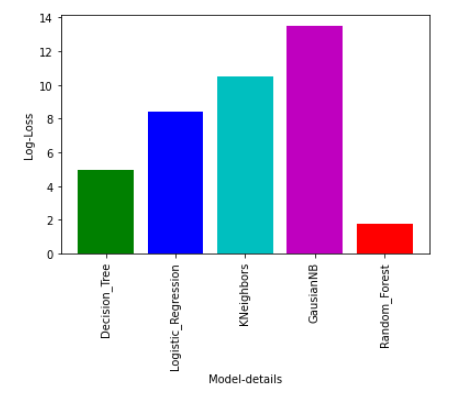
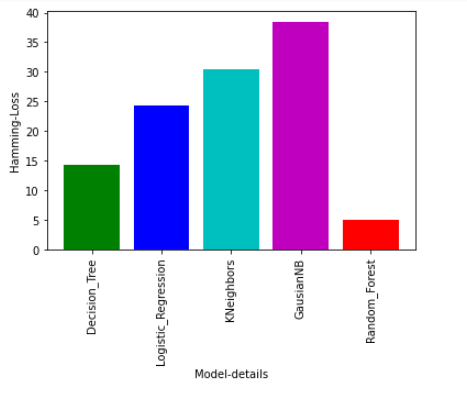


**Fig. 4.1.F.i: Different Algorithms applied and compared**



**Fig. 4.1.F.ii: Different Algorithms applied and compared**

1. **Analysis: (Evaluation Metrics)**

**Free Form Visualisation:** Let us have a plot showing the **hamming-loss** and **log-loss** of different models, which we selected.

**4.1.G.i : Log-Loss Fig. 4.1.G.ii: Hamming-Loss**

Here we got the highest score and least Log-loss,least Hamming-loss when we fitted the model with the Random Forest Algorithm. Out of all the algorithms, we obtained a high rate of accuracy in the case of the Decision Tree classifier and Random Forest Classifier.

When we want our model to be simple to interpret and to overfit, we can use the Decision Tree classifier.

But when we are not concerned about interpretation but only looking for more accurate prediction, we should shift towards a Random forest classifier that builds multiple decision trees and merges them all to get a stable prediction. It can be used for both classification and regression.

Though the Random Forest algorithm reduces variance and does not perform well on a training dataset when compared to the Decision tree classifier, always wins in terms of accuracy when an unexpected test case or validation dataset is given.

In our case, for predicting the real-time bullying comments, we need to consider the accuracy factor. As Random Forest Algorithm performs well in case of unexpected or different comments other than from the training dataset, we can undoubtedly go with it.

Thus for real-time applications like Bullying Comment prediction, we used the **Random Forest Algorithm.**

1. **Web Deployment:**

To test the data over the model and make it usable for the real-world application by which even other users can access the front end of the model and use it without knowing details behind, we deployed using Flask which is generally hosted at localhost.

The front-end is developed using HTML, CSS, Javascript, and then integrated with the flask app such that the data given by the user at the front-end will be further given to the model in the backend, and the prediction is rendered back to the web template respectively.

* 1. **OVERVIEW TECHNOLOGY:**

**ML Technology**:

Pandas

Numpy

Scikit-learn

Natural Language Tool Kit

(CountVectorizer, Tokenizer, TfidfVectorizer)

**Flask App**:

Serialization and Deserialization (model.pkl) file

Flask

**Web App**:

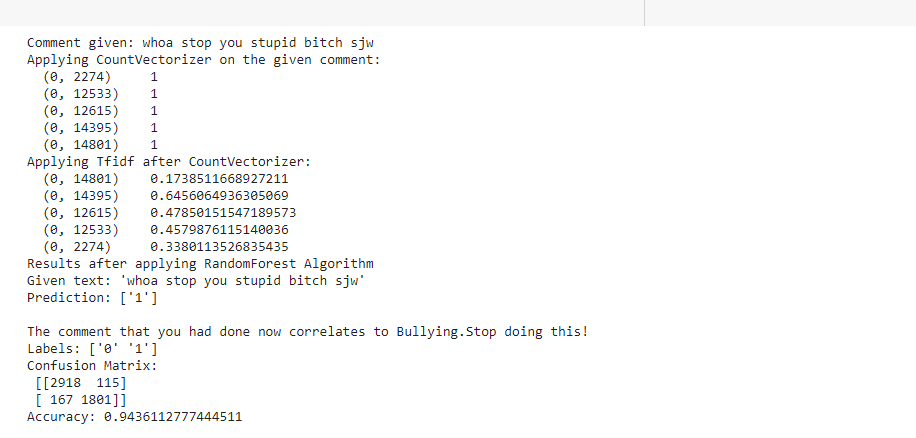
HTML

CSS

JavaScript

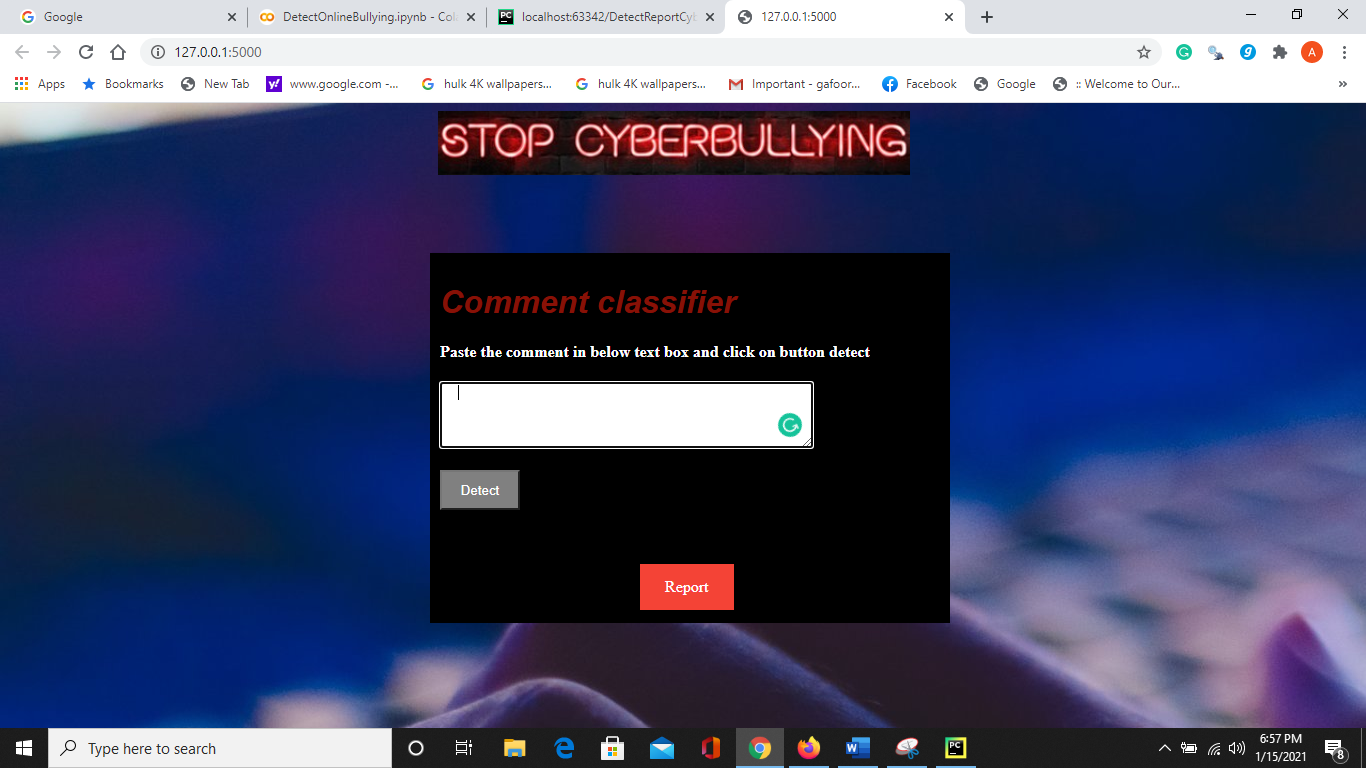
1. **TESTING & RESULTS**
   1. **ALGORITHM RESULT:**

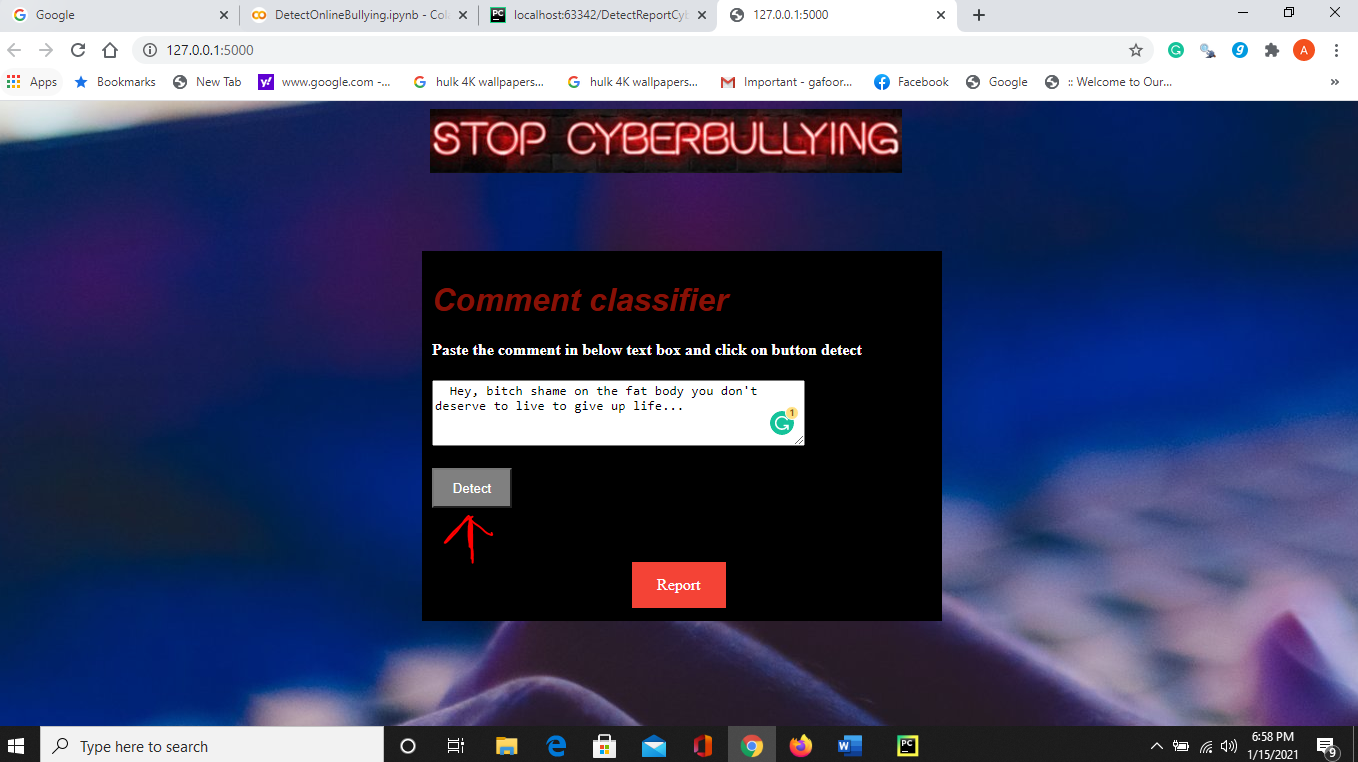
Random Forest Classifier is used in training the model and when validating it we got the test accuracy of about 94.36% which is stable enough for performing prediction over different input (not from the dataset).



**Fig. 5.1.i: Result obtained on testing on comment (not there in the dataset)**

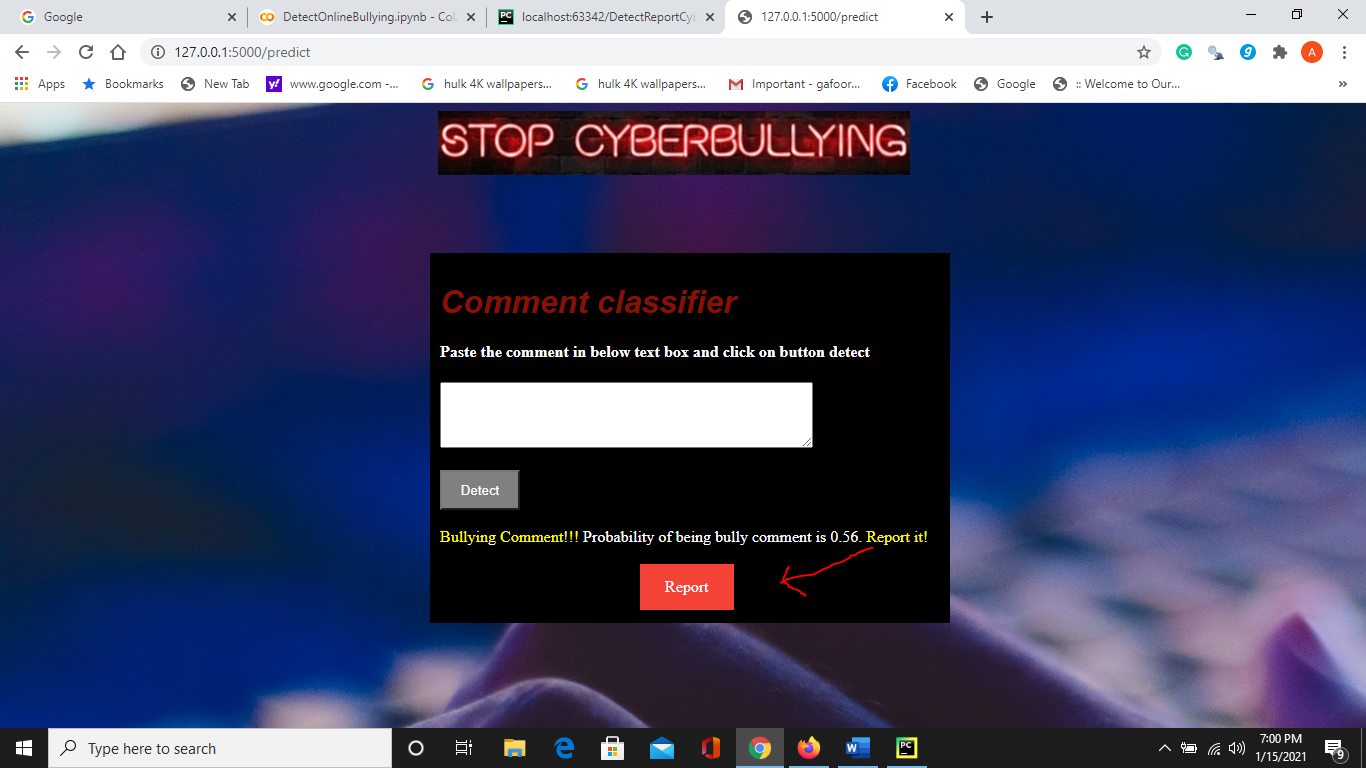
* 1. **WEB-APP RESULTS:**



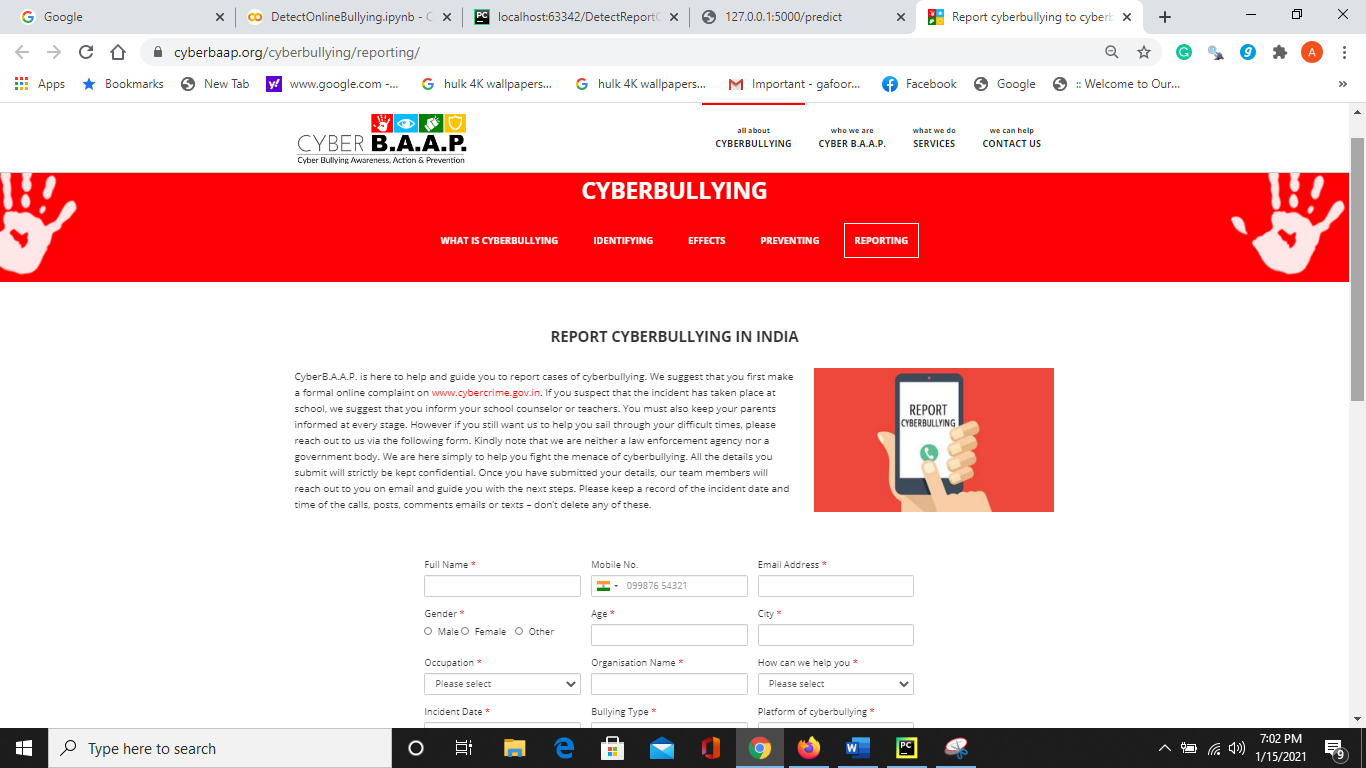
**Fig. 5.2.i: Interface of the Web-App**

**Fig. 5.2.ii: test case 1**

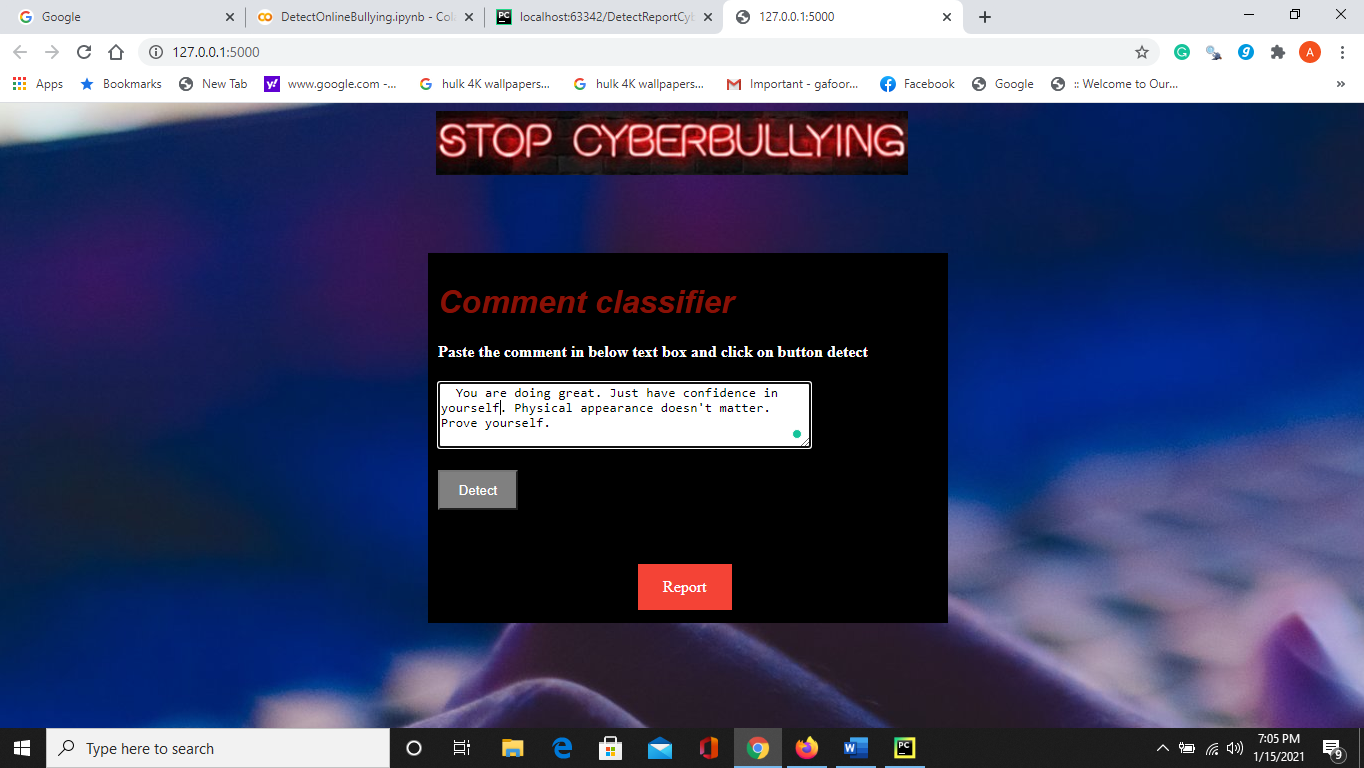
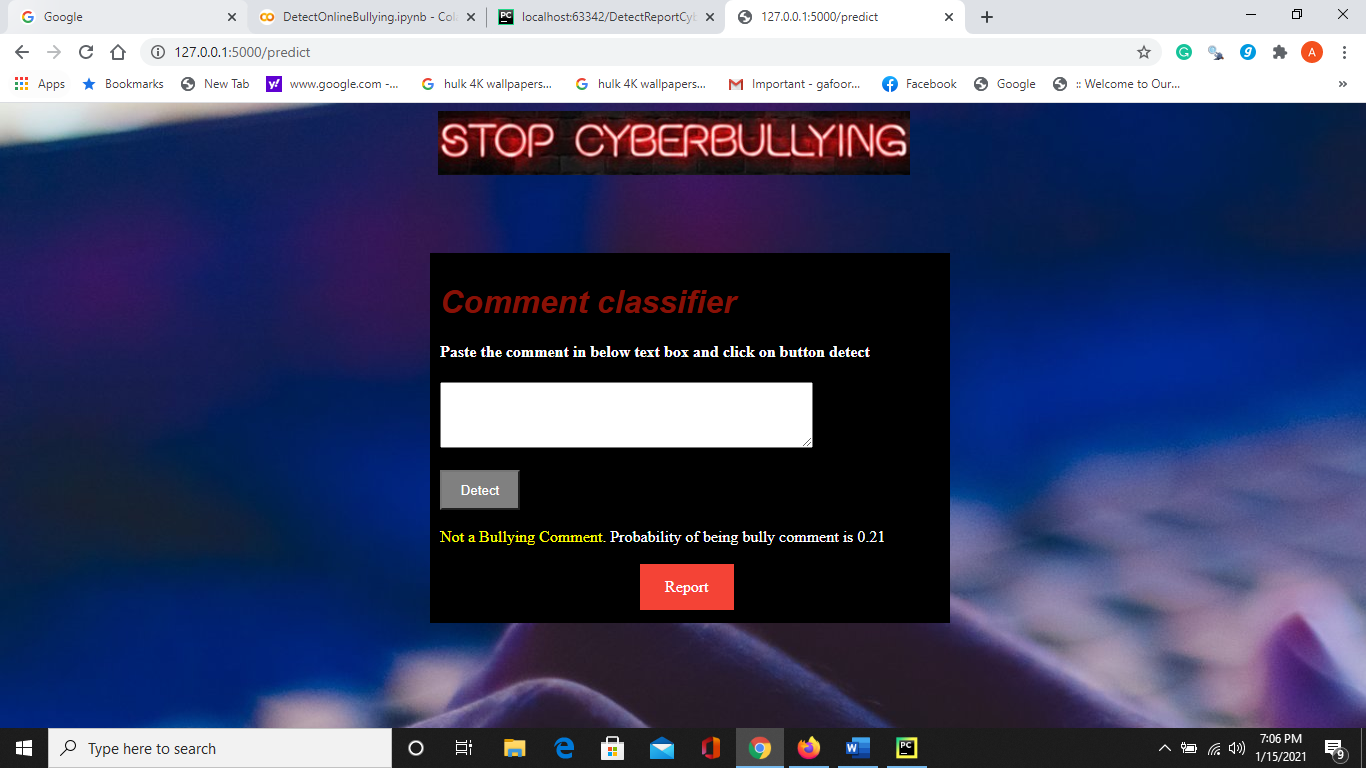
**The user is supposed to paste the comment in the text area & click the Detect button**



**Fig. 5.2.iii: The prediction displayed. Click the report button if bullying comment**



**Fig. 5.2.iv: When clicked on the Report button, it directs to CyberB.A.A.P page to report**

**Fig. 5.2.v: test case 2**

**Fig. 5.2.vi: The prediction displayed as Not a Bullying comment**

1. **CONCLUSION:**

Social media had become a necessary means of communication that connects any person at any point in the world. We cannot imagine our world without relying on it. As the volume of people is getting influenced by it, trolling started raising which is leading to adverse consequences. Though it relates to the glimpses of social life, people started misusing the actual purpose of social media.

It is very important to think before posting a comment over another’s feed because that can affect the person mentally. It is better to have a look over the comments that we post by using the web apps that depict the probability of the comment being abusive.

Even when we come across suspicious bully comments, it is better to paste them into the web app and find the prediction. Relatively report the account along with the proof of prediction from web app to the officials like CyberB.A.A.P such that those accounts would not trouble others. Cybersecurity terms must be followed accordingly. The present proposed model in this paper is trained by the Random Forest Algorithm with attained 94.32% accuracy and run behind the Flask app. The Web-App proposed is user friendly and easy for the user to make use of it and make decisions.

1. **FUTURE SCOPE:**

The further enhancements can be done by creating the extensions or plugin of the Web-App or model to the social media whereby automatic detection of comment as bullying displays as a warning while the person is typing and even resist that person from posting that comment. We can increase the performance of model by exposing it to the huge real time datasets and make it learn from its experiencing even.

Moreover, it is just a beginning; further system may be utilized in various types of cyber crimes prediction with enhancement of cyber bullying and extending its applications in real time.

Thus, at last, we would say that we should be very conscious of social media and report the accounts that try harassing people and beware of trolling as well as have a check on comments that you post. Let’s use the boons of social media without diverting towards the dark side!

**Project GitHub Repository:**

<https://github.com/Areefahnk/Web-App-Detecting-Cyber-Bullying-using-Machine-Learning>

1. **BIBLIOGRAPHY:**
2. <https://en.wikipedia.org/wiki/Cyberbullying>
3. <https://www.journalimcms.org/journal/security-to-prudent-cybercrimes/>
4. <http://www.jcreview.com/fulltext/197-1587745960.pdf>
5. <https://iopscience.iop.org/article/10.1088/1742-6596/1192/1/012029>
6. <https://in.docworkspace.com/d/sIL6Izb48nrv8_wU>
7. <https://iopscience.iop.org/article/10.1088/1742-6596/1192/1/012029>
8. Hariani, Imam Riadi, “Detection of Cyberbullying on Social Media Using Data Mining Techniques”, International Journal of Computer Science and Information Security (IJCSIS) Vol. 15, No. 3, March 2017
9. S. O. N. A. Taheri, "Learning the naive Bayes classifier with optimization models,‖ vol. 23, no. 4, pp. 787 –795, 2013.
10. <https://machinelearningmastery.com/train-test-split-for-evaluating-machine-learning-algorithms/#:~:text=The%20train%2Dtest%20split%20is,dividing%20it%20into%20two%20subsets>.
11. <https://www.datacamp.com/community/tutorials/stemming-lemmatization-python>
12. <https://www.kaggle.com/dataturks/dataset-for-detection-of-cybertrolls>
13. <https://stats.stackexchange.com/questions/285834/difference-between-random-forests-and-decision-tree>
14. <https://opensource.com/article/18/4/flask>
15. <https://irjmets.com/rootaccess/forms/uploads/comment-classification-into-toxic-and-non-toxic-using.pdf>
16. <https://machinelearningmastery.com/prepare-text-data-machine-learning-scikit-learn/>