##### CYBER BULLYING DETECTION

**A PROJECT REPORT**

###### ***Submitted by***

**OISHI BASAK (20BAI10092)**

**SAHIL ARORA (20BAI10264)**

**VATSAL AGARWAL (20BAI10324)**

*in partial fulfillment for the award of the degree*

*of*

##### BACHELOR OF TECHNOLOGY

*in*

# COMPUTER SCIENCE AND ENGINEERING WITH AI & ML

****

**SCHOOL OF COMPUTING SCIENCE AND ENGINEERING**

**VIT BHOPAL UNIVERSITY**

**KOTHRIKALAN, SEHORE**

**MADHYA PRADESH - 466114**

DECEMBER 2021

**VIT BHOPAL UNIVERSITY, KOTHRIKALAN, SEHORE**

**MADHYA PRADESH – 466114**

**BONAFIDE CERTIFICATE**

Certified that this project report titled **“REAL TIME MALWARE AND**

**ADWARE PROTECTION”** is the bonafide work of “**OISHI BASAK (20BAI10092), SAHIL ARORA (20BAI10264), VATSAL AGARWAL (20BAI10384) ”** who carried out the project work under my supervision. Certified further that to the best of my knowledge the work reported at this time does not form part of any other project/research work based on which a degree or award was conferred on an earlier occasion on this or any other candidate.

**PROGRAM CHAIR PROJECT GUIDE**

Dr S Sountharrajan, Associate Professor  **DR. LAKSHMI D (ASSOCIATE PROFESSOR) (SCSE)**

School of Computer Science and Engineering School of Computer Science and Engineering

VIT BHOPAL UNIVERSITY VIT BHOPAL UNIVERSITY

The Project Exhibition I Examination is held on 30ND April 2022.

**ACKNOWLEDGEMENT**

First and foremost I would like to thank the Lord Almighty for His presence and immense blessings throughout the project work.

I wish to express my heartfelt gratitude to Dr. Sountharran S, Program Chair, School of Computer Science and Engineering specialization in Artificial Intelligence and Machine Learning, for much of his valuable support and encouragement in carrying out this work.

I would like to thank my internal guide Dr Lakshmi D , for continually guiding and actively participating in my project, giving valuable suggestions to complete the project work.

I would like to thank all the technical and teaching staff of the School of Computer Science and Engineering , who extended directly or indirectly all support.

Last, but not least, I am deeply indebted to my parents who have been the greatest support while I worked day and night for the project to make it a success.

**LIST OF ABBREVIATIONS**

| **ABBREVIATION** | **WORD** |
| --- | --- |
| ML | Machine Learning |
| AI | Artificial Intelligence |
| SGD | Stochastic Gradient Descent |
| ANN | Artificial Neural Network |
| ROC Curve | Receiver Operating Characteristic Curve |
| DL | Deep Learning |
| SVM | Support Vector Machine |
| CART | Classification And Regression Tree |
| URL | Uniform Resource Locator |
| SVC | Support Vector Classifier |
| API | Application Programming Interface |
| CSS | Cascading Style Sheets |
| B4Soup | Beautiful4Soup |
| HTML | HyperText Markup Language |
| NB | Naive Bayes |

**LIST OF FIGURES AND GRAPHS**

| **FIGURE NO.** | **TITLE** | **PAGE NO.** |
| --- | --- | --- |
| 1. | SYSTEM ARCHITECTURE DIAGRAM |  |
| 2. | SOFTWARE ARCHITECTURAL DIAGRAM |  |
| 3. | CLASSIFICATION MODELS- ACCURACY AND F1 SCORE |  |
| 4. | COMPARISON OF MODELS |  |
| 5. | ROC CURVE |  |
| 6. | ACCURACY AND COST OF MODELS |  |

**ABSTRACT**

Many tweets are posted consistently and the disdain and hostile language happens like never before. It is as a rule extremely critical to recognize unique sorts of oppressive language. By characterizing tweets, we can be aware of people groups' demeanor to specific news, superstars and occasions in twitter very rapidly. For twitter overseers, they can screen and sift through tweets with outrageous language more proficiently founded on arranging. For occasion, a client can report any tweet with oppressive language. These tweets can be characterized first before they go to the directors, which moves along the effectiveness of the screen interaction. For twitter clients, there can be a new capacity to channel these harmful tweets which they probably shouldn't see. A key test for programmed hate speech discovery via virtual entertainment is the partition of disdain discourse from different examples of hostile language. Our task is an exemplary multi-class arrangement information mining issue.

**TABLE OF CONTENTS**

| **CHAPTER NO.** | **TITLE** | **PAGE NO.** |
| --- | --- | --- |
|  | List of Abbreviations  List of Figures and Graphs  List of Tables  Abstract | 4  5  6  7 |
| 1 | **CHAPTER-1:**  **PROJECT DESCRIPTION AND OUTLINE** Introduction 1.2 Motivation for the work  1.3 About Introduction to the project including  techniques  1.5 Problem Statement  1.6 Objective of the work  1.7 Organization of the project  1.8 Summary | 11  11-12  .13-14  .14-15  .15  15-16  17  17-18 |
| 2 | **CHAPTER-2:**  **RELATED WORK INVESTIGATION**  2.1 Introduction  2.2 Core area of the project  2.3 Existing Approaches/Methods  2.3.1 Approaches/Methods -1  2.3.2 Approaches/Methods -2  2.3.3 Approaches/Methods -3  2.4 Pros and cons of the stated Approaches/Methods  2.5 Issues/observations from investigation  2.6 Summary | 18  18-19    19  19  19-20  20-21    21-23  23-24.  24 |
| 3 | **CHAPTER-3:**  **REQUIREMENT ARTIFACTS**  3.1 Introduction  3.2 Hardware and Software requirements  3.3 Specific Project requirements  3.3.1 Data requirement  3.3.2 Functions requirement  3.3.3 Performance and security requirement  3.3.4 Look and Feel Requirements  3.4 Summary | 25  25  25-26  26  27  27-28  28 |
| 4 | **CHAPTER-4:**  **DESIGN METHODOLOGY AND ITS NOVELTY**  4.1 Methodology and goal  4.2 Functional modules design and analysis  4.3 Software Architectural designs  4.4 Subsystem services  4.5 User Interface designs  4.6 Summary | 29  29  29-30  31  31 |
| 5 | **CHAPTER-5:**  **TECHNICAL IMPLEMENTATION & ANALYSIS**  5.1Outline  5.2 Technical coding and code solutions  5.3 Working Layout of Forms  5.4 Prototype submission  5.5 Test and validation  5.6 Performance Analysis(Graphs/Charts)  5.7 Summary | 32  32  32-35    35-37    38  38-41  42-43  43-44 |
| 6 | **CHAPTER-6:**  **PROJECT OUTCOME AND APPLICABILITY**  6.1Outline  6.2 key implementations outlines of the System  6.3 Significant project outcomes  6.4 Project applicability on Real-world applications  6.4 Inference | 45  45  45-46  46  47  47 |
| 7 | **CHAPTER-7:**  **CONCLUSIONS AND RECOMMENDATION**  7.1Outline  7.2 Limitation/Constraints of the System  7.3 Future Enhancements  7.4 Inference | 48  48    48-49  49 |
|  | Appendix A  Appendix B  References |  |

**CHAPTER 1 : PROJECT DESCRIPTION AND OUTLINE**

**1.1 Introduction**

Cyberbullying is bullying with the use of digital technologies. It can take place on social media, messaging platforms, gaming platforms and mobile phones. It is repeated behavior , aimed at scaring, angering or shaming those who are targeted.

Face-to-face bullying and cyberbullying can often happen alongside each other. But cyberbullying leaves a digital footprint – a record that can prove useful and provide evidence to help stop the abuse.

The non-consensual distribution of intimate images involves the sharing of intimate images, often of a former partner, with third parties (either via the Internet or otherwise) without the consent of the person depicted in the image. Often the motivation is to take revenge against their former partner. Its effect is a violation of the former partner's privacy in relation to images, the distribution of which is likely to be embarrassing, humiliating, harassing, or degrading to that person.

Social Media is a stage that permits individuals to post anything like photographs, recordings, reports widely and cooperate with society . Individuals associate with virtual entertainment utilizing their PCs or cell phones. The most famous virtual entertainment incorporates Facebook1, Twitter2, Instagram3, TikTok4and so on. Nowadays, online entertainment is engaged with various areas like schooling , business , and furthermore for the honorable objective .Social media is likewise improving the world's economy through setting out many new position open doors

**1.2 Motivation for the work**

The expression "Cyberbullying" signifies, utilization of Information Innovation to hurt or bother others in a conscious, rehashed, and threatening way. It is not quite the same as customary harassment as it can happen 24 hours of the day and seven days every week . It occurs as mean instant messages, spreading rumours, posting messages, and sharing humiliating pictures and recordings on person to person communication destinations. Once such overly critical messages/pictures/recordings are posted, it is very hard to take these posts off the web-based entertainment destinations. Cyberbullying conduct isn't just unsatisfactory yet in addition can lead to disastrous results. Subsequently, to have a more secure and more helpful social climate, it is important to plan a savvy organization or online watch that will restrict such conduct by observing and separating the indecent, contemptuous, and inappropriate substance from the online entertainment. In contrast to the common methodologies, which are successive in nature, an appropriated worldview is more reasonable for identifying cyberbullying because of the accompanying reasons:

i) as the Twitter information is created in dispersed and nonconcurrent way, it is smarter to identify the cyberbully conduct at various areas in an organization,

ii) a consecutive discovery strategy will experience the ill effects of a solitary point disappointment, and

iii) a conveyed location can lessen the investigation time by taking advantage of the inborn equal nature related with the free age of tweets

**1.3 Introduction to the project including techniques**

Supervised machine learning techniques infer a classification function from labeled training data. Word vectors extracted from tweets play a key role for this experiment. Extraction of word vector from tweet is accomplished

by utilizing a tokenizer. Then, at that point, this word vector is utilized to connect with a given yield a worth or a name. It is normal that an ideal situation will permit a calculation to decide the class names accurately for each test dataset.

Fining tune the choice of this word vector is conceivable that then, at that point, shapes a trait set. This tuning is finished by choosing boundaries like the Minimum Term Frequency(MTF) furthermore, the Tokenizer. Least Term Frequency (LTF) permits sifting through words whose recurrence in preparing the dataset is underneath anticipated esteem. Tokenizer can be utilized to distinguish some expressions or grouping of words that generally show tormenting cases - e.g., "Go kick the bucket." The accompanying boundary settings are utilized with each AI calculation during the course of these analyses.

I) WordTokenizer with least recurrence of word 1

ii) WordTokenizer with least recurrence of word 2

iii) Bi-GramTokenizer with least recurrence of word 1

iv) Bi-GramTokenizer with least recurrence of word 2

The rest of this part talks about the presentation (as estimated by the exactness, accuracy and review) of various ML calculations (Naïve Bayes, Support Vector Machine and Strategic Regression) in an independent mode. The consequences of the cooperative location are portrayed in Section VI. In both the cases, the result of the investigations is parallel in nature - i.e., each tweet is characterized either as "tormenting" or "nonbullying". A. Explores different avenues regarding adjusted dataset The main arrangement of tests comprised of the utilization of a decent

informational collection that contained 170 harassing and 170 non-tormenting tweets. A reasonable dataset has arrangement classes uniformly addressed, and there is no slant that will deliver a predisposition for both of the classes. Subsequently, for the decent nature dataset, it is normal that every ML calculation will act in a comparable way. Additionally, expanding the MTF ought to result in the increment of accuracy related with the outcomes as the word that regularly shows up in tormenting tweets get higher loads. Be that as it may, the review ought to decrease as the MTF increments since the words that could have been significant for grouping however show up less often in dataset than the set cutoff of the MTF, are sifted through. These outcomes in lessening bogus up-sides. With bi-gram tokenizer, review values are expected to be better, this is on the grounds that bi-gram can rapidly arrange by distinguishing collocations. In any case, it could diminish accuracy because of the expansion in bogus up-sides as the

the dataset is tiny. The results of analyses, when the decent dataset is

utilized as the preparation set, are portrayed underneath in Figures 2 and 3.

F1 and F2 show recurrence of word 1 and recurrence of word 2 separately. N2 shows Bi-GramTokenizer is being utilized. According to the previously mentioned assumptions, all the ML strategies truly do perform comparably. Additionally, Bi-Gram tokenizer improves review than

word tokenizer in practically all cases. Following is a point by point conversation of different techniques.

I) Naive Bayes: This technique shows over 60% review and accuracy for the reasonable dataset. It moreover shows an improvement in exactness, accuracy, and

review when bi-gram tokenizer is utilized, and word recurrence is set to 2.

ii) Support Vector Machine: This calculation can't get an adequate number of positive cases without bi-gram. In any case, with the bi-gram tokenizer and word

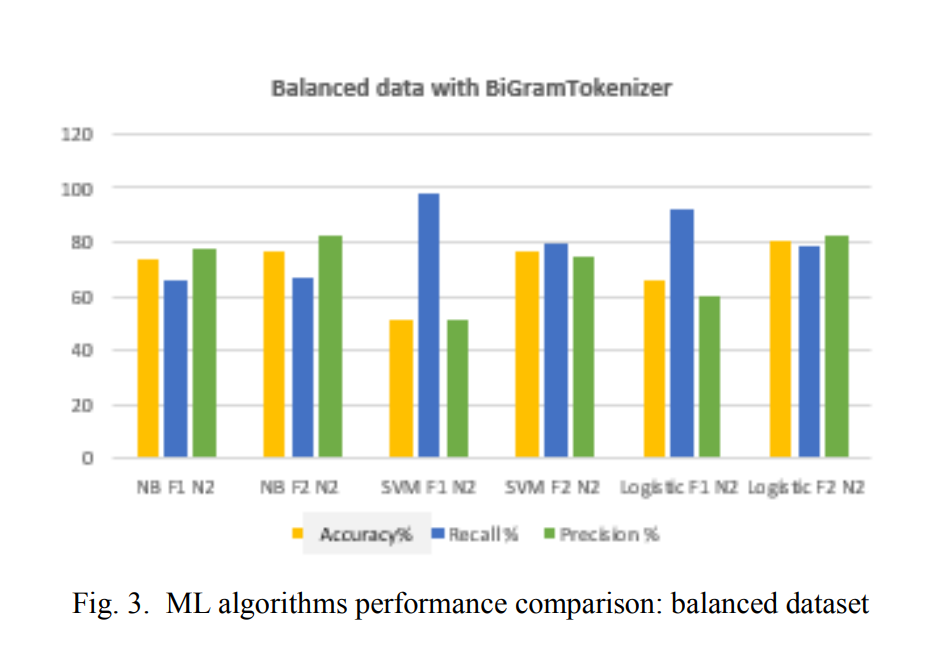
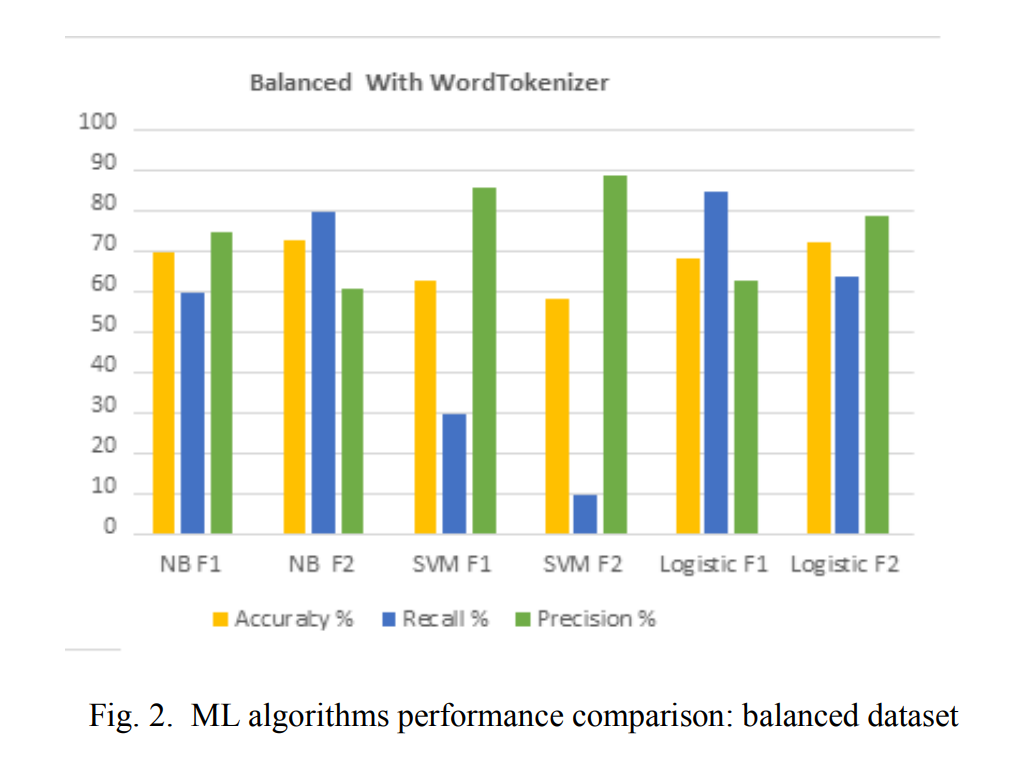
recurrence of 2, the review is worked on by more than 70% that thus expands the grouping abilities.

iii) Logistic Regression: This technique produces rational review than other two strategies, paying little mind to setting utilized for MTF or Tokenizer. The accuracy and review are more noteworthy than 60% in every one of the cases for this dataset.

These outcomes affirm the prevalent view that the discriminative model (Logistic Regression) plays out a minimal better than the generative model (Naive Bayes). The SVM improves review than the Logistic relapse with Bi-gram and MTF 1 yet with lesser accuracy. This is on the grounds that, SVM tracks down the choice

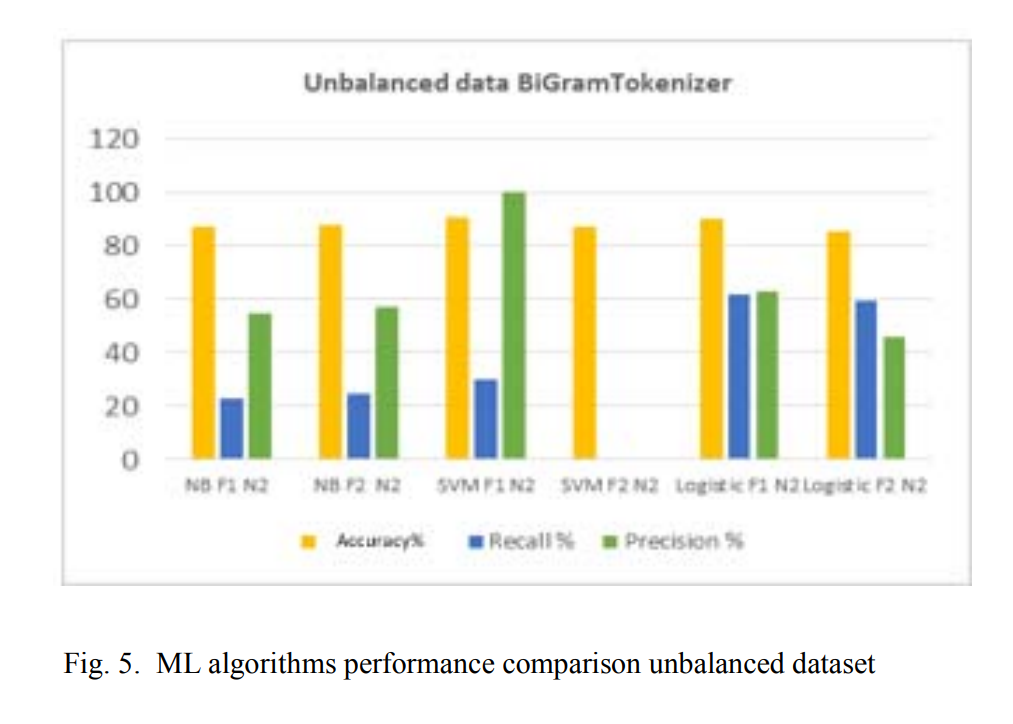
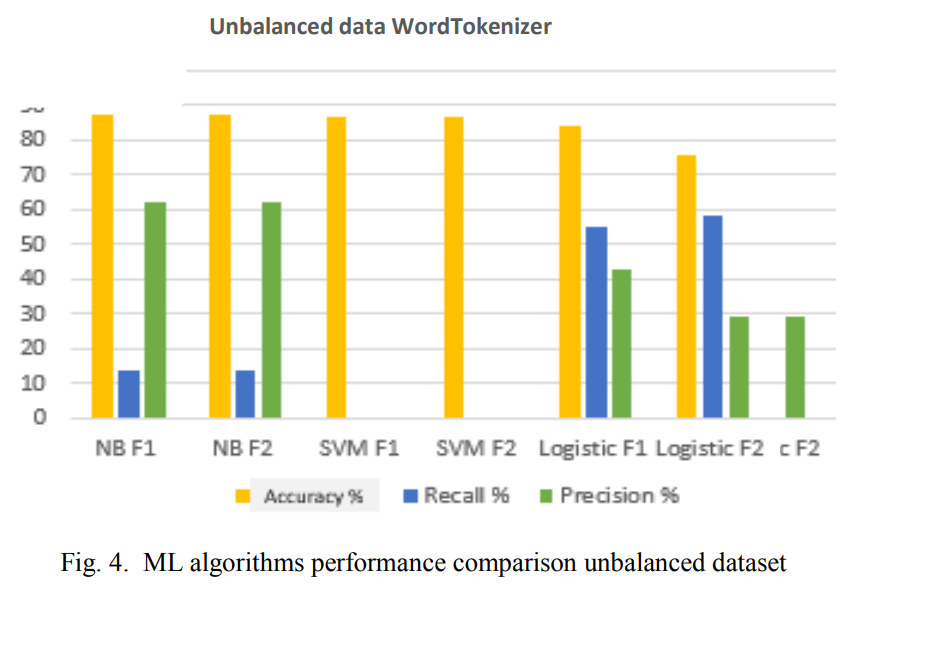
limit by expanding the edge between focuses nearest to the arrangement. In any case, for a given dataset, the limit contrived by SVM is attempting to fit all the outliers of "tormenting" class which diminishes accuracy. Strategic

Relapse attempts to amplify the probability that if a tweet has a place with "tormenting" classification relying upon its promise vector, then, at that point, dissimilar to SVM, it doesn't require adjusting among accuracy and review.

B. Experiments with imbalanced dataset

The second arrangement of trial included the utilization of the uneven information consisting of additional non-tormenting tweets than the tormenting tweets. In the event of uneven datasets, the SVM observes the choice limit that augments the distance between the occurrences of two classes for a given application

which neglects to find this limit and accordingly, performs more regrettable

than the other two ML strategies - a perception that is affirmed by trial results is demonstrated in Figures 4 and 5.Analysis of outcome

i) Naive Bayes: With an unbalanced dataset, the recall values have dropped; however, they are better than the ones obtained via the support vector machine.

ii) Support Vector Machine: It fails when classifying bullying tweets in unbalanced datasets .

iii) Logistics Regression: It captures a significant number of positive cases, and on an average more than 30% of the predictions are correct. It gives more than 50% precision and more than 30% recall for both balanced and unbalanced datasets.

These experiments indicate that the Naive Bayes and the Logistic Regression algorithms are more suited for detecting cyberbullying behavior than the SVM. Naive Bayes being a generative machine learning algorithm learns the distribution of individual classes, while Logistic Regression being a discriminative algorithm learns the decision boundary. In accordance with the popular belief that discriminative algorithms work better than the generative ones , for this application, Logistic Regression seems to be working better.

**1.4 Problem Statement**

The increased utilization of informal communication destinations and the right to speak freely of discourse has given ideal ground to people across all socio economics for cyberbullying and cyber aggression. This leaves intense and recognizable effects on conduct of a casualty, going from aggravation in passionate prosperity and segregation from society to more extreme

also, lethal outcomes.Programmed Cyberbullying recognition has remained

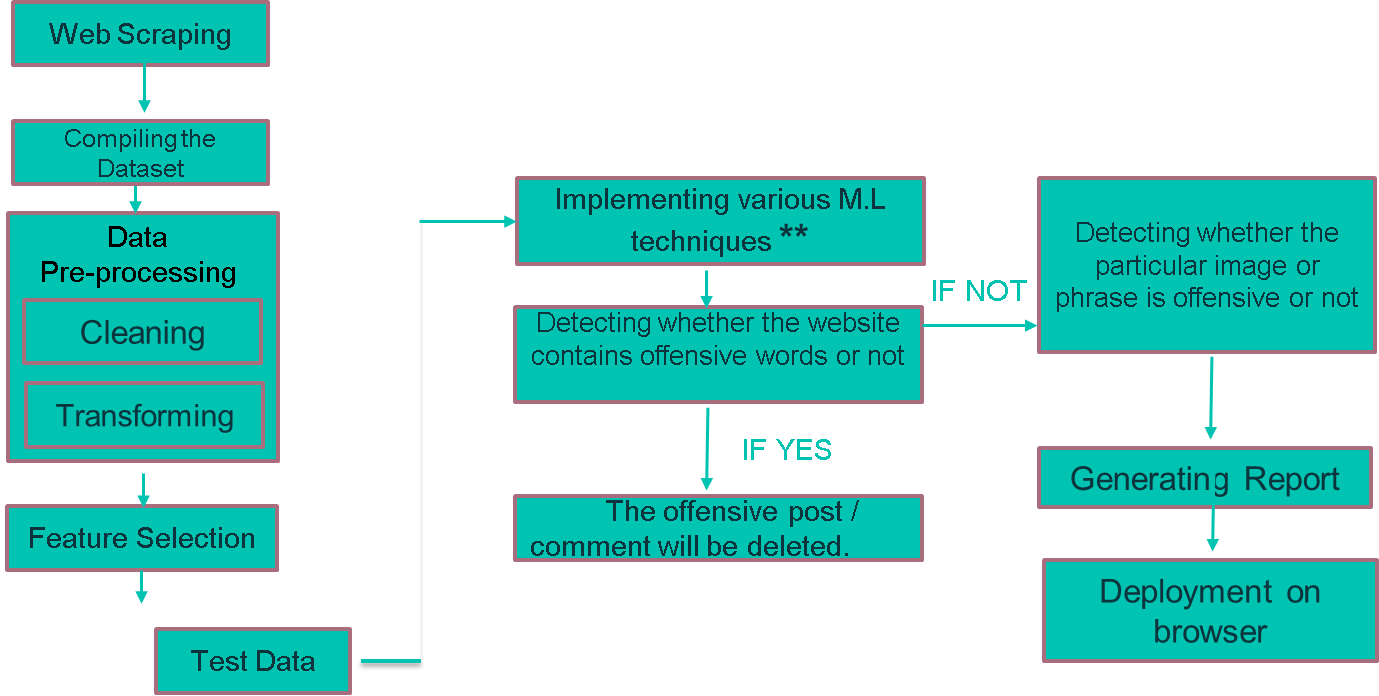
exceptionally testing task since web-based entertainment content is in regular language and is for the most part posted in unstructured free-text structure abandoning the language standards, rules, and principles. Obviously, there exists a significant number of explorations. Be that as it may, a large portion of the recognition conspires and mechanized approaches formed are for asset rich and mature dialects spoken around the world. The results of this review, whenever executed, will help cybercrime focus furthermore, examination offices for checking virtual entertainment contents and in making the internet a secure and more secure spot for all portions of society.

**1.5 Objective of the work**

In the last two decades, a variety of different ML techniques and feature selection algorithms have been widely applied to malware detection, predictions and blocking.

* This model (given that it gets accurate data and prediction after choosing the correct algorithm and proper use of feature engineering) can be used in blocking malicious advertisements and malwares.
* This model will help in reducing the human efforts which will help the user to take proper decisions and steps on time resulting in hassle-free access to websites. It will block ads that interrupt the browsing experience. Blocking annoyances like video ads, pop-ups, flashing banners and more means pages load faster.
* Traditional security products use a virus scanner to detect malicious code, these scanners use signatures created by reverse engineering malware. But with malware that became polymorphic or metamorphic the traditional signature-based detection method used by antivirus is no longer effective against the current issue of malware (Willems, G., Holz, T. & Freiling, F., 2007). In current anti-malware products, there are two main tasks to be carried out from the malware analysis process, which are malware detection and malware classification. The main objective of malware detection is to be able to detect malware in the system. There are two types of analysis for malware detection which are dynamic analysis and static analysis. For effective and efficient detection, the uses of feature extraction are recommended for malware detection (Ahmadi, M. et al., 2016). There are various types of detection methods, the method that we are using will be detecting through hex and assembly files of the malware. Features will be extracted from both hex view and assembly view of malware files. After extracting features to its category, all categories are combined into one feature vector for the classifier to run on them (Ahmadi, M. et al., 2016). For feature selection, separating binary files into blocks to compare the similarities of malware binaries. This will reduce the analysis overhead which causes the process to be faster (Kim, T.G., Kang, B. & Imp, E.G., 2013). To build a learning algorithm, feature that are extracted with the label will be undergo classification with using any classification method for example Random Forest, Neural Network, N-gram, KNN and many others, but Support Vector Machine (VCM) is recommended for the presence of noise in the extracted feature and the label (Stevin, P. & Bystrov, I., 2016). As to generate results, the learning model is to test a dataset with labels to generate a graph which indicates detection rate and false positive rate. To find the best result, repeat the process using many other classification and create a learning model to test on the same dataset. The best result will be the one graph that has the highest detection rate and lowest false positive rates (Linzi, A. et al., 2010).

**1.6 Organization of the project**



**1.7 Summary**

In this Chapter , we have discussed about what is cyberbullying , how it affects others, what was our motivation for our work i.e. what is the real idea behind our project, then we have talked about the different techniques which will be used in our Project, then we have discussed the problem statement, objective and the organization of the project.

In the last two decades, a variety of different ML techniques and feature selection algorithms have been widely applied to malware detection, predictions and blocking them. ​Our model is a clubbed version of malware detection, adware detection from real data taken from the existing active websites. Also, for user’s discretion we are going to deploy it in the form of an extension on Google chrome browser which 95% of people use worldwide.

**CHAPTER-2: RELATED WORK INVESTIGATION**

**2.1 Introduction**

In this segment, cyberbullying discovery approaches zeroing in on internet based informal organization (OSNs) are explored. Dinakar et al partitioned cyberbullying events into different subjects, including race, sexuality, culture, and insight. Therefore, they used a few disputable recordings from YouTube as a utilization case to order the remarks posted on them utilizing four distinct classifiers (Naive Bayes (NB), Rule-based Jrip, Tree-based J48, and SVM). The dataset has around 50,000 remarks and separated into half preparation,

30% approval and 20% testing. Be that as it may, the best precision as gotten by Rule-based Jrip has not surpassed 80%. Hee et al.proposed a method to distinguish fine-grained sorts of cyberbullying, like put-downs and dangers. The creators used cyberbullying content that has phonetic qualities like those found in OSNs; this substance (English and Dutch) was removed from the Ask.fm site. The creators classified the possible subjects of a cyberbullying discussion into three classes: harasser, casualty, and onlooker. The onlooker class was parted into two classifications: the observer who safeguards the person in question, i.e., observer protector, and the spectator who empowers the harasser, i.e., observer colleague. Then, SVMs were utilized to separate the remarks. Be that as it may, in this paper we will zero in on the recognition of cyberbullying on Twitter. Recognition of harassing

words in the tweet contents are seriously difficult.

**2.2 Core area of the project**

The core area of our Project is Machine Learning and Cyber Security. To develop a cyber bullying detector software that implements machine learning to detect offensive words. To validate that offensive words that implements machine learning will be able to achieve a high accuracy rate with low false positive rate.

**2.3 Existing Approaches/Methods**

There exists some existing models and approaches to detect malware and

prevent it. We will discuss some here:

Due to the accretion of social media communication and adverse effects arising from its darker side on users, the field of automatic cyberbullying detection has become an emerging and evolving research trend . Research work presents a cyberbullying detection algorithm for textual data in the English language. It is considered as one of the pioneers and highly cited research. They divided the task into text-classification sub problems related to sensitive topics and collected 4500 textual comments on controversial YouTube videos. This study implemented Naive Bayes, SVM and J48 binary and multiclass classifiers using general and specific feature sets. Study contributed in applied deep learning architectures on Kaggle dataset and conducted experimental analysis to determine the effectiveness and performance of deep learning algorithms LSTM, BiLSTM, RNN and GRU in detecting antisocial behavior. Authors extracted data from four platforms i-e Twitter, YouTube, Wikipedia, and Reddit for developing an online hate classifier in English language using different classification techniques. Research carried out in developed an automated approach to detect toxicity and unethical behavior in online communication using word embeddings and varying neural network layers. They suggested that LSTM layers and mimicked word embedding can uncover such behavior with a good accuracy level.

Few of the studies in recent years have been contributed by researchers on other languages apart from English. Research work is unique and has gathered textual data from Instagram and twitter in Turkish language. They have implemented Naïve Bayes Multinomial, SVM, KNN and decision trees for cyberbullying detection along with Chi-square and information gain (IG) for feature selection. Work accomplished also addresses the problem of cyber aggression in the Turkish language. The work extends comparison of different machine learning algorithms and found optimal results using the Light Gradient Boosting Model. Van Hee, Cynthia, et al. in a proposed cyberbullying detection scheme for Dutch language. This is the first study on Dutch social media. Data was collected from ASKfm where users can ask and answer questions. The research uses default parameter settings for un-optimized linear kernel SVM based on n-grams and keyword systems to identify bullying traces. The F1 score for Dutch language was 61%. Problem of Arabic language cyberbullying detection was addressed and accomplished in. This study used Dataiku DSS and WEKA for ML tasks. The data was scrapped from facebook and twitter. The study concluded that even though the detection approach was not comparable with the other studies in English language, overall Naive Bayes and SVM yield reasonable performance. Research work by Gomez-Adorno, Helena, et al. proposed automatic aggression detection for Spanish tweets. Several types of n-grams and linguistically motivated patterns were used but the best run could only achieve an F1 score of 42.85%. Studies presented in are based on automatic detection of cyberbullying content in German language. Research conducted proposed an approach based on SVM, CNN and ensemble model using unigram, bigrams and character N-grams for categorizing offensive tweets in German language. Research presented in an attempt for the very first time to identify bullying traces in the Indonesian language. Association Rule mining and FP growth text mining were used to identify trends for bullying patterns in Jakarta and Surabaya cities using social media text. This baseline study on Indonesian language was further extended by Nurrahmi, Hani et al. in . Study in made first attempt to develop a corpus of code-mixed data considering Hindi and English language. They proposed a scheme for hate speech detection using N-grams and lexical features. An ensemble approach by combining the predictions of Convolutional Neural Network (CNN) and SVM algorithms were used for identifying such patterns. The weighted F1 score for Hindi language ranged between 0.37 and 0.55 for different experiments . In the year 2019, Association for computational linguistics initiated the project for automatic detection of cyberbullying in Polish language. Research conducted in attempts to uncover cyberbullying patterns in Bengali language implementing passive aggressive, SVM and logistic regression. The optimum accuracy achieved was 78.1%. Recently, work contributed to the first study in Roman Urdu using a lexicon based approach. The dataset was highly skewed, consisting of only 2.2% toxic data. According to , biased sampling and measurement errors are highly prone to classification errors when working on such datasets. Moreover, pattern detection based on predefined bullying and non-bullying lexicons were shortcomings of this study.

For automated detection of complex cyberbullying patterns, studies contributed by different scholars employ supervised, unsupervised, hybrid and deep learning models, vast feature engineering techniques, corpora, and social media platforms. However, the existing literature is mainly oriented towards unstructured data in the English language. Some recent studies and projects have been initiated in other languages as discussed previously. To the best of our knowledge and literature review, no detailed work has been contributed in Roman Urdu to systematically analyze cyberbullying detection phenomenon using advanced preprocessing techniques (involving the usage of Roman Urdu resources) and deep learning approaches under different configurations.

**2.4 Summary**In this Chapter we have discussed the core area of the project, the existing approaches which we are already in use , pros and cons of their approaches and the issues and observations with their used approaches and methods.

**CHAPTER-3:** **REQUIREMENT ARTIFACTS**

**3.1 Introduction**

In this we are going to discuss the Hardware, Software and Data requirements which are used for making our Project i.e. **“CYBER BULLYING DETECTION”.** Anti-malware companies turned to machine learning, an area of computer science that had been used successfully in image recognition, searching and decision-making, to augment their malware detection and classification. Today, machine learning boosts malware detection using various kinds of data on host, network and cloud-based anti-malware components

**3.2 Hardware and Software requirements**

**3.2.1 Hardware Requirements**

Talking about the minimum Hardware Requirements which we will be using for our project are :

For this we will be using a PC with following specifications:

→ It must have at least 16Gb of DDR4 RAM

→ It must have a NVIDIA GTX 1080 (4 GB RAM) Graphic Card

→ It must have a Intel Core i5-9300H or above Processor.

→ Hard disk : 100 GB (minimum) and above

→ We can use any Operating software like Windows, MacOS, or

Linux (Ubuntu).

**3.2.2 Software Requirements**

Talking about the software requirements which we will be using for our project are:

→ Visual Studio Code (Visual Studio Code is a source-code editor )

→ Jupyter Labs (web-based interactive development environment )

→ PyCharm ( integrated development environment used in computer

programming)

→ Languages: Python and Web browsers: Chrome, Firefox

→ Microsoft Excel ( For creating our .csv dataset)

**3.3 Specific Project requirements**

**3.3.1 Data requirement**

→ For Data Requirement and extraction We will be doing Web Scraping with the help of Beautiful Soup. It extracts content and data from a website. Unlike screen scraping, which only copies pixels displayed onscreen, web scraping extracts underlying HTML codes, and with it data stored in a database. The scraper can then replicate the entire website content elsewhere.

→ Data collection is the process of gathering and measuring information on  variables of interest, in an established systematic fashion that enables one to answer stated research questions, test hypotheses, and evaluate outcomes.

→ Data wrangling is the process of cleaning and unifying messy and complex data sets for easy access and analysis. It is also known as data cleaning. It encompasses all the work done on your data prior to the actual Analysis.

→ Data normalization is a process in which data attributes within a data model are organized  to increase the cohesion of entity types. The goal of data normalization is to reduce and even eliminate data redundancy.

**3.3.2 Functions requirement**

i7 family processors were the best particularly for extreme 3D gaming, intensive graphics tasks, multimedia production in standard computer level. Now, this processor becomes a previous generation CPU, for the reason that Intel introduced new and improved processors called 2nd generation Intel® Core™ processors family. But this doesn’t mean that we shouldn’t buy a Core i7 processor. It still does the job it is designed for. The Core i7 processor has several versions both in high end and budget groups. Depending on your work type and budget, you can pick the right one.

**Core i7 General specification**

- All support 64-bit execution

- Integrate 4 Cores (latest Core i7 processor incorporate 6 cores)

- Speed ranges from 2.66GHz to 3.33GHz

- Front Side Bus Speed include 2GHz, 4.8GHz or 6.4GHz

- Support DDR3 main memory

- Support Hyper-threading technology

**3.3.3 Performance and security requirement:**

**Performance And Security of Intel core i7 processor**

The Core i7 processors series targets the gaming industry and for the applications that demand efficient performance and high-end functioning. Generally, Core i7 processor is recommended for:-

- Multitasking, for running multiple programs at the same time

- Multithreading applications

- Intel® hardware-enabled security boosts protection and enables the ecosystem to better defend against evolving and modern cybersecurity threats. Silicon-enabled security technologies help create a trusted foundation, protect workloads, and improve software resilience.

- Creating professional movies and editing graphical tasks

- More than enough for basic tasks such as word processing, internet

browsing and email

* Foundational Security: critical protection to help verify trustworthiness of devices and data.
* Workload and Data Protection: trusted execution for hardware-isolated data protection.
* Software Reliability: platforms that help protect against a range of cybersecurity threats.

**3.4 Summary**

In this Chapter we have discussed the Software and Hardware requirements which will be used in our project , then we have discussed some specific project requirements where we have discussed about the Data Requirements, Function Requirement , and Performance and security requirements. Many kinds of research show that one single malware couldn’t be analyzed in a single tool. Experimental results show that every malware analysis tool has a different metric and way to analyze the malicious code.

**CHAPTER-4: DESIGN METHODOLOGY AND ITS NOVELTY**

**4.1 Methodology and goal**

Cyberbullying Detection implements our coded, machine learning algorithms, in finding a negative comment from the messages it receives by a user. The algorithm first gives the message a value and then based on our pre trained data, it decides if the comment is harsh enough to be transformed or not. If so, the message is run through a series of models in order to change negative components of the sentence into positive components. The transformed sentence is then checked by our initial algorithms. The users communicate through a developed web front face and they are connected to a central server.

Our project provides a feasible solution to detect cyberbullying behavior and its severity in online social networks. We have proposed a cyberbullying detection framework to generate features from Twitter content by leveraging a pointwise mutual information technique. Based on these features, we developed a supervised machine learning solution for cyberbullying detection and multi-class categorization of its severity in Twitter.

**4.2 Functional modules design and analysis**

**1. Web Scraping -**

It extracts content and data from a website. Unlike screen scraping, which only copies pixels displayed onscreen, web scraping extracts underlying HTML codes, and with it, data stored in a database. The scraper can then replicate the entire website content elsewhere.

**2. Data Collection –**

Data collection is the process of gathering and measuring information on variables of interest, in an established systematic fashion that enables one to answer stated research questions, test hypotheses, and evaluate outcomes.

**3. Data Wrangling –**

Data wrangling is the process of cleaning and unifying messy and complex data sets for easy access and analysis. It is also known as data cleaning. It encompasses all the work done on your data prior to the actual Analysis.

**4. Data Normalization -**

Data normalization is a process in which data attributes within a data model are organized to increase the cohesion of entity types. The goal of data normalization is to reduce and even eliminate data redundancy.

**5. Feature Selection -**

Feature Selection is the process where you automatically or manually select those features which contribute most to your prediction variable or output in which you are interested in.

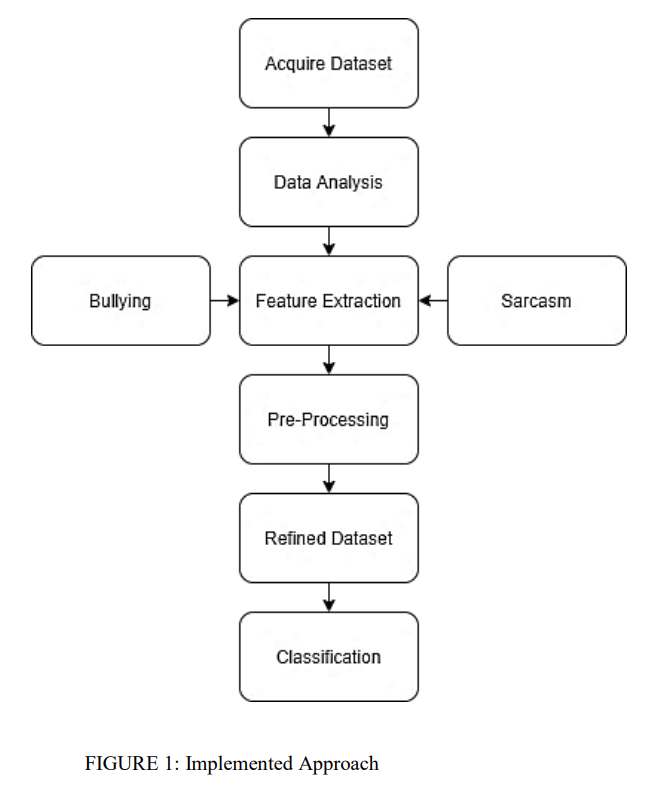
**6. Model Building -**

Model building is the process of developing a probabilistic model that best describes the relationship between the dependent and independent variables.

**7. Deployment-**

The concept of deployment in data science refers to the application of a model for prediction using new data.

**4.3 Software Architectural designs**



**This is our Project Software architecture Design**

So first is web scraping which will extract the data from website then the compilation of the data set is done which we got from web scraping then pre processing of the data which will be needed for building our model , and for feature selection we will be training and testing the model by implementing various Machine Learning Algorithms.

**CHAPTER 5: TECHNICAL IMPLEMENTATION AND ANALYSIS**

**5.1 Outline**

An efficient, robust and scalable malware recognition module is the key component of every cybersecurity product. Malware recognition modules decide if an object is a threat, based on the data they have collected on it. This data may be collected at different phases:

– Pre-execution phase data is anything you can tell about a file without executing it. This may include executable file format descriptions, code descriptions, binary data statistics, text strings and information extracted via code emulation and other similar data.

– Post-execution phase data conveys information about behavior or events caused by process activity in a system.

A machine learning algorithm discovers and formalizes the principles that

underlie the data it sees. With this knowledge, the algorithm can ‘reason’ the properties of previously unseen samples. In malware detection, a previously unseen sample could be a new file. Its hidden property could be malware or benign. A mathematically formalized set of principles underlying data properties is called the model.

**5.2 Technical coding and code solutions**

**5.2.1. Web Scraping**

# dependencies

import pandas as pd

import tweepy

import json

import requests

import os

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

import zipfile

# url for labeled tweet ids

url = 'https://github.com/ZeerakW/hatespeech/archive/master.zip'

# use requests to establish connection

response = requests.get(url)

# create folder 'data'

folder\_name = 'data'

if not os.path.exists(folder\_name):

os.makedirs(folder\_name)

# download zip file

with open(os.path.join(folder\_name, 'master'), mode = 'wb') as file:

file.write(response.content)

# extract zipfile

with zipfile.ZipFile('data/master.zip', 'r') as zipf:

zipf.extractall(os.path.join('data'))

# read in the csv file of labeled tweet ids

labeled\_ids = pd.read\_csv('data/hatespeech-master/NAACL\_SRW\_2016.csv', names = ['id', 'label'])

labeled\_ids.head(2)

# Insert secret tokens and keys from Twitter Developer account

consumer\_key = '###'

consumer\_secret = '###'

access\_token = '###'

access\_secret = '###'

# authenticate as per tweepy docs

auth = tweepy.OAuthHandler(consumer\_key, consumer\_secret)

auth.set\_access\_token(access\_token, access\_secret)

# create api object

api = tweepy.API(auth, wait\_on\_rate\_limit = True, wait\_on\_rate\_limit\_notify = True)

# init counter to keep track of tweets collected and of the failed ids

i = 0

j = 0

failed\_ids = []

# open file to write json objects from api

with open('data/tweets.txt', 'w') as outfile:

for \_ in labeled\_ids.id:

# try-except block since few tweet IDs in the archive may have been deleted

try:

tweet = api.get\_status(\_, tweet\_mode = 'extended')

except:

failed\_ids.append(\_)

j = j+1

print(f'Failed: {\_} | {j} of {len(labeled\_ids.id)}')

continue

# print the number of tweets collected

print(f'Success: {\_} | {i} of {len(labeled\_ids.id)}')

i = i+1

# dump the json object corresponding to the tweet collected from the api

json.dump(tweet.\_json, outfile)

outfile.write('\n')

print(f'Number of Successful Tweets Querried: {i}')

print(f'Number of Failed Queries: {j}')

#load the json data and store it in a list

data = []

with open('data/tweets.txt') as f:

for line in f:

data.append(json.loads(line))

df\_api = pd.DataFrame(data)

#select columns of interest

columns\_of\_interest = ['id', 'full\_text']

df\_api = df\_api[columns\_of\_interest]

# join the dataframes with ID's and tweets

df = labeled\_ids.merge(df\_api, left\_on = 'id', right\_on = 'id', how = 'left');

# drop the id's whose tweets could not be retrieved

df.dropna(how = 'any', inplace = True)

# map labels to binary classes

df['label'] = df.label.map({'none': 'Non-offensive', 'sexism': 'Offensive', 'racism': 'Offensive'});

# save file

df.to\_csv('labeled\_tweets.csv', index = None)

#saving the data to csv file

with open('results.csv', 'w', newline = '', encoding = 'utf-8') as f:

writer = csv.writer(f)

writer.writerow(['Description','Price','Rating','Url'])

writer.writerows(records)

**5.2.2 Initialization of Machine Learning Model**

*The dataset is loaded from the file and is saved in memory.*

import pandas as pd

import matplotlib as plt

import numpy as np

df=pd.read\_csv('MalwareData.csv',sep='|')

**5.2.3 Feature Extraction**

*Since every feature is in numeric form, therefore there is no need for this particular step.*

*Code:*

stopwords=stopwords = nltk.corpus.stopwords.words("english")

other\_exclusions = ["#ff", "ff", "rt"]

stopwords.extend(other\_exclusions)

stemmer = PorterStemmer()

def preprocess(text\_string):

"""

Accepts a text string and replaces:

1) urls with URLHERE

2) lots of whitespace with one instance

3) mentions with MENTIONHERE

This allows us to get standardized counts of urls and mentions

Without caring about specific people mentioned

"""

space\_pattern = '\s+'

giant\_url\_regex = ('http[s]?://(?:[a-zA-Z]|[0-9]|[$-\_@.&+]|'

'[!\*\(\),]|(?:%[0-9a-fA-F][0-9a-fA-F]))+')

mention\_regex = '@[\w\-]+'

parsed\_text = re.sub(space\_pattern, ' ', text\_string)

parsed\_text = re.sub(giant\_url\_regex, '', parsed\_text)

parsed\_text = re.sub(mention\_regex, '', parsed\_text)

return parsed\_text

def tokenize(tweet):

"""Removes punctuation & excess whitespace, sets to lowercase,

and stems tweets. Returns a list of stemmed tokens."""

tweet = " ".join(re.split("[^a-zA-Z]\*", tweet.lower())).strip()

tokens = [stemmer.stem(t) for t in tweet.split()]

return tokens

def basic\_tokenize(tweet):

"""Same as tokenize but without the stemming"""

tweet = " ".join(re.split("[^a-zA-Z.,!?]\*", tweet.lower())).strip()

return tweet.split()

vectorizer = TfidfVectorizer(

tokenizer=tokenize,

preprocessor=preprocess,

ngram\_range=(1, 3),

stop\_words=stopwords,

use\_idf=True,

smooth\_idf=False,

norm=None,

decode\_error='replace',

max\_features=10000,

min\_df=5,

max\_df=0.75

)

**5.2.4 Feature Selection**

df=df.drop(['Name','md5'],axis=1)

5.2.4 Splitting the data

X=df.iloc[:,:-1]

y=df.iloc[:,-1]

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

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2)

**5.2.5 Data Pre-processing**

***Vectorization:***

pos\_vectorizer = TfidfVectorizer(

tokenizer=None,

lowercase=False,

preprocessor=None,

ngram\_range=(1, 3),

stop\_words=None,

use\_idf=False,

smooth\_idf=False,

norm=None,

decode\_error='replace',

max\_features=5000,

min\_df=5,

max\_df=0.75,

)

***Tokenization:***

#Get POS tags for tweets and save as a string

tweet\_tags = []

for t in tweets:

tokens = basic\_tokenize(preprocess(t))

tags = nltk.pos\_tag(tokens)

tag\_list = [x[1] for x in tags]

tag\_str = " ".join(tag\_list)

tweet\_tags.append(tag\_str)

POS Tagging:

pos = pos\_vectorizer.fit\_transform(pd.Series(tweet\_tags)).toarray()

pos\_vocab = {v:i for i, v in enumerate(pos\_vectorizer.get\_feature\_names())}

***Sentiment Analyzing:***

sentiment\_analyzer = VS()

def count\_twitter\_objs(text\_string):

"""

Accepts a text string and replaces:

1) urls with URLHERE

2) lots of whitespace with one instance

3) mentions with MENTIONHERE

4) hashtags with HASHTAGHERE

This allows us to get standardized counts of urls and mentions

Without caring about specific people mentioned.

Returns counts of urls, mentions, and hashtags.

"""

space\_pattern = '\s+'

giant\_url\_regex = ('http[s]?://(?:[a-zA-Z]|[0-9]|[$-\_@.&+]|'

'[!\*\(\),]|(?:%[0-9a-fA-F][0-9a-fA-F]))+')

mention\_regex = '@[\w\-]+'

hashtag\_regex = '#[\w\-]+'

parsed\_text = re.sub(space\_pattern, ' ', text\_string)

parsed\_text = re.sub(giant\_url\_regex, 'URLHERE', parsed\_text)

parsed\_text = re.sub(mention\_regex, 'MENTIONHERE', parsed\_text)

parsed\_text = re.sub(hashtag\_regex, 'HASHTAGHERE', parsed\_text)

return(parsed\_text.count('URLHERE'),parsed\_text.count('MENTIONHERE'),parsed\_text.count('HASHTAGHERE'))

def other\_features(tweet):

"""This function takes a string and returns a list of features.

These include Sentiment scores, Text and Readability scores,

as well as Twitter specific features"""

sentiment = sentiment\_analyzer.polarity\_scores(tweet)

words = preprocess(tweet) #Get text only

syllables = textstat.syllable\_count(words)

num\_chars = sum(len(w) for w in words)

num\_chars\_total = len(tweet)

num\_terms = len(tweet.split())

num\_words = len(words.split())

avg\_syl = round(float((syllables+0.001))/float(num\_words+0.001),4)

num\_unique\_terms = len(set(words.split()))

###Modified FK grade, where avg words per sentence is just num words/1

FKRA = round(float(0.39 \* float(num\_words)/1.0) + float(11.8 \* avg\_syl) - 15.59,1)

##Modified FRE score, where sentence fixed to 1

FRE = round(206.835 - 1.015\*(float(num\_words)/1.0) - (84.6\*float(avg\_syl)),2)

twitter\_objs = count\_twitter\_objs(tweet)

retweet = 0

if "rt" in words:

retweet = 1

features = [FKRA, FRE,syllables, avg\_syl, num\_chars, num\_chars\_total, num\_terms, num\_words,

num\_unique\_terms, sentiment['neg'], sentiment['pos'], sentiment['neu'], sentiment['compound'],

twitter\_objs[2], twitter\_objs[1],

twitter\_objs[0], retweet]

#features = pandas.DataFrame(features)

return features

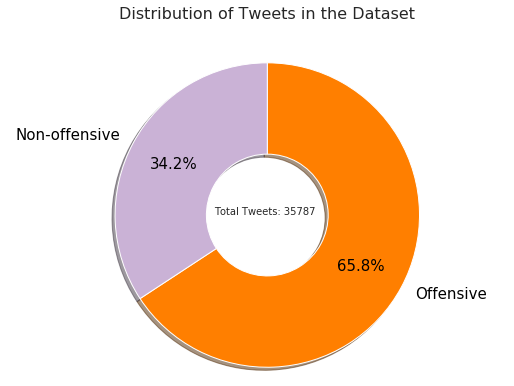
def get\_feature\_array(tweets):

feats=[]

for t in tweets:

feats.append(other\_features(t))

return np.array(feats)

**

***Standard Scalar:***

Standardize features by removing the mean and scaling to unit variance The standard score of a sample x is calculated as: z = (x - u) / s where u is the mean of the training samples or zero if with\_mean=False, and s is the standard deviation of the training samples or one if with\_std=False. Centering and scaling happen independently on each feature by computing the relevant statistics on the samples in the training set. Mean and standard deviation are then stored to be used on later data using transform. Standardization of a dataset is a common requirement for many machine learning estimators: they might behave badly if the individual features do not more or less look like standard normally distributed data.

from sklearn.preprocessing import StandardScaler

SS = StandardScaler()

Train = SS.fit\_transform(Train)

**5.3 Working Layout of Forms**

*Applying Classification Models*

1. **Logistic Regression:**

Logistic regression is a machine learning algorithm for classification. In this algorithm, the probabilities describing the possible outcomes of a single trial are modelled using a logistic function.

Advantages: Logistic regression is designed for this purpose (classification), and is most useful for understanding the influence of several independent variables on a single outcome variable.

Disadvantages: Works only when the predicted variable is binary, assumes all predictors are independent of each other, and assumes data is free of missing values.

from sklearn.linear\_model import LogisticRegression

lr = LogisticRegression()

lr.fit(X\_train,y\_train)

y\_pred=lr.predict(X\_test)

1. **Multinomial Naïve Bayes:**

Naïve Bayes algorithm based on Bayes’ theorem with the assumption of independence between every pair of features. Naïve Bayes classifiers work well in many real-world situations such as document classification and spam filtering.

Advantages: This algorithm requires a small amount of training data to estimate the necessary parameters. Naïve Bayes classifiers are extremely fast compared to more sophisticated methods.

Disadvantages: Naïve Bayes is known to be a bad estimator.

from sklearn.naive\_bayes import GaussianNB

nb = GaussianNB()

nb.fit(X\_train,y\_train)

y\_pred1 = nb.predict(X\_test)

1. **K-Nearest Neighbor:**

Neighbors based classification is a type of lazy learning as it does not attempt to construct a general internal model, but simply stores instances of the training data. Classification is computed from a simple majority vote of the k nearest neighbors of each point.

Advantages: This algorithm is simple to implement, robust to noisy training data, and effective if training data is large.

Disadvantages: Need to determine the value of K and the computation cost is high as it needs to computer the distance of each instance to all the training samples.

from sklearn.neighbors import KNeighborsClassifier

knn = KNeighborsClassifier()

knn.fit(X\_train,y\_train)

y\_pred2=knn.predict(X\_test)

1. **Decision Tree:**

Given a data of attributes together with its classes, a decision tree produces a sequence of rules that can be used to classify the data.

Advantages: Decision Tree is simple to understand and visualize, requires little data preparation, and can handle both numerical and categorical data. Disadvantages: Decision tree can create complex trees that do not generalize well, and decision trees can be unstable because small variations in the data might result in a completely different tree being generated.

from sklearn.tree import DecisionTreeClassifier

tr = DecisionTreeClassifier()

tr.fit(X\_train,y\_train)

y\_pred3=tr.predict(X\_test)

1. **Random Forest:**

Random forest classifier is a meta-estimator that fits a number of decision trees on various sub-samples of datasets and uses average to improve the predictive accuracy of the model and controls over-fitting. The sub-sample size is always the same as the original input sample size but the samples are drawn with replacement.

Advantages: Reduction in over-fitting and random forest classifier is more accurate than decision trees in most cases.

Disadvantages: Slow real time prediction, difficult to implement, and complex algorithm.

from sklearn.ensemble import RandomForestClassifier

rf = RandomForestClassifier()

rf.fit(X\_train,y\_train)

y\_pred4=rf.predict(X\_test)

1. **Support Vector Machine:**

The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future. This best decision boundary is called a hyperplane.

SVM chooses the extreme points/vectors that help in creating the hyperplane. These extreme cases are called as support vectors, and hence algorithm is termed as Support Vector Machine.

Linear SVM is used for linearly separable data, which means if a dataset can be classified into two classes by using a single straight line, then such data is termed as linearly separable data, and classifier is used called as Linear SVM classifier.

param\_grid = {

'C': [0.25, 0.5, 0.75, 1, 1.2] }

clf\_linsvc = LinearSVC()

param\_tuning(clf\_linsvc, param\_grid, training\_data, y\_train, testing\_data, y\_test)

1. **SGD Classifier:**

In Stochastic Gradient Descent, a few samples are selected randomly instead of the whole data set for each iteration. In Gradient Descent, there is a term called “batch” which denotes the total number of samples from a dataset that is used for calculating the gradient for each iteration. In typical Gradient Descent optimization, the batch is taken to be the whole dataset. Although, using the whole dataset is really useful for getting to the minima in a less noisy and less random manner, but the problem arises when our datasets get big.

In SGD, we find out the gradient of the cost function of a single example at each iteration instead of the sum of the gradient of the cost function of all the examples. Since only one sample from the dataset is chosen at random for each iteration, the path taken by the algorithm to reach the minima is usually noisier than the typical Gradient Descent algorithm. But that doesn’t matter all that much because the path taken by the algorithm does not matter, as long as we reach the minima and with a significantly shorter training time.

from sklearn.linear\_model import SGDClassifier

param\_grid = {

'alpha' : [0.095, 0.0002, 0.0003],

'max\_iter' : [2500, 3000, 4000] }

clf\_sgd = SGDClassifier()

param\_tuning(clf\_sgd, param\_grid, training\_data, y\_train, testing\_data, y\_test)

1. **AdaBoostClassifier:**

Ada-boost or Adaptive Boosting is one of ensemble boosting classifier which combines multiple classifiers to increase the accuracy of classifiers. AdaBoost is an iterative ensemble method. AdaBoost classifier builds a strong classifier by combining multiple poorly performing classifiers so that you will get high accuracy strong classifier. The basic concept behind Adaboost is to set the weights of classifiers and training the data sample in each iteration such that it ensures the accurate predictions of unusual observations. Any machine learning algorithm can be used as base classifier if it accepts weights on the training set. Adaboost should meet two conditions:

The classifier should be trained interactively on various weighed training examples.

In each iteration, it tries to provide an excellent fit for these examples by minimizing training error.

How does the AdaBoost algorithm work?

It works in the following steps:

Initially, Adaboost selects a training subset randomly.

It iteratively trains the AdaBoost machine learning model by selecting the training set based on the accurate prediction of the last training.

It assigns the higher weight to wrong classified observations so that in the next iteration these observations will get the high probability for classification. Also, It assigns the weight to the trained classifier in each iteration according to the accuracy of the classifier. The more accurate classifier will get high weight.

This process iterate until the complete training data fits without any error or until reached to the specified maximum number of estimators.

To classify, perform a "vote" across all of the learning algorithms you built.

**5.4 Prototype submission**

from nltk.util import pr

import pandas as pd

import numpy as np

from sklearn.feature\_extraction.text import CountVectorizer

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

from sklearn.tree import DecisionTreeClassifier

data = pd.read\_csv("twitter.csv")

#print(data.head())

data["labels"] = data["class"].map({0: "Hate Speech", 1: "Offensive Language", 2: "No Hate and Offensive"})

#print(data.head())

data = data[["tweet", "labels"]]

#print(data.head())

import re

import nltk

stemmer = nltk.SnowballStemmer("english")

from nltk.corpus import stopwords

import string

stopword=set(stopwords.words('english'))

def clean(text):

text = str(text).lower()

text = re.sub('\[.\*?\]', '', text)

text = re.sub('https?://\S+|www\.\S+', '', text)

text = re.sub('<.\*?>+', '', text)

text = re.sub('[%s]' % re.escape(string.punctuation), '', text)

text = re.sub('\n', '', text)

text = re.sub('\w\*\d\w\*', '', text)

text = [word for word in text.split(' ') if word not in stopword]

text=" ".join(text)

text = [stemmer.stem(word) for word in text.split(' ')]

text=" ".join(text)

return text

data["tweet"] = data["tweet"].apply(clean)

#print(data.head())

x = np.array(data["tweet"])

y = np.array(data["labels"])

cv = CountVectorizer()

X = cv.fit\_transform(x) # Fit the Data

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.33, random\_state=42)

clf = DecisionTreeClassifier()

clf.fit(X\_train,y\_train)

clf.score(X\_test,y\_test)

def hate\_speech\_detection():

import streamlit as st

st.title("Hate Speech Detection")

user = st.text\_area("Enter any Tweet: ")

if len(user) < 1:

st.write(" ")

else:

sample = user

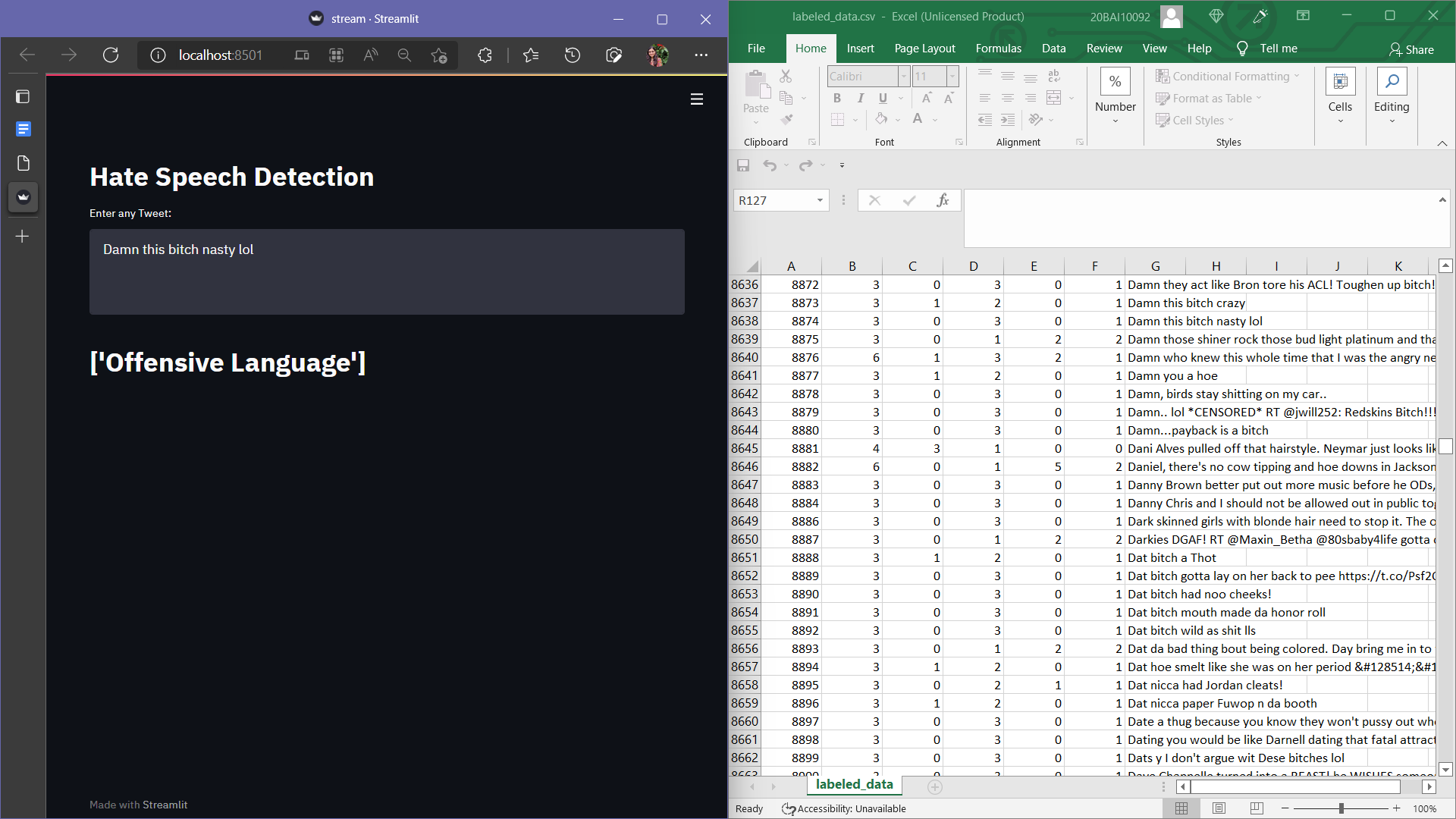
data = cv.transform([sample]).toarray()

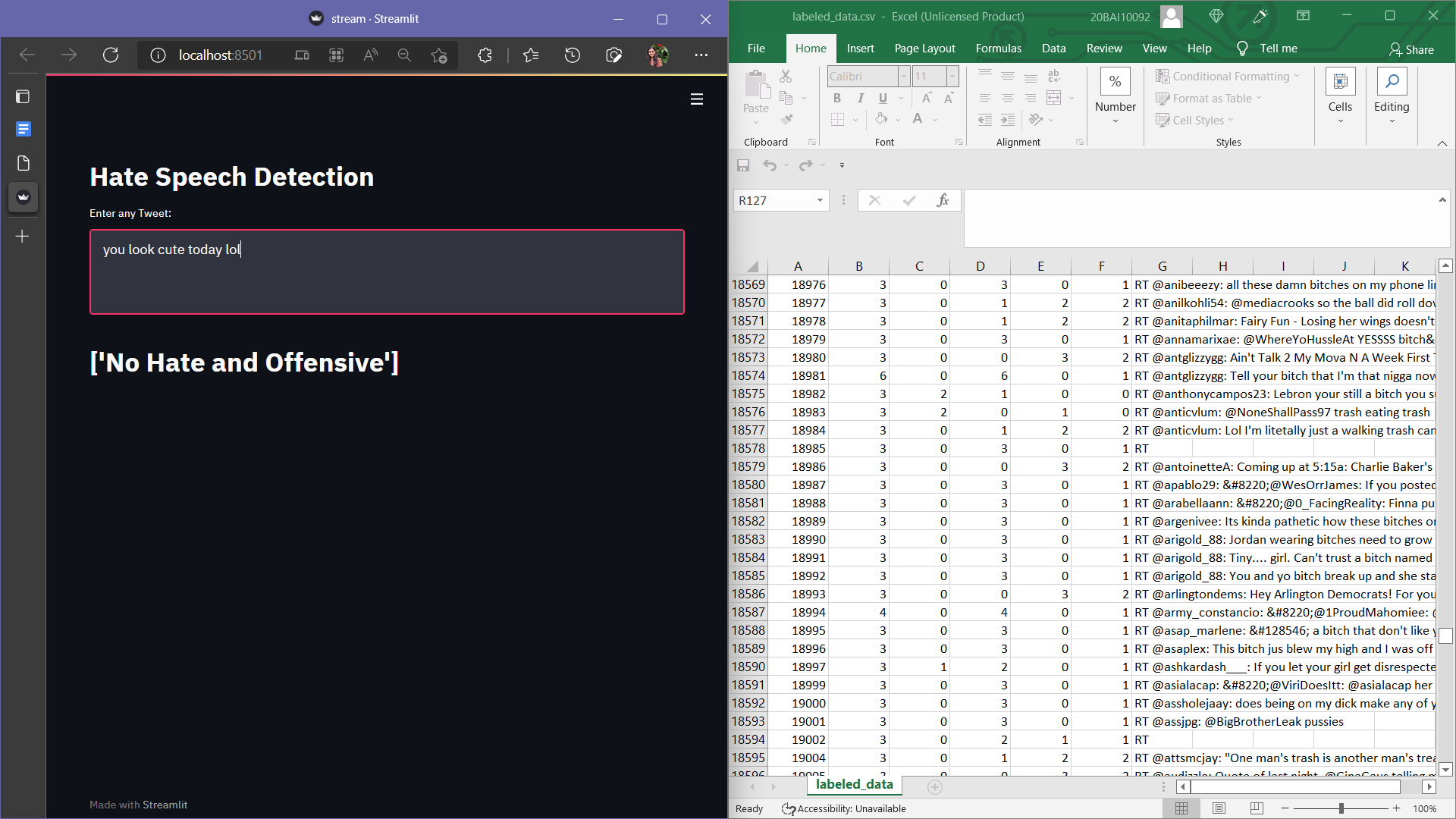
a = clf.predict(data)

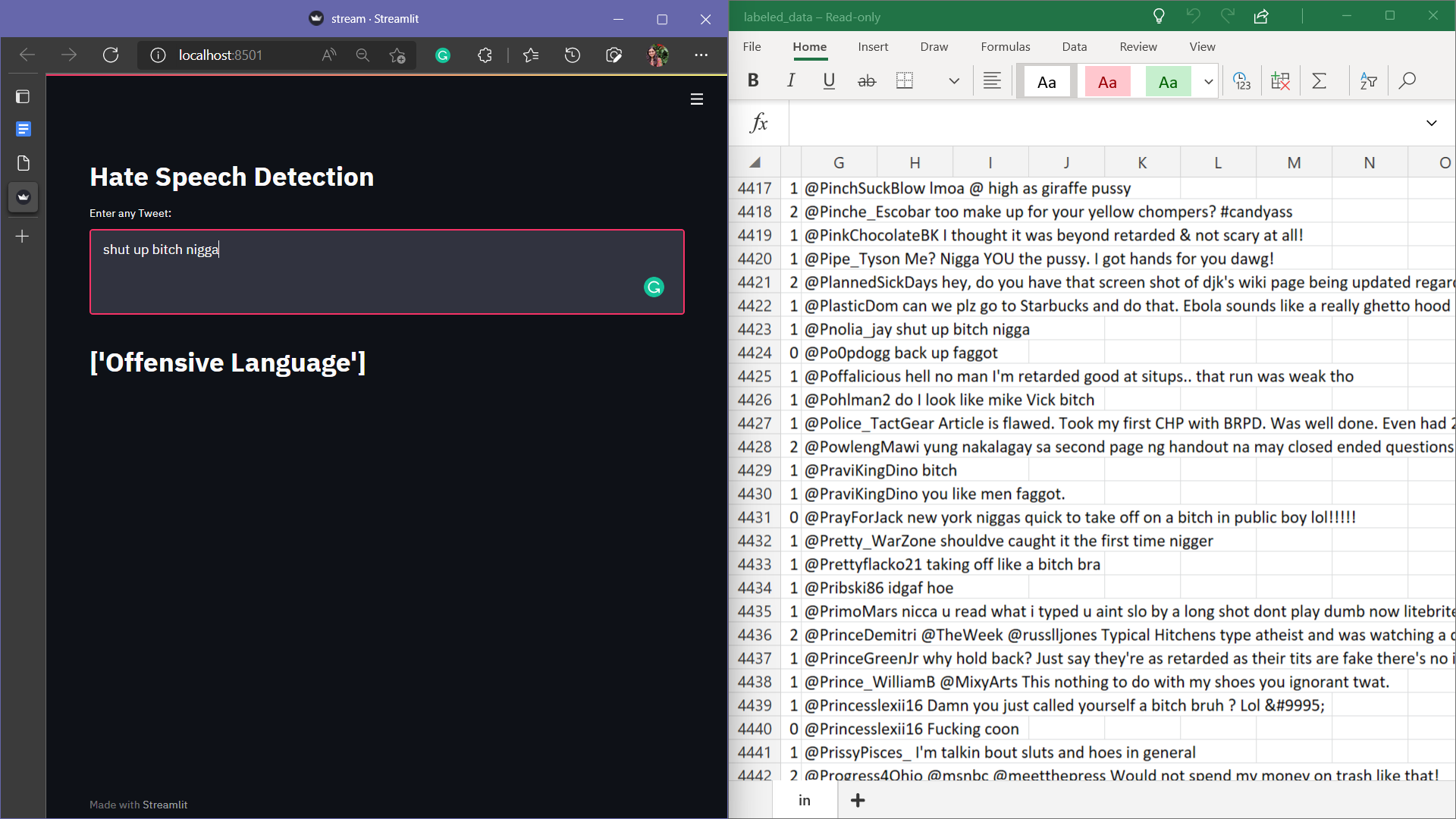
st.title(a)

hate\_speech\_detection()

OUTPUT:

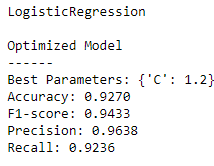




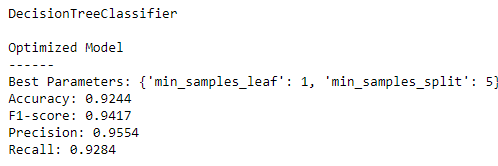


**5.5 Test and validation**

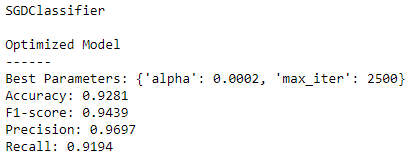
**Logistic Regression:**

****

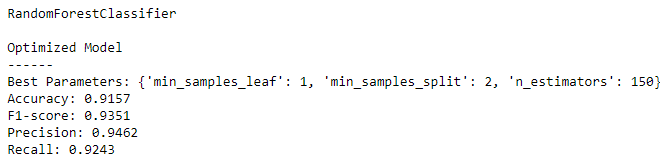
**Decision Tree Classifier:**

****

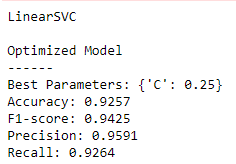
**Stochastic Gradient Descent Classifier:**

****

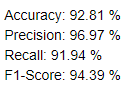
**Random Forest Classifier:**

****

**Linear Support Vector Machine:**

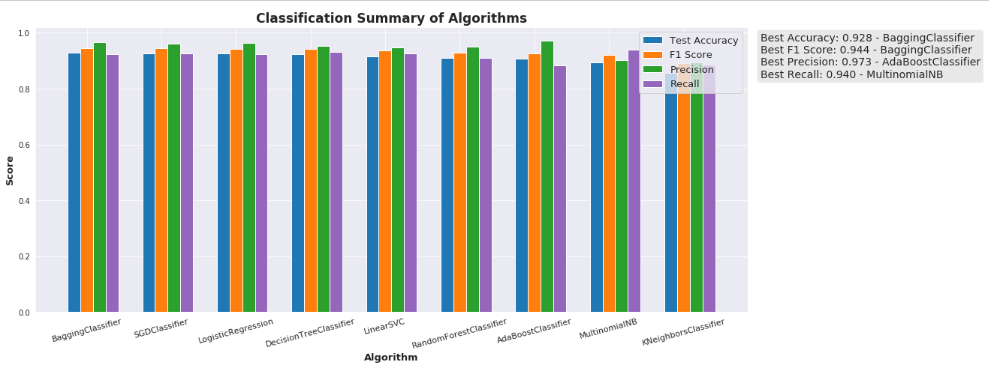
****

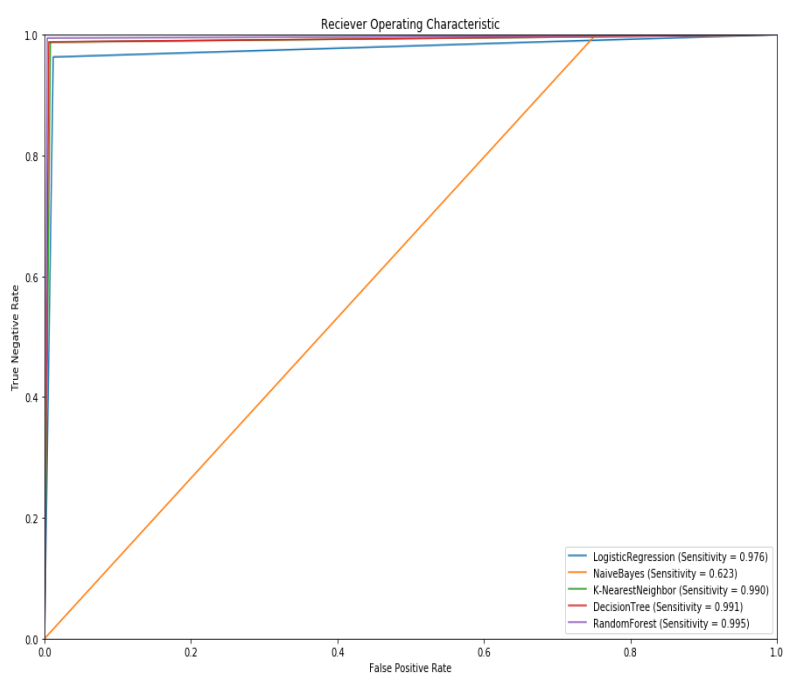
**ADABoost Classifier:**

****

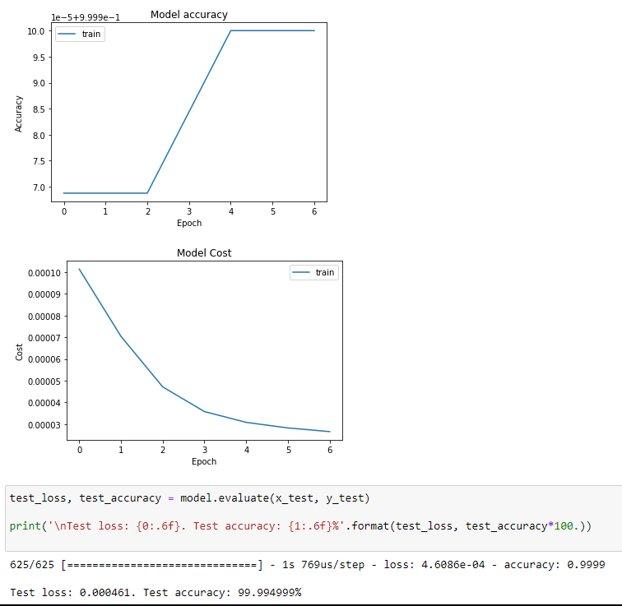
**5.6 Performance Analysis**

*Comparison of Models:*

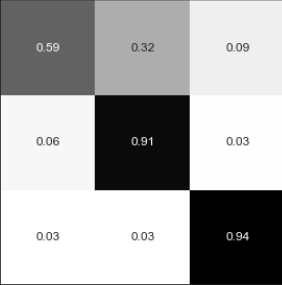
**

*Roc curve:*

*General model cost and accuracy:*



*Confusion Matrix:*



**5.7 Summary**

Our model has shown the different types of tools and ways in which a particular hate speech can be analyzed. Many kinds of research show that one single type of cyberbullying (catcalling, harassing, hate speech, sarcasm, rhetoric, coloquials, etc.) couldn’t be analyzed in a single tool. Experimental results show that every cyberbullying analysis tool has a different metric and way to analyze the offensive words. The possible future work in this domain can be developing an algorithm by which we can use all the software simultaneously and get better results and protect the systems more efficiently. The desired feature extraction and representation methods were selected and the selected machine learning algorithms were applied and evaluated. Collecting a dataset is a tedious task that requires a lot of time and effort. For a more accurate evaluation of the predictors, it is advised to test the models on all the possible types of cyberbullying. In addition to that, it is important to understand that the model will only be able to predict the samples of the families that it has seen earlier.​

From the results above its clear that all models except KNN are showing the best results on the given data. While KNN shows poor result because of its probability function on a huge real-life dataset made by us by web scraping.

**CHAPTER 6 : PROJECT OUTCOME AND APPLICABILITY**

**6.1 OUTLINE :**

Cyberbullying (also known as hate speech, cyberaggression, and toxic speech) is a serious social problem that affects today's Internet users, particularly young people, and can result in serious consequences such as low self-esteem, anxiety, depression, hopelessness, and in some cases, a lack of motivation to live, which can lead to a victim's death. Cyberbullying can take place in a variety of ways. It can take the form of sharing/posting objectionable video content, uploading violent images, or distributing photos without the owner's consent, for example. Text-based cyberbullying, on the other hand, is significantly more common; and we have worked only with the textual harassment words.

**6.2 KEY IMPLEMENTATION OUTLINE OF THE SYSTEM :**

* Our coded, machine learning methods are used by Cyberbullying Detection to discover a negative comment in the messages it gets from a user. The algorithm assigns a value to the message and then determines whether or not the comment is harsh enough to be changed based on our pre-trained data. If it is actually harsh, the algorithm will search across our vast network of users to determine how this user interacts with others on a regular basis, as well as how they interact with the end user on a regular basis.
* The system will determine if the message needs to be changed based on this information. If this is the case, the message is processed through a number of models to transform the sentence's negative components into positive ones.
* Our early algorithms then check the altered text. It is given a value, and if that value yields a positive sentence, the system will deliver the modified positive sentence to the user. Otherwise, the sentence will go through the models once again. Users communicate with one another using a created web front end that is linked to a central server. Clients are the people who use the service. The receiving user will be notified along with the updated message if any messages are modified.
* Our artificial intelligence (AI) system uses machine learning to automatically collect and extract data from the entire user base web page— then trains the model. The engines work across devices (both on the cloud and PCs), they use static and dynamic analysis techniques, and they are deployed in many of the layers of our defense engine.
* The technology will be used by members of the organisations to monitor the activity of the social network community, particularly cyberbullies and victims on Twitter, in order to prevent the cyberbullying incident from worsening. To achieve these goals, a large amount of literature on event detection and cyberbullying detection systems has been examined and studied in order to get insight into the system's development and effectiveness. We gathered cyberbullying-related keywords from research experts and captured targeted tweets as part of our investigation.
* Twitter, for example, is a microblogging service and broadcast medium that has emerged as a disruptive platform for users to broadcast their daily activities, sentiments, and opinions to their friends circle by publishing simple tweets (messages). Cyberbullying is a form of harassment that occurs on social networking sites, allowing cyberbullies to carry out their crimes on susceptible victims, and it is a serious crime in the cyber world that has resulted in death. As a result, the Twitter Cyberbullying Detection System is a solution aimed at effectively discovering cyberbullying-related tweets and we have implemented the same in our model in a likewise manner.

**6.3 SIGNIFICANT PROJECT OUTCOMES:**

The present study took step forward and highlighted limitations in existing cyberbullying detection system. In this study, we provided a systemic framework for identifying cyberbullying severity in Twitter, which is based on previous research from different disciplines. In order to achieve this, we build machine learning multi-classifier for classifying cyberbullying severity into different levels. In order to test the significance of our proposed framework for detecting cyberbullying severity we used publicly available harassment dataset. We developed a framework to create semantic orientation of each word from dataset and then used as input feature in combination of other well-known features namely, word embedding, sentiment features, and multiple phrase level lexicons that identify positive and negative contextual polarity of sentiment expressions. An extensive set of experiments were performed for detecting cyberbullying behavior in binary scheme (either cyberbullying behavior exists in the tweet or not) and multi-classification scheme (low, medium, high, or none) to detect severity in tweets. Main focus and contribution of the current study was to provide systematic way to apply level of severity in cyberbullying behavioral text using multi-class classification. We focused on the binary classification and did not highlight the systematic procedure to go about detecting cyberbullying severity. Moreover, aim of our study was to compare well-known approaches rather than results from their datasets.

**6.4 PROJECT APPLICABILITY ON REAL WORLD APPLICATIONS :**

* In the last two decades, a variety of different ML techniques and feature selection algorithms have been widely applied to cyberbullying detection, predictions and blocking. Asynchrony and optional anonymity are characteristic of online communication as we know it today; it heavily relies on the ability to communicate with people who are not physically present, and stimulates interaction with people outside of one’s group of close friends through social networks.
* This model (given that it gets accurate data and prediction after choosing the correct algorithm and proper use of feature engineering) can be used in blocking malicious posts and comments in Twitter. We propose a versatile framework in which one can employ different machine learning algorithms to successfully distinguish between offensive and non-offensive content, while aiming to minimize the number of false positives.
* This model will help in blocking all the offensive posts and comments on the user’s profile in order to prevent the spread of cyberbullying, allowing the user to enjoy better content.

**6.5 INFERENCE:**

Harassment occurs online as a result of cyberbullying, and the consequences are grave. It occurs in a variety of forms and may be found in text format on most social media platforms. There is little doubt that over 1.96 billion of them would be forced to participate in some form of social operation. However, the coming decade will be fraught with challenges, and clients' online conduct will be scrutinised. Expanding the number of incidents of provocation and harassment, as well as the number of casualties, has proven to be a challenging task. Smart frameworks are required for programmed discovery of such episodes. A great number of recent studies have used typical machine learning models to address this issue, and the majority of the models developed in these studies can be scaled up to a single social network at a time. New approaches have been developed to identify digital harassing incidents, claiming to be able to overcome the limitations of traditional models and improve the discovery process. Despite the fact that a variety of old-school methods are available to govern the incident, the necessity to correctly order the torturing is still lacking. With the use of machine learning and language preparation, we were able to successfully filter harassing in the virtual world and stop the violent conclusion. A mechanism is developed to give cyberbullying a dual classification. Our method makes use of a novel pipeline concept for content analysis and prediction. For testing, a scraped dataset is used, and our approach is compared to other current methods and shown to have greater precision and grouping.

**CHAPTER-7: CONCLUSIONS AND RECOMMENDATION**

**7.1 Outline**

• Have the right data. This is the fuel of machine learning. The data must be

representative, relevant to the current malware landscape and correctly labeled

when needed. We became experts in extracting and preparing data and training our

algorithms. We made an efficient collection with billions of file samples to empower machine learning.

• Understand theoretical machine learning and how to apply it to cybersecurity.

We understand how machine learning works in general and keep track of state-of-the-art approaches emerging in the field. On the other hand, we are also experts

in cybersecurity and we recognize the value each innovative theoretical approach

brings to cybersecurity practices.

• Understand user needs and be an expert at implementing machine learning

into products that help users with their practical needs. We make machine learning

work effectively and safely. We build innovative solutions that the cybersecurity

market needs.

• Build a sufficient user base. This introduces the power of ‘crowdsourcing’ to

detection quality and gives us the feedback we need to let us know if we are right or wrong.

**7.2 Limitation/Constraints of the System**

The detection of Cyber-Bullying is so difficult in today’s era. Bullying is most apparent in younger age groups through direct verbal outings , and more subtle in older groups, mainly manifested in more complex social dynamics such as exclusion, sabotage, and gossip. To understand the task of cyberbullying detection as a specific domain of text classification, one should consider the full scope of the register that defines it which lags in every Cyber - Bullying Detection application / Software.

While some roles clearly show from frequent interaction with either a positive or negative sentiment , others might not be observable through any form of conversation , prove too subtle, or not distinguishable from other roles.

**7.3 Future Enhancements**

**Use a wider dataset**

In the future, we plan to apply our method to a larger dataset. We believe that by using a larger dataset, the performance of our method can be enhanced.

Increasing the number of channels when using a large dataset could improve the performance of the framework. A large dataset can also help to optimize the weights and other parameters of deep and large neural networks. Furthermore, we will also plan to test our proposed framework with tweets in different languages.

When applying our model to a real industrial scenario, it is hard to evaluate how this model performs since real labels are unknown all the time. Twitter can invite some users to label tweets by simply clicking ”hate” or ”offensive” when looking through their twitter page daily. Based on labeled data, Twitter is able to evaluate and thus improve the model. When it comes to application, the model should be updated periodically with tons of new tweets being generated because they can lead to new features, like new prejudiced terms. As a result, the model can always change.

**Evaluating model based on objectives**

A model can be evaluated based on different objectives. A single model cannot always achieve its best performance no matter what the business objective is. When a new different business scenario occurs, the firm should change the evaluation metric with it. For example, if Twitter wants to add a filter function for the users. An evaluation metric which weights precision more than recall might be more useful since the user experience will be severely undermined if the wrong twitter is filtered.

**Ethical consideration**

There is no absolute criterion to judge whether a tweet is hate speech, offensive speech or neither and labels depend on certain people’s opinion. Ethical problems may exist, for example, if Twitter forces users to filter all the tweets that they think may include hate speech or offensive language and leaves the rest to the users, freedom of speech can be an issue.

**RELATED WORK INVESTIGATION**

User products that implement machine learning make decisions autonomously. The

quality of the machine learning model impacts the user system performance and its

state. Because of this, machine learning-based malware detection has specifics. Outside the malware detection domain, machine learning algorithms regularly work under the assumption of fixed data distribution, which means that it doesn’t change with time. When we have a training set that is large enough, we can train the model so that it will effectively reason any new sample in a test set. As time goes on, the model will continue working as expected. The related work investigation and comparison goes as follows:

* Large representative datasets are required:

It is important to emphasize the data-driven nature of this approach. A created model depends heavily on the data it has seen during the training phase to determine which features are statistically relevant for predicting the correct label.

* The trained model has to be interpretable:

Generalizing this, we must train our models on a data set that correctly represents the conditions where the model will be working in the real world. This makes the task of collecting a representative dataset crucial for machine learning to be successful. Most of the model families used currently, like deep neural networks, are called black box models. Black box models are given the input X, and they will produce Y through a complex sequence of operations that can hardly be interpreted by a human. This could pose a problem in real-life applications. For example, when a false alarm occurs, and we want to understand why it happened, we ask whether it was a problem with a training set or the model itself.

* False positive rates must be extremely low:

The interpretability of a model determines how easy it will be for us to manage it, assess its quality and correct its operation. False positives happen when an algorithm mistakes a malicious label for a benign file. Our aim is to make the false positive rate as low as possible, or zero. This is not typical for a machine learning application. This is important, because even one false positive in a million benign files can create serious consequences for users. This is complicated by the fact that there are lots of clean files in the world, and they keep appearing.

* Algorithms must allow us to quickly adapt them to malware writers’ counteractions:

To address this problem, it is important to impose high requirements for both machine learning models and metrics that will be optimized during training, with the clear focus on low false positive rate (FPR) models. This is still not enough, because new benign files that go unseen earlier may occasionally be falsely detected. We take this into account and implement a flexible design of a model that allows us to fix false-positives on the fly, without completely retraining the model.

**REFERENCES**

1. **Automated Hate Speech and Offensive Language Detection** [**https://www.researchgate.net/publication/314942659\_Automated\_Hate\_Speech\_Detection\_and\_the\_Problem\_of\_Offensive\_Language**](https://www.researchgate.net/publication/314942659_Automated_Hate_Speech_Detection_and_the_Problem_of_Offensive_Language)
2. **Collaborative Detection of Cyberbullying Behavior in Twitter Data Amrita Mangaonkar, Allenoush Hayrapetian, Rajeev Raje** [**(PDF) Collaborative detection of cyberbullying behavior in Twitter data (researchgate.net)**](https://www.researchgate.net/publication/308866973_Collaborative_detection_of_cyberbullying_behavior_in_Twitter_data)
3. **A Review of Automated Detection Methods for Cyberbullying**

[**https://ieeexplore.ieee.org/document/8601278**](https://ieeexplore.ieee.org/document/8601278)

1. **Optimized Twitter Cyberbullying Detection based on Deep Learning**

[**https://ieeexplore.ieee.org/document/8593146**](https://ieeexplore.ieee.org/document/8593146)

1. **A Study of Cyberbullying Detection Using Machine Learning Techniques**

[**https://ieeexplore.ieee.org/document/9076550**](https://ieeexplore.ieee.org/document/9076550)