Hate Speech Detection Using Machine Learning Techniques

by

Tahbib Manzoor
17101147
Md. Wahidur Rahman Araf
17101404
Monjurul Sharker Omi
18301017
Arpan Das Abir
18101526
Tanvir Ahmed Abir
18301060

A Thesis submitted to the Department of Computer Science and Engineering
in partial fulfillment of the requirements for the degree of
B.Sc. in Computer Science

Department of Computer Science and Engineering
Brac University
May 2022

Declaration

It is hereby declared that

1. The thesis submitted is our own original work while completing degree at Brac University.
2. The thesis does not contain material previously published or written by a third party, except where this is appropriately cited through full and accurate referencing.
3. The thesis does not contain material which has been accepted, or submitted, for any other degree or diploma at a university or other institution.
4. We have acknowledged all main sources of help.

Student's Full Name & Signature:

Tahbib Manzoor
17101147

Araf

Md. Wahidur Rahman Araf
17101404

Omi

Monjurul Sharker Omi
18301017

Arpan

Arpan Das Abir
18101526

Tanvir Abir

Tanvir Ahmed Abir
18301060

Approval

The thesis titled "Hate Speech Detection using Machine Learning Techniques" submitted by

1. Tahbib Manzoor (17101147)
2. Md.Wahidur Rahman Araf (17101404)
3. Monjurul Sharker Omi (18301017)
4. Arpan Das Abir (18101526)
5. Tanvir Ahmed Abir (18301060)

of Spring, 2022 has been accepted as satisfactory in partial fulfilment of the requirement for the degree of BSc. in Computer Science and Engineering on May,2022.

Examining Committee:

Supervisor:
(Member)

Faisal Bin Ashraf
Lecturer
Department of Computer Science and Engineering
Brac University

Thesis Coordinator:
(Member)

Md. Golam Rabiul Alam, PhD
Asscociate Professor
Department of Computer Science and Engineering
Brac University

Head of Department:
(Chair)

Sadia Hamid Kazi, PhD
Chairperson and Associate Professor
Department of Computer Science and Engineering
Brac University

Ethics Statement

The proposed paper is a noble work. Furthermore, the members hereby and genuinely declare that this was done based on the conclusions of our exhaustive investigation. All of the resources utilized are correctly recorded and cited in the report. This research work, in whole or in part, has never been submitted to another university or institution for the granting of a degree or for any other reason.

Abstract

Social media and sharing platforms are improving in tandem with the internet. Although, humanity has benefited from these platforms in a variety of ways. However, many people were faced with a variety of obstacles and conflicts while using social media. As a result, the usage of derogatory language and hate speech has skyrocketed, posing a major threat. As a result, many varieties of machine language have been developed to overcome this challenge. Hate speech is defined as the use of slang or insulting phrases directed against a specific person, race, or any religion. Hate speech and other hate content can be detected using the suggested methodology on platforms like Facebook. Our goal is to develop a model that can identify such actions with more precision so that our current and future generations are not subjected to this scourge.

Keywords: Social Media, Hate Speech, Detection, Methodology

Dedication

We dedicate this paper to all the cyberbullying victims and to all the Enthusiasts who are fighting everyday against cyber bullying and trying to create a nontoxic and hatred free environment for our next generation.

Acknowledgement

To begin, We would want to offer all appreciation to the Almighty, because of whom our thesis was finished without any significant interruptions. Then we want to acknowledge our supervisor Mr. Faisal Bin Ashraf sir without his valuable guidance and feedback we wouldn't be abale to come this far.

Disclaimar

In the following paper there are some abusive words and sentences shown as sample of Dataset in the paper some readers may find it offensive .

Table of Contents

List of Figures

Chapter 1

Introduction

1.1 What is hate speech?

Despite the fact that there is no precise definition of hate speech and no characterization which is considered as hateful or abusive. But to generalize, any kind of communication writing that attacks or criticizes any religion, race, gender or any individual is considered as hate speech. As the internet progresses so do many social platforms. The breeding and spread of hateful content is heavily influenced by social media and other online platforms. The detection of this hate speech is indeed very challenging, as there are no exact definitions and there are various agreements and disagreements on what the hate speech is. As for example the word "Nazi" which indicates the German Army which fought in World War II. But many social platforms think this as an attacking or hateful phrase.

1.2 Hate Speech and Crimes

Some societies are full of hate crimes, it is nothing new. Moreover, social platforms are playing a huge role in these sorts of crime. As for instance, the terror attack that happened in Christchurch, New Zealand was broadcasted live on Facebook [1]. As there are also some anonymous platforms people take them as liberty and start to write whatever they want. The American Bar Association said that in the United States of America, hate speech is legal and will be protected by the first amendment, although not it directly calls for violence [2].Even during national elections in the USA the hate content was reported in social media and it resulted in some riot with the police. Hatzipanagos (2018) in her article posted in Washington post stated that White supremacist groups utilize social media to propagate their message because it allows them to incubate and spread their hate online. Also when their rhetoric reaches certain people, their online message can turn into real life violence. [3]

1.3 Hate Speech Case Ratio

Various incidents of hate speech can be found in various geographic places and platforms. Because the majority of minority groups live in these areas, they are subjected to hate speech and hate violence both online and in their daily lives. According to a survey conducted by the Council on Foreign Relations, the majority of individuals in North America believe they have the right to make offensive statements to minority groups. As a result, the minority groups who live there are more vulnerable. This has an impact on both on and off the internet.

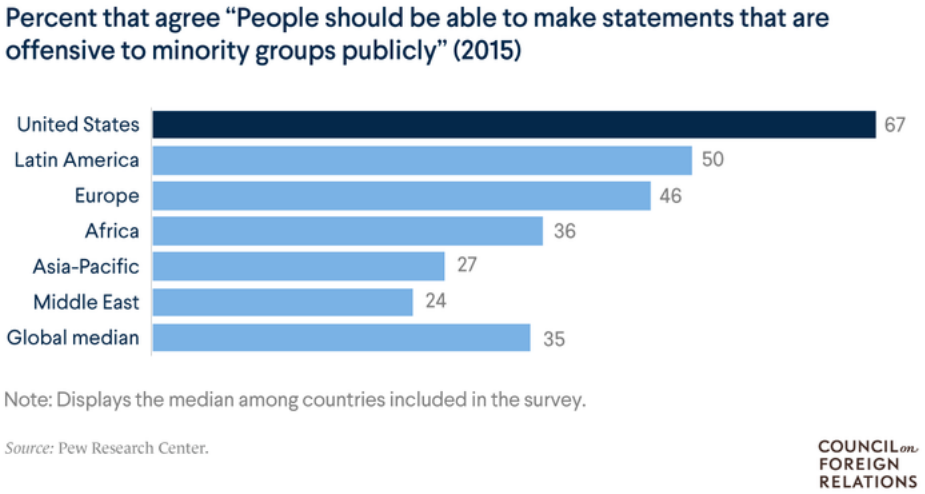


Figure 1.1: Regions where people approve hate speech towards minority groups.[4]

Another poll conducted by the Safe home group demonstrates the vulnerability of Twitter in terms of hate speech and freedom of expression. It is a rich mine for hate speech data analysts due to its open API. From 2008 to 2016, the enjoyment of hate speech has risen dramatically, as shown in the graph below.

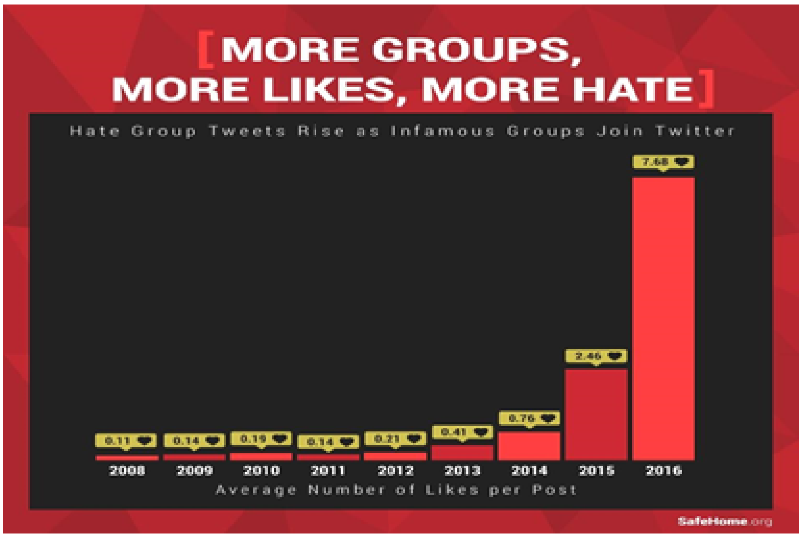


Figure 1.2: Increasing hate speech cases on Twitter [5]

1.4 ML and Hate Speech

As previously noted, the manual process of identifying and removing hate speech content takes a long time also its work intensive. So here comes the ML and the algorithm to solve the problem. As the online platforms are vast and many users act or use the platform simultaneously there prevails concern regarding wide spread of the hate speech for this automatic hate speech detection has a strong motive.

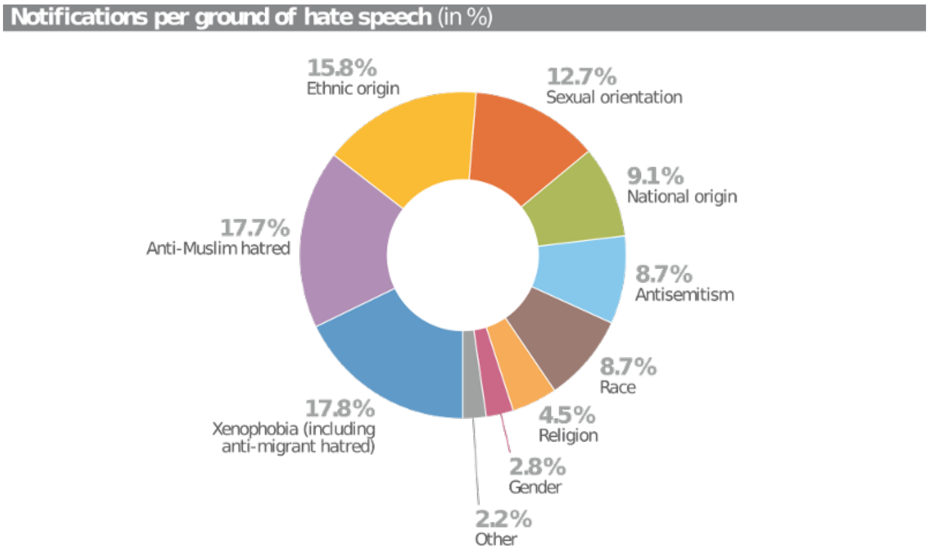


Figure 1.3: Percentage of hate speech removed by the platforms that had signed the EU Code of Conduct [6]

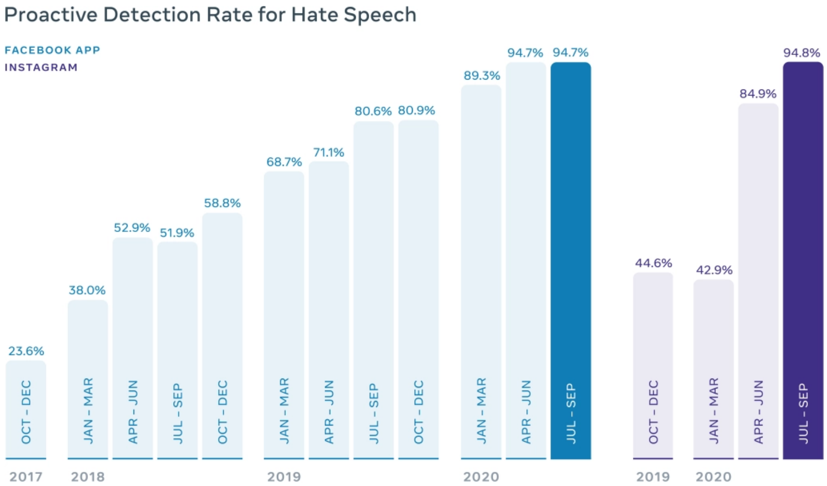


Figure 1.4: Detection Rate for Hate Speech of Instagram Facebook [7]

If we analyze the above figure we can see that with the increase of years the rate of online hate content also increased and ML played a vital role in predicting the hate content and also in removing them.

1.5 Problem Statement

As previously mentioned, detecting hate speech or hate content is indeed a challenging task as the definition varies from source to source. Some internet users face various bullying and cyber hatred due to their race or ethnicity. Although various algorithms have been implemented to detect those hateful comments or contents yet, some biases or lacking prevails. Sometimes the A.I or the algorithm fails to distinguish between the hate content and sometimes some general speeches, which are also considered as hate speech. This results in irrelevant bans from social platforms. Our aim is to classify the hatred comments into division such as Appearance, Racism, Sexual Abuse, Slang, Religious affiliation and so on. We propose a model which will work on a data set and it will be able to detect what kind of hate speech it is. Our proposed model will identify and also categorize whether the speech is about sexism, racism or other types of abuse.

1.6 Research Objectives

Our main objective of the paper is to-

- Build a machine learning based approach to detect hate speech in social media platforms.
- Work with both Multiclass & Binary Dataset.
- Categorize the hate Content according to the target victim group.
- Bring more precision or more accuracy in detection by distinguishing between hate speech and free expression.
- Focus on faster prediction hence more time efficiency.

Chapter 2

Literature Review

The extensive and comprehensive study on hate speech detection done by scholars all around the world is large and encyclopedic.

Many researchers around the world are researching to find a more accurate hate speech detecting system.Students of Sukkur IBA University, Pakistan - Sindhu Abro, Sarang Shaikh, Zahid Hussain Khand and Zafar Ali (2020)[8] published a paper on Automatic hate speech detection using machine learning. In that paper they stated that, humans had been benefited with the increase of social media and platforms but it has also given rise to various challenges and hate speech is one of them. Detecting hate speech manually is tedious, time consuming, and extremely difficult. According to the author, detecting hate speech might be difficult as there are no universal definition of hate speech. Therefore, the author suggested an Ml algorithm with eight ML classifiers and three feature engineering techniques on standard datasets for resolving the difficulties. In the first place, data is drawn from a publicly available dataset of hate speech tweets, which has been divided into three main categories which are hate speech, offensive and non-offensive. After Preprocessing, the data was followed by use of three feature engineering techniques: n-gram with TFIDF, Word2vec, and Doc2vec. The classifiers were used: NB, SVM, KNN, DT, RF, AdaBoost, MLP, and LR. After performing the experiment, best result were achieved by SVM and AdaBoost since threshold function was used by SVM for data seperation. Moreover, NB,MLP, KNN and DT classifiers performed the lowest.The drawback of the paper was that in some cases the hatred word was labelled as offensive.

Another amazing research "Exploring Hate Speech Detection in Multimodal Publications" done by Raul Gomez, Jaume Gibert, Lluis Gomez, and Dimosthenis Karatzas (2019)[9] from Generalitat de Catalunya and the Universitat Autònoma de Barcelona (UAB), attempted to solve the problem of hate speech identification in multimodal publications made up of text and images. The content is combined with a specific visual, some of these multimodal publications are only hate speech. This is especially true on Twitter, where multimodal messages consist of a picture and a short word, which is often insufficient to appraise them. As a result, they attempted to combine visual and textual data analysis. They constructed the MMHS150K dataset, which is a manual annotated multimodal hate speech dataset consisting of 150, 000 tweets, each with text and a picture. The dataset was given the name MMHS150K, and it was made publicly available online. A CNN and an

RNN are trained together in this procedure to learn a shared embedding space from aligned multimodal data. The goal of this project is to create a hate speech detector that uses both textual and visual data to detect hate speech. They evaluated three models: a Feature Concatenation Model (FCM), a Spatial Concatenation Model (SCM), and a Textual Kernels Model to see how a multimodal context can improve performance when compared to a unimodal context (TKM). They're all CNN+RNN models with three inputs: the image of the tweet, the twitter text, and the text that appears in the image (if any). Despite the fact that models trained solely on images have been shown to be beneficial for hate speech detection, the suggested multimodal models are unable to outperform textual methods. These models, on the other hand, gradually learn to rely almost totally on text input for prediction, producing results that are strikingly comparable to textual models. TKM is one of the multimodal models that has shown good performance. Noisy data, Complexity and diversity of multimodal relations, Small sets of multimodal examples are the reasons why they do not perform well. They looked at the difficulties of the suggested task and dataset. Given that the majority of content on social media currently is multimodal, they strongly believe that this research should be pursued.

Another profound research published on "A Lexicon-based Approach for Hate Speech Detection" by the students of Central South University Changsha, China - Njagi Dennis Gitari, Zhang Zuping, Jun Long and Hunan University, China - Hanyurwimfura Damien (2015)[10]. They explored the potential of developing a classifier that may be used to identify hate speech in online forums and blogs. They suggested a model classifier that can not only recognize subjective sentences, but can also detect and score the polarity of the expressed sentiment . When it came time to develop the classifier for hate speech detection, they recommended creating a vocabulary that would be used as input. For starters, the researcher seeks to distinguish between words that express subjective sensations and those that do not. How to create a vocabulary for hatred using the rule-based approach, which blends subjective features discovered in phrases with semantics learned directly through corpus analysis. After that, a sophisticated classifier analyzes textual texts for hate speech using terms from their lexicon. This is followed by an advanced classifier that uses characteristics from their vocabulary to detect hate speech in written documents. Subjectivity identification, objective sentiments, rule-based approach, data collection, semantic characteristics, SUBJCLUE and SWN Lexicons Corpus. The only drawback to this approach is the lack of available data. Regular updates are required for the dataset. In today's culture, hate speech and the tactics used to spread it, are continuously developing. Because it's a dataset built on top of a Lexicon, the resource has certain limitations.

The paper is based on the effectiveness of detecting hate speech in Bengali language using some methods. Here in this paper - Md. Nur Hossain,Abdullah Al Asif, Anik Paul,Amit Kumar Das (2021)[11] have focused on the very alarming issue of present times which is spreading negativity in the social media and the detection of hate speech to create a healthier environment in the social media for the future generation. Again, for this kind of evil practices in the social media, people get often bullied according to their color, religion, sex and political views and many other aspects. That's why the authors here mentioned some of the methods to detect hateful speech in Bengali language. Here, they have used a large dataset which has 7,425 Bengali comments that consists of seven different types of hate speeches, and has been used to train and evaluate this model. And to extract and encode local features from the comments, they have used 1D convolutional layers. To collect comments, they have prioritized popular public pages of celebrities, politicians, models, actors, singers, news portals, and players because somehow, they represent the national community and culture to the outside world and people has many mixed feelings towards them and that is why it is the best place to find and verify hateful scenarios. Then, they collected Facebook comments as data using Facebook graph API and categorized them as hateful or non-hateful through tokenization, Dataset creation, Bangla Emote Module, TF-IDF vectorization, word embedding, stemming, Machine learning algorithms like LSTM and Gated Recurrent Unit (GRU) based decoders with an improved model by using recurrent neural network (RNN) with Attention Mechanism - these methodologies. Among them, 6020 comments were hateful and they are labeled as ethnical attack, Aggressive, Hate speech and religious hatred. The hateful comments were put into four groups, and the non-hateful comments were put into two groups: Religious Comment and Suicide Comment. Political commentary is said to include both hateful and non-hateful political remarks. For a better understanding of hateful speech, seven classes were made. Above mentioned three encoder–decoder algorithms, he has found 74% accuracy for LSTM and GRU based decoders and the best accuracy (77%) in RNN with attention-based decoder. Further, the authors had shown the precision after running the tests with each methodology and got the highest precision of success to detect hate speech through Recurrent Neural Network (RNN) with Attention Mechanism with high recall (0.75), high f1-score (0.78) and high precision (0.78), while LSTM based one achieved recall (0.71), precision (0.72), f1-score (0.72) and GRU one achieved recall and f1-score (0.69) and (0.70) in precision. They added that as there are so many words in a language and words can be manipulated differently in different sentences, it is hard to detect 100% of hate speeches with any methods. They suggested using the Data Parsing Model to make this research more useful. Last but not the least, people nowadays post various types of photo comments that also has some negativity in itself, so they added it would be a huge step if we could detect those photo comments as hateful or not for our future generation to grow with more positivity in themselves and a healthy and helping environment for all in the social sites.

HateXplain: A Benchmark Dataset for Explainable Hate Speech Detection, a research study released in December 2020[12], demonstrated some remarkable results in hate speech detection. They also labeled their findings as hate, offensive, and regular utterances. Firstly, they gathered a Dataset that included a variety of hate speech examples, such as tweets and Gab posts. They also requested that Amazon Mechanical Turk workers annotate these postings in three different ways. To emphasize the length of the text, they employed human-annotated data. The data was annotated by Amazon Mechanical Turk workers. They worked on a smaller collection of training data to annotate. BiRNN, CNN-GRU, and BERT are some of the models used. In terms of performance metrics, BiRNN-HateXplain [LIME Attn], BERT-HateXplain [LIME Attn] 15) performs somewhat better. In comparison to BiRNN-Attn [LIME Attn], BiRNN-HateXplain [LIMEAttn] has a higher acceptable score for all plausibility metrics and comprehensiveness. The BERT-HateXplain

[LIME] model appears to be the most accurate. For sufficiency, CNN-GRU showed outstanding performance.

Chapter 3

The Dataset

3.1 Dataset Type

We will be working on both Multiclass & Binary Dataset.

3.2 Language of Dataset

The datasets comment that we collected are all Bengali transliterated comment from English. A sample of the dataset along with the english translation is shown in the table below.

Transliterated Bengali Comment	**English Translation**	**Type**
Tui ekta magi	You are a whore	Sexual
Discount ki pabo?	Will I get a discount?	Not Hate
Hindu Gulare mar	Kill the Hindus	Religious
Apnar jama eto choto ken?	Why is your dress so small?	Appearance
Khankir pola pagol hoi gace	The son of a bitch has gone mad	Slang

Table 3.1: Multiclass Dataset with Translation

Transliterated Bengali Comment	**English Translation**	**Type**
Tui ekta magi	You are a whore	Hate
Discount ki pabo?	Will I get a discount?	Not Hate
Hindu Gulare mar	Kill the Hindus	Hate
Apnar jama eto choto ken?	Why is your dress so small?	Hate
Khankir pola pagol hoi gace	The son of a bitch has gone mad	Hate

Table 3.2: Binary Dataset with Translation

3.3 Dataset Collection

As we are working on Transliterate Bengali comment data, this type of dataset was not available. So we decided to make one dataset that consists of Transliterate Bengali hate comments. We decided to use Facebook as the medium of data collection.

But there were some challenges due to the restriction of Facebook's graph API. We cannot extract comments directly from any post or page.Nevertheless,we collected the comments manually and labeled them accordingly.Our main goal was to find various hate comments like religious, appearance, sexual, etc.For this, we collected comments from various controversial news and public figures. As we mentioned previously that hate speech definition varies from person to person, so after collecting the dataset, we voted them accordingly, like what type of hate speech it is, then we took the max vote and finalized the labeling of the data. We named our dataset as "Pandulipi" which means manuscript in the Bengali Language.

3.4 Data Analysis

We collected around 5000 comments, and from that, around 2837 comments were labeled as hate comments and the rest 2163 comments were not hate comments. We labeled them according to type as for example, we used the label Appearance, Racial, Sexual, Religious, Slang and others, and Not Hate. Some comments were hateful, but those were controversial comments so we labeled them as others.We also made a copy of the dataset and labeled it as only hate and non hate.In our dataset, most of the data is in between 6-10 word range, which is 42.96%. Secondly, 32.86% data is in range 1-5 words. If we combine the First two ranges, 75.82% data is in between range 1-10. Apart from that, from range 11 to 50 have 24.02% data. Besides, 51 to 90 words have insignificant amounts of data which is 0.18%.Our goal is to work on both multiclass dataset and as well as Binary label dataset. Visual representation of dataset "Pandulipi" is shown below-

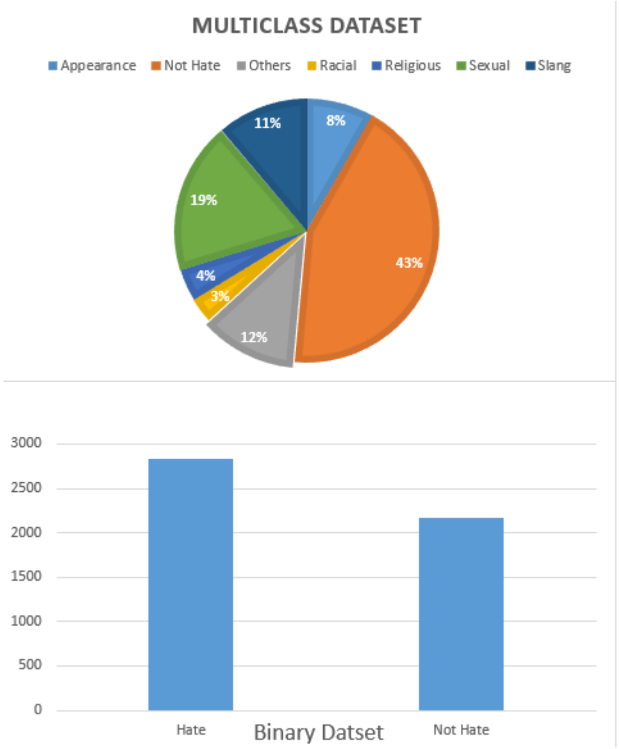


Figure 3.1: Representation of the Multiclass & Binary Dataset

3.5 Data Annotation

For further work, we labeled the data into different

- *Appearance*: This class includes hate speech which is related to someone's appearance or how they look.
- *Racial*: This class includes hate comments directed towards any race or any group.
- *Sexual*: Hate comments which are related to any sexual abuse or any sort of sexual discrimination.
- *Religious*: The hate comments which are made on any religion or to any religious aspects.
- *Slangs*: Any sort of informal or abusive comments which indicate hate, disrespect, or bullying belongs in the slang class.
- *Others*: Some hate comments which don't fall in either of the criteria, but it indicates hate we labeled as others.
- *Not Hate*: The comments which don't indicate any hate is in this not-hate section.

3.6 Data Pre-Processing

For further analysis and for running the classifiers, we need to prepare the raw data and filter them. As the data is in transliterated Bengali, so there is no specific language for this data. As a result, there are no stop words as well. So first, we remove the punctuation and other unnecessary symbols. After that, we take all the sentences to lowercase, and then we tokenize all the sentences using the split function. After that, our data is ready and we process this data for further work. We then use TF-IDF and Bag of Words as the pre-processing technique then we pass it to the classifiers for further results.

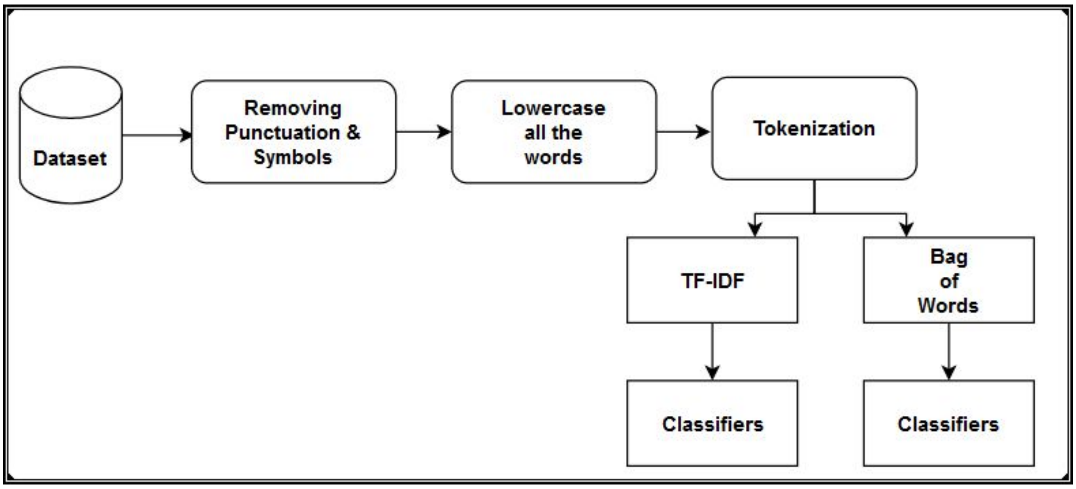


Figure 3.2: Data Pre-Processing and Procedure

3.7 Pre-possessing Technique

3.7.1 TF-IDF

TF-IDF is a analytical measure that helps us to determine how relevant a word is in a document.The abbreviation of TF-IDF is “Term Frequency Inverse Document Frequency” Automated text analysis relies heavily on this tool, as does Natural Language Processing, which employs machine learning techniques to assign numerical values to words (NLP). To begin with, TF-IDF was used to search for and retrieve information from documents when it was first developed. In practice, it works by increasing in comparison to the amount of times a term is used in a document, while decreasing in direct proportion to the number of documents that include the same phrase. Due to this, phrases that recur often in every text, such as this and what, and if, are assigned a low ranking despite the fact that they appear numerous times since they have little significance in that particular document.

There are two terms in TF-IDF. TF helps us determine how many times a word is used in a document by measuring its frequency. And dividing by its frequency, the number of words used is found. The second term IDF, calculates the logarithm of the number in a document and divides it by the number of documents in which the specified words are used. Term Frequency, measures how often the phrase is mentioned in a text.It is very likely, a phrase will appear considerably more times in large texts than in abbreviate documents, due to the fact that every document is distinct in length. For this the frequency which is counted per document, is divided by the total number of words used in a text for normalization.

$$TF = \frac{\text{Number of words used in document}}{\text{Total Number of words of the document}}$$

The importance of a phrase may be gauged using the IDF (Inverse Document Frequency) method. When calculating the TF, all terms are given equal weight and importance. However, it is well-known that some phrases, such as ”is,” ”of,” and ”that,” may occur often but have little significance. As a result, we must reduce the frequency of the phrases while increasing the frequency of the uncommon terms, as determined by the following computations:

$$IDF(t) = \frac{\text{Total number of documents}}{\text{Number of documents with term t in it}}$$

For instance, If we take 3 sentences like sentence1 : Good boy; sentence 2:Good girl; sentence 3: Boy girl good. To find out the Term Frequency (TF) of Good, Boy and Girl in three sentences, we will find how many times the word Good, Boy and Girl is used in every sentence and divide it by the number of total words in every sentence. As for the IDF, we use a log of the number of the total sentences , in our case, the total number of sentences is 3 and divide it by the number of sentences containing the specific word. Like for the Good, it will be log(3/3) as the total number of sentences is 3 and the word good is used in all 3 sentences. Now for the final output, we will multiply the TF with IDF.

3.7.2 Bag of Words(BOW)

Bag of Word can be defined as a natural language processing technique for producing fixed-length vectors that counts the amount of times each word occurs in a given text.Vectorization is a term used to describe this procedure. In the case of image processing, every pixel of the image is numbered to extract data from the image. Basically, the machine cannot understand an image as input. To make it understandable, it is converted to a set of numbers. Similarly, to understand the natural language, it needs to be converted into numbers. And this process is called vectorization. Bag of word is a vectorization process. It is because the text has no data about the pattern or words that has been eliminated,it is referred to as a “bag” of words. The model mainly focuses on whether or not recognized the terms used in a document, not where they are used. So considering a few sentences, it will determine how many times each word occurred in those sentences and give them a number.

Firstly the model splits every word from the sentence. All the punctuations will be discarded. Also, all the uppercase words will be converted into lowercase. Then it will count how many times a word appeared in a single sentence. For example, if there are 3 sentences, sentence 1 contains 3 words, for sentence 2 it is 5 and for sentence 3 the number of words is 8. In total, there are 16 words in the bag. Then the model assigns them a number depending on if that specific word appeared in the sentence or not. It creates a vector for each sentence. And that’s the core idea behind the bag of words. It uses the n-gram method, basically which takes consequently n numbers of words to avoid a similar meaning misconception. This occurs when two sentences with different meanings have similar words. For example, “This is a good job. I will not miss it for anything” and “This is not good at all” these two sentences have “This”, “is”, “not”, “good” in common. Which may identify both as negative meaning. By taking consecutively two or more words at a time the model can show a different outcome. However, there is an issue. If the variation of words is huge, then the matrix and vector length will be huge. The computational time will take much longer. Otherwise, it is a very simple technique for vectorization and tokenization.

Chapter 4

Methodology

4.1 GaussianNB

The complete form of GaussianNB stands for Gaussian Naive Bayes. It's a type of Naive Bayes Algorithm variation. The Bayes theorem is utilized to create a series of supervised machine learning classification algorithms which is known as Naive Bayes. Despite the fact that this technique is a simple classification technique, it is beneficial. In addition, the Bayes Theorem is utilized to determine conditional probability. In the studies of probability, this theorem is a helpful tool. Machine Learning makes use of it as well. There are three different kinds of Naive Bayes algorithms: Multinomial Naive Bayes, Gaussian Naive Bayes and Bernoulli Naive Bayes.

Gaussian Naive Bayes is a special kind of Naive Bayes Algorithm. When there are features which have continuous values or continuous inputs, GaussianNB will be used there specifically. And all the features also follow Gaussian distribution or Normal distribution. Assume that the data is characterized by a Gaussian distribution with no co-variance (independent dimensions) between dimensions to develop a simple model. The mean and standard deviation of the points within each label can be used to fit this model. That is all which is needed to define Gaussian distribution. This approach works with features and models that have a continuous value and follow a Gaussian (or Normal) distribution.

We commonly assume that the continuous values associated with each class are all distributed according to a normal (or Gaussian) distribution when working with continuous data. The Gaussian distribution's probability density function is -

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$$

The parameter µ here refers to the distribution's mean or expectation (as well as its median and mode), while the parameter σ refers to the distribution's standard deviation and the parameter σ^2 refers to the distribution's variance. This Gaussian function is a special kind of function that has a bell-shaped curved graph.

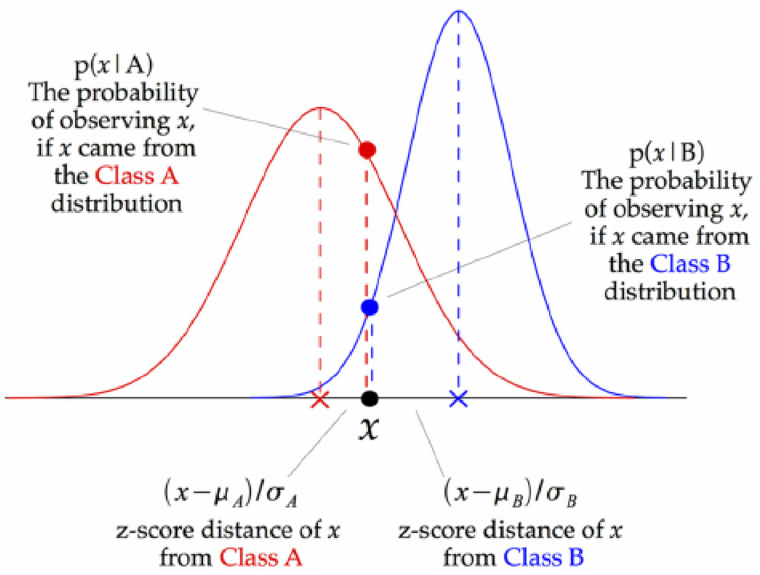

p(x | A)
The probability of observing x, if x came from the Class A distribution
p(x | B)
The probability of observing x, if x came from the Class B distribution
x
$(x-\mu_A)/\sigma_A$
z-score distance of x from Class A
$(x-\mu_B)/\sigma_B$
z-score distance of x from Class B

Figure 4.1: Illustration of how a Gaussian Naive Bayes (GNB) classifier works[13]

A Gaussian Naive Bayes (GaussianNB) classifier works as shown in the graph above. The z-score distance between each data point and each class mean is determined, which is the distance from the class mean divided by the class standard deviation. However, a crucial difference between GaussianNB and other classifiers emerges only when there are multiple input dimensions . Although other types of classifiers model the covariance between dimensions, The GaussianNB does not account for inter-dimensional covariance. As a result, we can see that the Gaussian Naive Bayes has a somewhat different manner of producing a result, but it can also be employed effectively.

As we are working with text-type data, we cannot feed the texts directly to our classifier. So we use the number of times a word appears in a single text data set (term frequency-TF) and the number of times that word appears across all text data sets (inverse document frequency-IDF). Following that, we may create feature vectors that represent text and contain the probability of the terms in the text appearing inside the texts of a specific category. The program can then calculate the likelihood of that text appearing in each of the categories. After that, we choose the category with the highest likelihood and use that to categorize our content.

4.2 Decision Tree Classifier

In statistics, data mining, and machine learning, The decision tree classifier is a type of predictive modeling approach. It is indeed a controlled machine learning algorithm. It makes decisions according to a set of rules, just like humans. In this algorithm, The categorization model in this approach constructs a decision tree. It can also be applied to problems involving classification and regression. In this treestructured classifier, internal nodes represent dataset attributes, branches represent decision rules, and each leaf node represents the final output.

In the Decision tree, the leaf node and the decision node are two different types of nodes. Here in the decision node, there remains a test and through that test it can make a decision. The test or decision is performed according to the features of a given dataset. A decision node can have numerous branches, while leaf nodes are the results of decision nodes that do not have any child branches.
To create a decision tree from a dataset, the CART Algorithm (Classification and Regression Tree Algorithm) is utilized. The root node is the topmost node in the decision tree, and there was given a dataset at the root, and then it is expanded on additional branches from the root and looked like a tree-type structure. A decision node asks a question depending on the features of the input dataset and then grows the tree into subtrees and leaf nodes based on the answer (Yes/No).

The decision tree is a recursive algorithm that shares decision-making logic that other algorithms, such as Neural Networks lack. The number of records and attributes in a given dataset determines the time complexity of decision trees. The decision tree is really not based on any assumptions about probability distributions. It has a high level of accuracy when dealing with high-dimensional data.

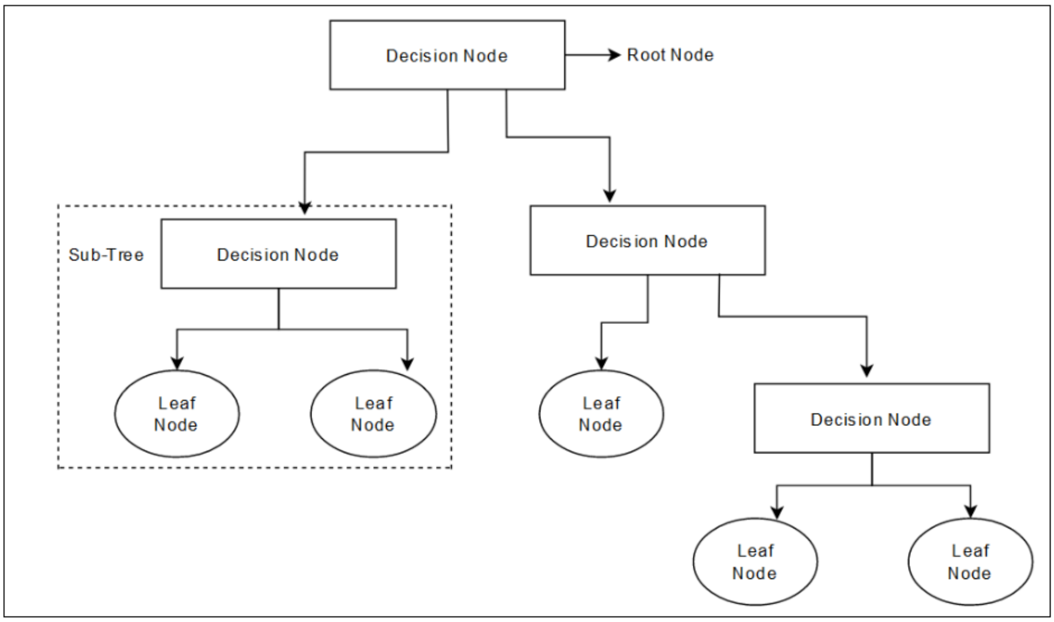


Figure 4.2: Decision Tree Classifier

Using Attribute selection measures(ASM), a decision tree finds the best attribute. The attribute is then turned into a decision node, and the dataset is divided into smaller subsets. After that, it creates new decision nodes recursively using subsets of the dataset, and the process is repeated until it reaches a point where it can no longer classify nodes, at which point the final nodes are referred to as leaf nodes. As

we are working with text type data and in our dataset there are features where we can easily implant a decision tree classifier to sort the data to a particularly valued

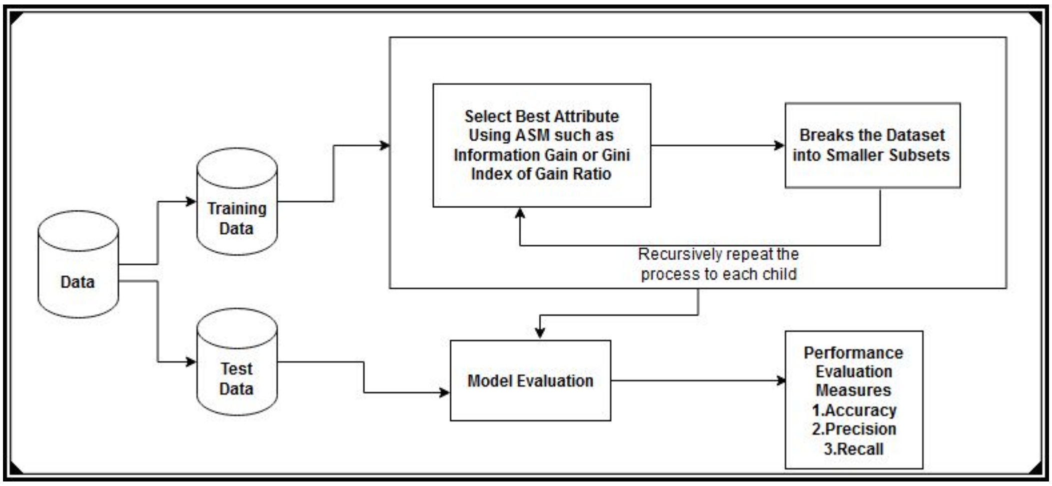

Data
Training Data
Test Data
Select Best Attribute Using ASM such as Information Gain or Gini Index of Gain Ratio
Breaks the Dataset into Smaller Subsets
Recursively repeat the process to each child
Model Evaluation
Performance Evaluation Measures
1.Accuracy
2.Precision
3.Recall

Figure 4.3: How Decision tree Classifier Works

attribute. However, the decision tree can sometimes have a lot of layers, making it a little more complicated. However, it may be highly useful for decision-making difficulties, and it is quite simple to grasp because it follows the same decision-making process that humans do.

4.3 AdaBoost

AdaBoost (Adaptive Boosting) is a prominent boosting strategy that tries to construct a robust classifier by the combination of many classification techniques in such a manner that each classification technique seeks to improve the categorization on data incorrectly classified by the weak classifier that came before it. It is a group method for training and deploying trees in a logical order. With the training data and weighted samples, a weak classifier is created. Its simplicity of use, real-time target detection performance, feature selection, and rapid training time all contribute to its success. Ada-Boost is mainly used to improve the performance of decision trees in binary classification situations. When we use boosting, we may create a more robust classifier by connecting together many weak classifiers.

A weak classifier is one that outperforms random guessing but still has trouble assigning classes to objects. These weak classifiers are known as “stump”. A stump is basically a small tree that has only two branches. Because each decision tree tends to be shallow models that do not overfit but might be biased, boosting approaches employ “stump” decision trees. An individual tree is taught to focus solely on the preceding tree’s flaws. Also, some stumps get some more say in the final classifiers. The weight of a sample that was misclassified by the prior tree will be increased, allowing the next tree to focus on accurately identifying the previously misclassified sample. AdaBoost is best for unbalanced datasets. However, it performs poorly when there is noise. AdaBoost is more difficult to train.

As we can see, AdaBoost makes multiple stumps from a single decision tree which makes every decision much prioritized in decision making, though a single stump has less say in the final decision. Initially, we set the weight of the sample equally.

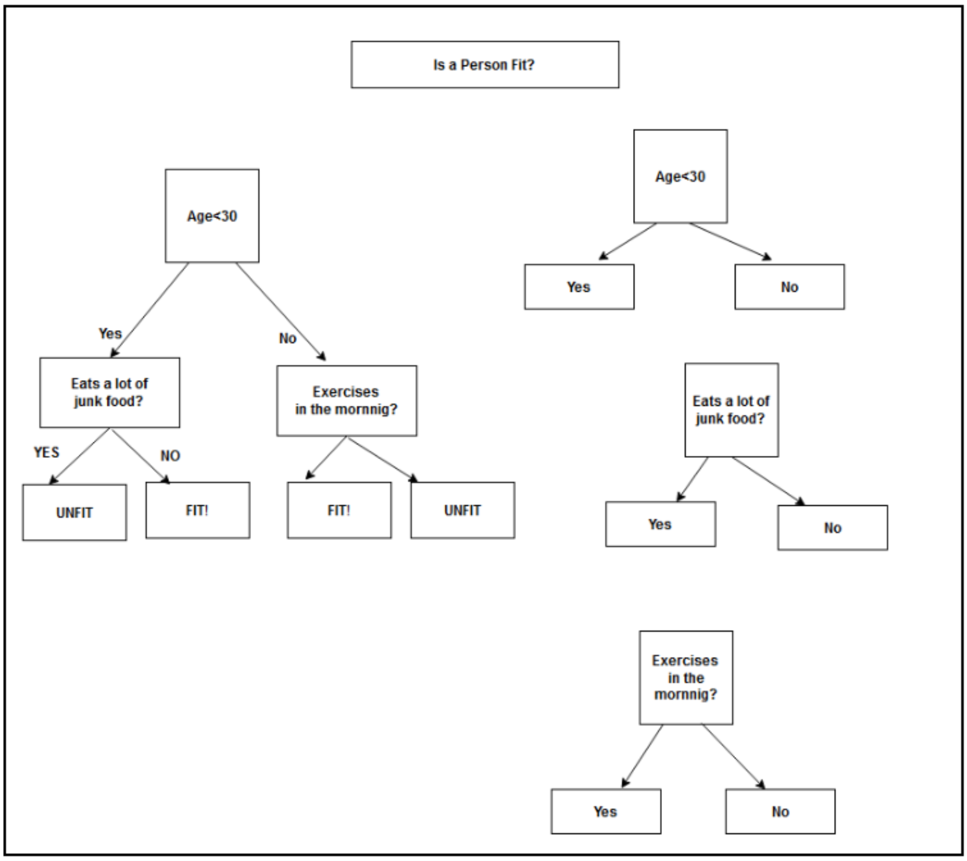


Figure 4.4: Seperated Stumps

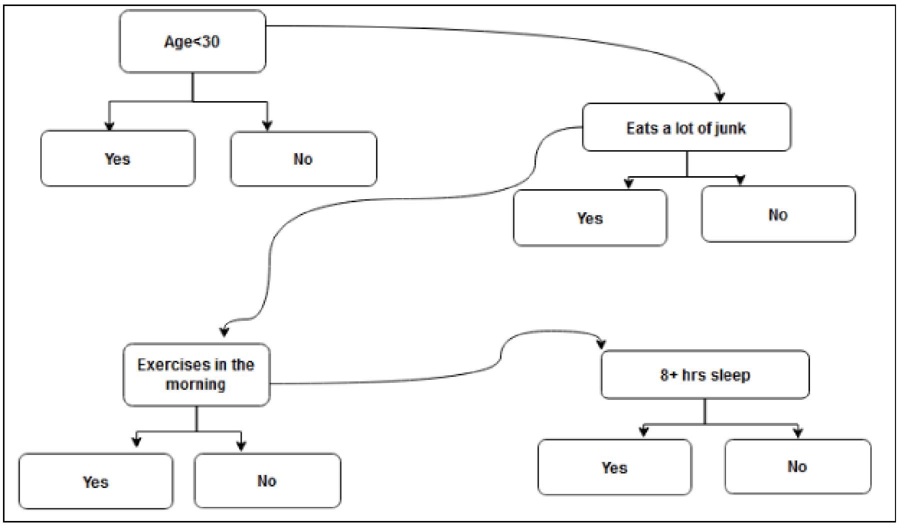


Figure 4.5: Connecting Stumps

The next stump will be affected by the errors of the previous stump error. We also check the fitness of the stumps based on their classification accuracy.
First, we consider the rows of our dataset as N.

$$x_i \epsilon R^n, y_i \epsilon \{-1, 1\}$$

Here, n is the attribute of our data. X is the set of data points. Y is to determine the classification is fit or unfit. We know initially, AdaBoost set each training sample the same weight to determine its importance in the training dataset. When the given weights are substantial, that set of training data points is more likely to influence the training set. In the same way, when the allocated weights are low, they have little impact on the training dataset.

$$w = 1/N \epsilon [0, 1]$$

Here w is the weight of each data. So if there is 5000 data, the weight of each data will be 1/5000. In this way, weight will be in between 0 and 1.

$$\alpha_t = \frac{1}{2} ln \frac{1 - TotalError}{TotalError}$$

Here alpha determines how much a single stump has to say to the final classification and total error is the total number of misclassifications divided by the number of training sets.

4.4 Logistic Regression

Machine Learning has quite a few algorithms for classification problems, like those Logistic Regression is one of them. This is a supervised classification model. This algorithm does classification in a given Dataset. There are mainly two types of logistic regression, one is Binary logistic Regression and the other one is Multinomial logistic regression. Binary logistic regression works with two types of data in a dataset. When there are more than two classes in data, in this case Multinomial logistic regression is used. As our dataset contains more than two classes, we will explain the Multinomial logistic regression. Multinomial Logistic Regression has some other names too,which are: multiclass logistic regression, softmax regression, polytomous logistic regression and multinomial logit. Multinomial logistic regression is a modified version of logistic regression model. But with this we can predict the class of a data from more than two classes of the dataset.

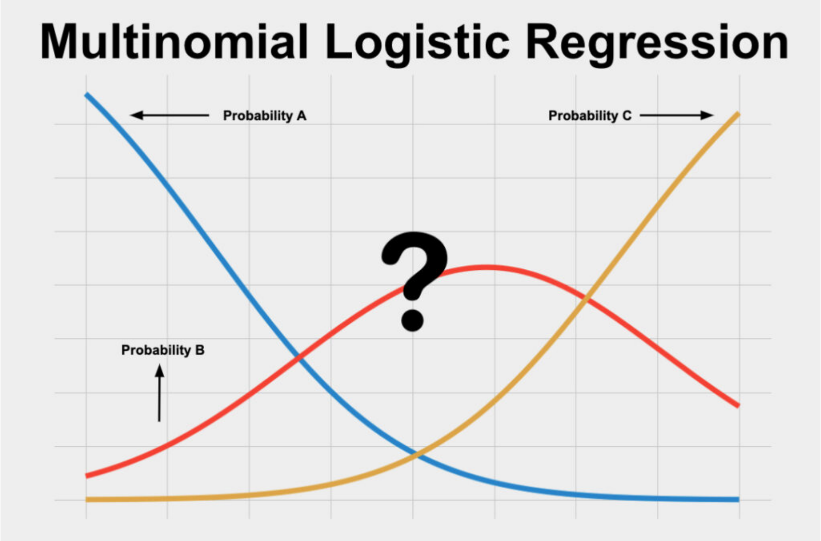


Figure 4.6: Multinomial Logistic Regression Example[14]

In this model, we can predict an outcome from multiple dependent variables categorically based on multiple independent variables. Before working with this model, we need to make sure that our dependent variables are nominal or ordinal variables. In our study, dependent variables of our dataset are nominal values which means categories are labeled by name. Now, there are two ways to solve this with Multinomial logistic regression. First one is “k models for k class”. The picture below shows, we have 3 classes. So, we build 3 models for 3 classes. In every model we will have two sub-section one is itself and the other one consists of other classes. Then by calculating the probabilities, we will predict the possible outcome.

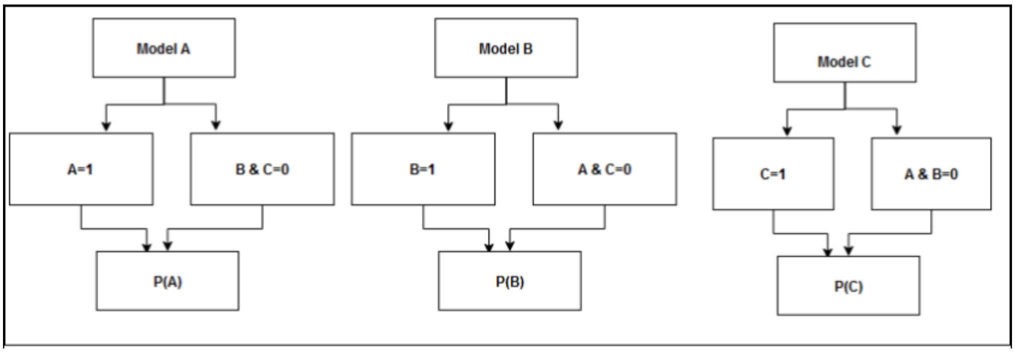


Figure 4.7: Example of “k” models for k class

Second one is “Simultaneous Models”. In this way we will develop k-1 models from the k classes. Let’s have 3 classes, we will develop 2 models. We will make a probability equation for the class A as the formula below us

$$(\frac{p(A)}{p(C)}) = a1 + b1x1 + ... + bnxn$$

$$(\frac{p(A)}{p(C)}) = exp(a1 + b1x1 + ... + bnxn)$$

$$p(A) = p(C) * exp(a1 + b1x1 + ... + bnxn)$$

Again we follow the same as classA for the classB like below

$$(\frac{p(B)}{p(C)}) = a2 + b1x1 + ... + bnxn$$

$$(\frac{p(B)}{p(C)}) = exp(a2 + b1x1 + ... + bnxn)$$

$$p(B) = p(C) * exp(a2 + b1x1 + ... + bnxn)$$

Since P(A)+P(B)+P(C)=1 then

$$p(C)*exp.exp(a1+b1x1+....+bnxn)+p(C)*exp.exp(a2+b1x1+....+bnxn)+p(C) = 1$$

$$p(C) = \frac{1}{1 + exp(a1 + b1x1 + ..bnxn) + exp.exp(a2 + b1x1 + ...bnxn)}$$

After all of this we get the predicted value for the data.

4.5 Random Forest Classifier

Random forest is a machine learning algorithm that is supervised that operates with a bunch of decision trees.In this approach, the algorithm creates a forest of decision trees at training time.As an output, we get the specific classification selected by most tree branches.To be specific, to generate a precise prediction, random forest creates a bunch of decision trees and blends them together and the majority of the tree selected by the random forest is the final decision.Random forest classifiers can be used both for classifiers and regression.

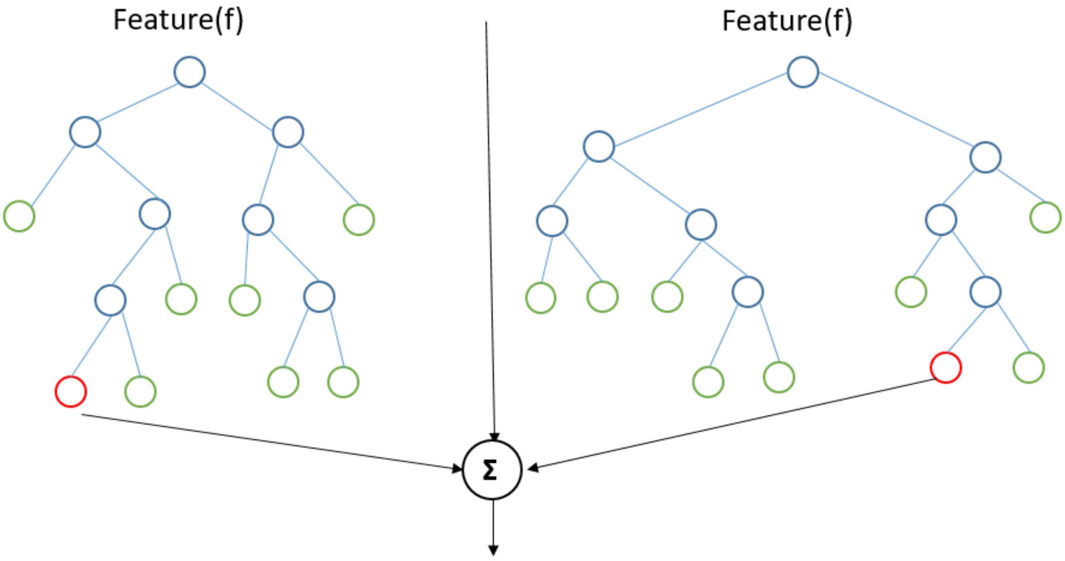


Figure 4.8: Summation of Best feature

Random forest classifier works like a decision tree. But in this case the algorithm adds randomness while spanning the decision tree. Features are a subset of the data, from the dataset.On the basis of features final classification is set in a dataset.The algorithm selects the best features from random features and starts spanning the tree.The decision tree, on the other hand,is used to identify the most essential features that influence the final classification of a particular data set. As a result, we receive a more diversified set of results, which enhances the classification's ultimate output. The forest, or more precisely, the collection of trees, can be extended further. Finally, the best classifications are chosen as the final result. Also random forests are really good to handle missing data.

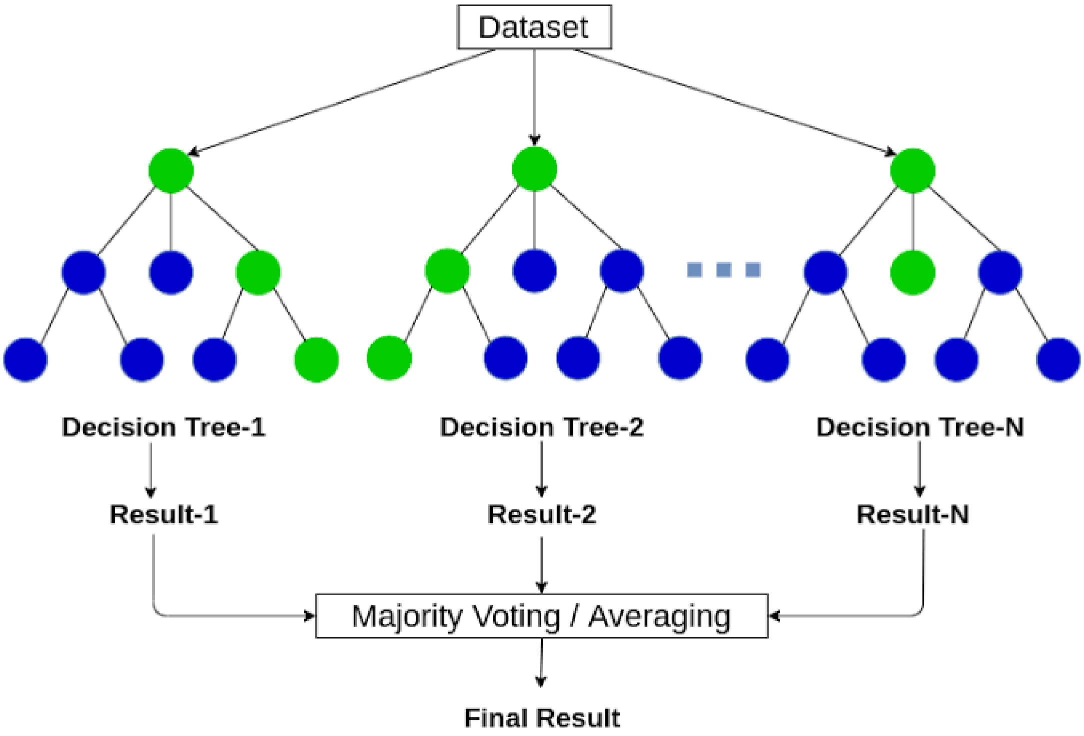


Figure 4.9: How Random Forest Selects Decision[15]

For example, during training time the algorithm will randomly span a tree based on random attributes. Each branch of the tree will provide a distinct classification.The algorithm will then choose the classification that the majority of the trees produce. Finally, the final classification created by those trees will be determined by a majority vote. As a random forest classifier combines all the results produced by different random trees it can solve the overfitting problem easily. Overfitting Refers to taking into account undesirable data from a dataset. It can offer accurate results for enormous data sets. The key benefit of random forest is that it can retain acceptable accuracy even if a substantial amount of data is missing, and it does not require scaling. The use of a random forest classifier has various drawbacks.The random forest algorithm can radically alter, if data changed slightly.This Algorithm's complexity is a significant concern. In comparison, Random Forest is more complex than other algorithms. In Addition, because of the enormous number of trees involved, training the model takes longer.

4.6 Multinomial Naive Bayes

In Natural Language Processing (NLP), a probabilistic learning strategy called "Multinomial Naive Bayes" is frequently used to learn about the structure of natural language (NLP). According to Bayes theorem, a text-tagging system may be able to predict the tag for a piece of document, such as message, mail or article. If a sample has more than one tag, the algorithm evaluates which tag has a higher possibility of being selected and then provides that result. In data analysis Multinomial NB is one of the favourite classifiers and it is one of the most used classifications.

Text data analysis and challenges involving numerous classes are both made possible by using the Naive Bayes method, which is a strong tool. Because the Naive Bayes theorem is built on the Bayes theorem notion, it is essential to grasp how the latter works before understanding how the former works.This theorem is developed by Thomas Bayes.

It works on a mathematical formula, which is :

$$P(A|B) = \frac{P(A) * P(B|A)}{P(B)}$$

P(A) is defined as the probability of occurring A in the document.

P(B) is defined as the probability of occurring B in the document.

$P(B|A)$ is defined as the probability of occurrence of prediction of B in the class of A. When determining the likelihood of a tag appearing in a document, this formula is used.
$P(A|B)$ is defined as the probability of events occurring on a document. (Depending on the condition related to the event)

4.7 SVM

Support Vector Machine is the complete form of SVM. It is one of the most used algorithms for ML.This is mostly used for problems with classification and regression. This approach is useful for a variety of applications such as face identification, picture classifications, text categorization, and many more. Because we are dealing with text data, we have chosen to employ this algorithm. The goal of the SVM method is to discover the best path or decisions limit for classifying the provided set of data into classifications, in order to make it easier to insert further data points in the intended form with in long term. Such as if we train SVM with a data set, which is sorted in a category.
After training, It will be able to identify the class of a data. The optimal decision boundary is called a hyperplane.

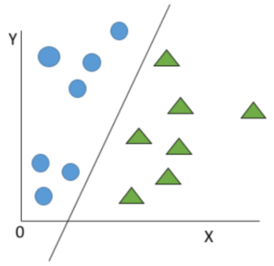


Figure 4.10: Graph of SVM

Hyperplanes are created by choosing extreme points or vectors. Support vectors are a term used to describe such extreme points or vectors. That's why the classifier is called as Support Vector Machine. In the picture above, two categories are being created with an optimal hyperplane.
We know that there are mainly two kinds of SVM. They are Linear Svm and Non-Linear Svm. The first one mainly used when there are two categories which can be separated by a simple linear line. The second is used when we can not classify data by a linear line. Now, we will see how to choose an optimal hyperplane for a dataset.

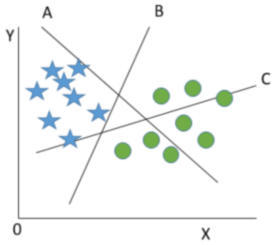


Figure 4.11: SVM Scenario-1

Here in the picture above, we will use Linear Svm and choose that hyperplane which will classify the data set in a better way. Now in the picture below, we will choose

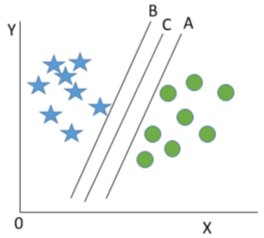


Figure 4.12: SVM Scenario-2

that hyperplane which will maximize distance between the data points that are closest to each category.

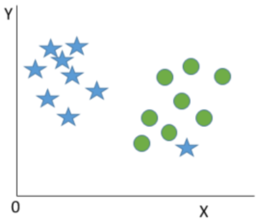


Figure 4.13: SVM Scenario-3

Again in the above picture, we can see that there is a star with the other side of the category. In this type of case to increase the margin, SVM ignores this kind of oddity.

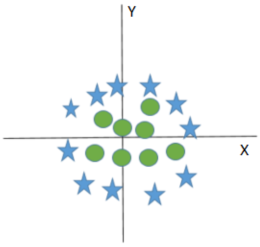


Figure 4.14: SVM Scenario-4

In this case we need to use Non-Linear SVM. Here,SVM will use the equation of the circle and categorize the data.
There is a special trick SVM uses to find a hyperplane. This is called Kernel(finds a pattern). It executes some pretty complicated data transformations before determining how to segregate the data using the labels or outputs you provide.

4.8 LSTM

Every time, humans don't start over with their thoughts from the beginning. As we are writing this article, we are making sense of each word depending on our comprehension of the words that came before it. In the end, we don't have to start over from scratch. Our thoughts are steadfast.
Recurrent Neural Networks (RNNs) are a form of Neural Network in which the output from the previous step is used as the input for the next phase in the network. For each input, it makes use of the same parameters since it performs the same work on all of the inputs or hidden layers in order to create the output.
LSTM's full form is Long Short Term Memory. It's used for sequence predicting problems. Among all other algorithms for sequence predicting problems this one is most effective. Along with it, LSTM is a modified version of RNN(Recurrent Neural Network). It can predict text, handwriting, genomes, and the spoken word. LSTMs are explicitly designed to avoid the problem of long-term dependency. Remembering information for extended periods of time is practically their default behavior; it is not something they have to work hard to learn.

LSTM is one of the most popular and common RNN architectures. Sepp Hochreiter and Juergen Schmidhuber came up with it as a way to solve the vanishing gradient problem. In their paper, they tried to resolve the long term dependencies problem. If the previous state influences a major effect in the current prediction that did not occur recently then the RNN model may not be able to predict the current state accurately. For instance, let's assume that we want to predict the next italicized word of the following sentence.”Robben is allergic to beef. He can't eat steak . Here, the context of a beef allergy might assist us in anticipating the meals that are prohibited from consumption including beef. However, if the context was not clear or stated about the allergy in more detail and complex sentences then it would be very difficult for RNN to learn and connect the information. To overcome this problem, the LSTM does have the ability to remove or add information to the cell state, carefully regulated by structures called gates; those are an input gate, an output gate and a forget gate. These gates control the flow of information and decide which data to remember and which data to forget by binary output.
First, we must pick what information from the cell state we are going to discard. This is the first stage of LSTM. Known as the ”forget gate layer,” this sigmoid layer is responsible for making this choice. It examines the values of ht1 and xt and produces a number between 0 and 1 for each number in the cell state Ct1 that is encountered. A 1 indicates that something should be kept totally, whereas a 0 indicates that something should be completely removed.

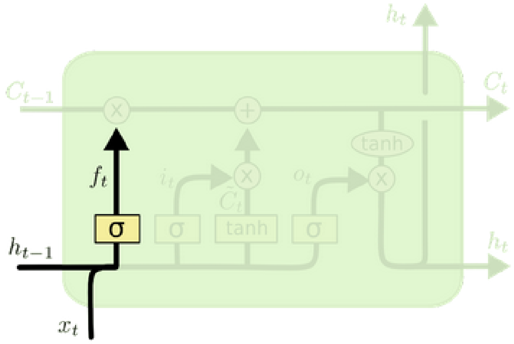


Figure 4.15: LSTM-1

$$f_t = \sigma(W_f.[h_{t-1}, x_t] + b_f)$$

The next step is to determine what additional information will be stored in the cell state as a result of this decision. There are two components to this. First, a sigmoid layer known as the ”input gate layer” determines which values are going to be changed. Following that, a tanh layer generates a vector of new candidate values, Ct, that might be used to update the state of the system. Using these two pieces of information, we will produce an update to the state in the next step.

$$i_t = \sigma(W_i.[h_{t-1}, x_t] + b_i)$$

$$\tilde{C} = tanh(W_i.[h_{t-1}, x_t] + b_C)$$

It is now necessary to transition from the previous cell state, Ct1, to the new cell state Ct. What we need to do now is put into action based on the decisions made

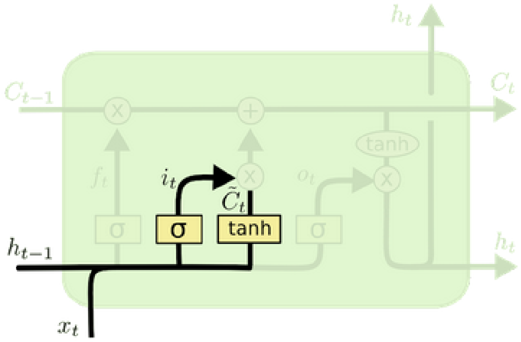


Figure 4.16: LSTM-2

in the previous phases. We double the old state, completely forgetting about the things we had previously agreed to ignore. After that, we append it Ct. This is a list of the new candidate values, scaled to reflect the amount by which we opted to change each state value.

In the instance of the language model, this is the point at which we would actually remove the information about the old subject's gender and replace it with the new information, as we had chosen in the previous phases.

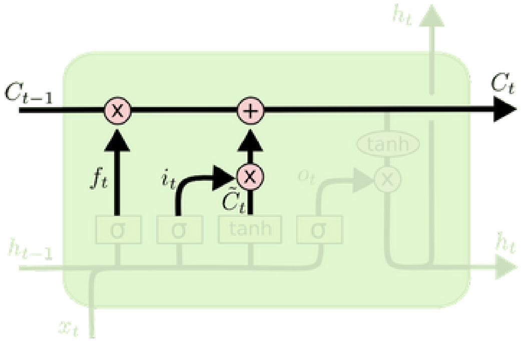


Figure 4.17: LSTM-3

$$C_t = f_t * C_{t-1} + i_t * \tilde{C_t}$$

At long last, we must pick what we are going to produce. This output will be based on the current status of our cell, but it will be a filtered version of the data. First, we run a sigmoid layer, which determines which parts of the cell state we will output and which parts we will not. Afterwards, we pass the cell state via tanh (which causes the values to be pushed between 0 and 1 and multiply it by the output of the sigmoid gate, which ensures that we only output the portions of the cell state that we choose to output.

In the instance of the language model, since it has just seen a subject, it may wish to output information that is related to a verb in case that is what is going to happen next.

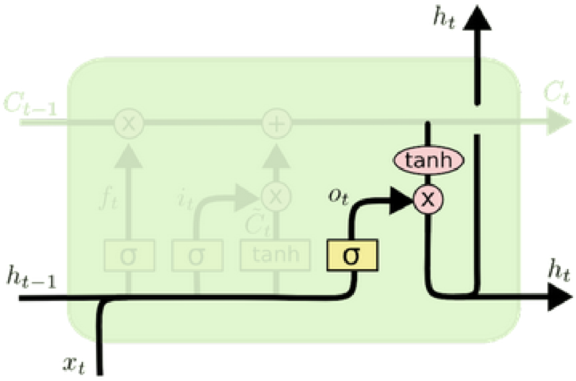


Figure 4.18: LSTM-4

$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o)$$
$$h_t = o_t * tanh(C_t)$$

4.8.1 BiLSTM

Bidirectional LSTM (BiLSTM) is a recurrent neural network used primarily on natural language processing. In contrast to typical LSTM, the input flows in both ways, and the algorithm is capable of exploiting information from both sides. It's also an effective tool for modeling the sequential interdependence between words and sentences on both sides of the sequence.

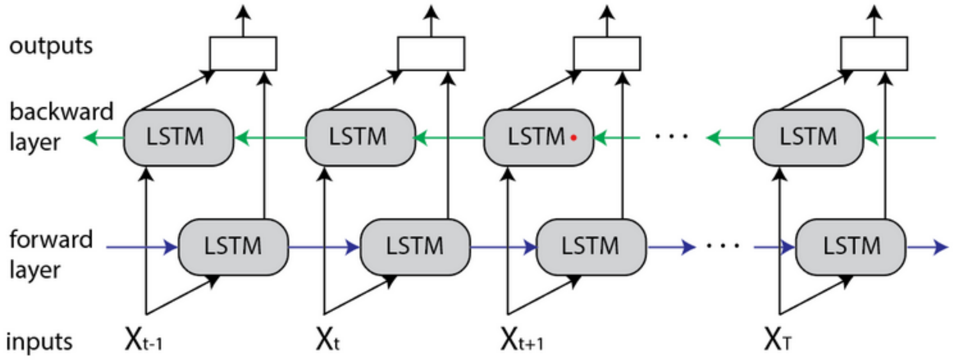


Figure 4.19: BiLSTM[16]

Bidirectional LSTMs are distinguished from standard LSTMs by the fact that their input may flow in both directions at the same time. We can make input flow in just one way using the ordinary LSTM that we discussed before. This may be done either backwards or forwards. The input may, however, be made to flow in both ways in bi-directional mode in order to keep both the upcoming as well as the past information in tact.

As an example, consider the statement "Girls go to........" for clarification. We are unable to fill in the blanks in this section. But with a future statement such as " Girls came out of the market ", we can readily anticipate what would happen in a previous blank area. Bi-LSTM enables the neural network to carry out this function.

Chapter 5

Result Analysis

5.1 Precision, Recall and F1 Score

Precision, recall, F1 score and confusion matrix are different approaches to describe the result from models. The result eventually compares the predicted value and actually labelled values but in different approaches. Before coming to these result analysis techniques, we must know the key factors on which these performance measure techniques are dependent on.

True Positives: True positive determines if the predicted value and actually labeled values are the same. This can be described as correctly predicted positive values. For example, if the predicted value is “Hate” also the labeled value is “Hate” it is considered as true positive (TP).

True Negative: True negative refers to values that were accurately expected to be negative. It might be termed true negative (TN) if the projected value is negative and the labeled value is likewise negative. For example, if the predicted value is “Not Hate” also the labeled value is “Not Hate” it is considered as true negative.

False Positives: Similarly, if the anticipated value is yes but the class is actually not positive and the result is a false positive (FP). For example, If actual class says “Not Hate” but the predicted value is “Hate” it is considered as false positive.

False Negatives: When labeled class is yes but predicted value is negative. For example, if the predicted value is “Not Hate” but the actual label is “Hate” we will consider that as false negative (FN).

Accuracy: This is the simplest way to calculate the performance. This is done by comparing the prediction and the real data. It shows how correctly the model predicts the classifications.

$$Accuracy = \frac{\text{TP+TN}}{\text{TP+FP+TN+FN}}$$

Recall: Recall shows us what amount of prediction is correct where all are positive. The formula is given below-

$$Recall = \frac{TP}{TP + FN}$$

Precision: Precision shows us what amount of prediction is really positive where all predictions are positive. The formula is given below-

$$Precision = \frac{TP}{TP + FP}$$

F1 Score: F1 Score gives us a classifiers performance metric using Precision and Recall.As a result,it takes false positives and false negatives all together.When the distribution of class is not same, F1 Score is more useful than accuracy. Accuracy works well, where positives and false negatives have identical costs.

$$\text{F1 Score} = \frac{2 * \text{(Recall*Precision)}}{\text{Recall+Precision}}$$

5.2 Result

We divided our hate comment data into two categories. To begin, we categorised them as "Hate" or "Non Hate" in Binary. Every remark was manually categorized. Multi-class classification is another type of classification. We divided our data into seven categories during this procedure. "Non-Hate", "Sexual", "Appearance", "Racial", "Slang", "Religious", and "Others". For example, to tokenize or vectorize the comments, we employed two distinct vectorization algorithms (data). For tokenization, we employed the TF-IDF and Bag of Words (BOW) algorithms. Then, to categorize them, we used eight different classification algorithms. The Logistic Algorithm, Multinomial Naive Bayes, Gaussian Naive Bayes, Decision Tree Classifier, Random Forest Tree, AdaBoost Classifier, BiLSTM and SVM have all been utilized. We calculated accuracy, precision, recall, and F1 score for binary and multiclass classifications using BOW and TF-IDF vectorization. In a nutshell, two datasets are categorised in four different ways. The results of the classifiers are listed below, organized by class type and tokenization procedure.

5.2.1 TF-IDF tokenized Multi-Class Dataset

Classifiers	Accuracy	Precision	Recall	F1 score
Multinomial NB	0.530	0.617	0.532	0.429
Gaussian NB	0.460	0.480	0.456	0.465
Decision Tree	0.460	0.330	0.464	0.311
AdaBoost	0.474	0.349	0.474	0.343
Random Forest	0.590	0.585	0.590	0.546
Logistic Regression	0.593	0.625	0.593	0.530
SVM	0.621	0.626	0.621	0.578
BiLSTM	0.438	0.191	0.438	0.266

Table 5.1: Multiclass TF-IDF

In the multiclass dataset, the data are tokenized by TF-IDF method and here, we take n-gram range of 3 for tokenization. In the TF-IDF method, we can have the highest accuracy in the Support Vector Machine (SVM) Algorithm which is 0.621. Followed by, Logistic Regression and Random Forest Algorithm which have an accuracy of 0.593 and 0.590 respectively. Also Multinomial Naive Bayes has an accuracy of 0.530. Again, AdaBoost Classifier Algorithm has an accuracy of 0.474. Decision Tree Classifier and Gaussian NB both have an accuracy of 0.46. The lowest accuracy we get from Bi-LSTM which is 0.438. So, in short we can say that, If we follow TF-IDF method to tokenize hate comments from multiclass dataset, SVM is the best algorithm for TF-IDF method as we get the highest accuracy from this algorithm. Following the SVM algorithm for multiclass we get the highest precision value which is 0.626. This means, we get 62.6% actual positive values among the predictable positive values here. SVM also has the highest recall value 0.621 which means we get 62.1% actual positive value rate among all the other values. Finally, SVM gives the highest F1 score 0.578 which gives us the actual positive values that have hate comments in it and thus it is a good model. We get precision values from Logistic regression, Multinomial NB and Random Forest are 0.625, 0.617 and 0.585 respectively which is above 50% and that means by these algorithms we get better actual positive values among the predictable positive values. And then we get precision values less than 50% in GaussianNB, Adaboost and Decision tree classifiers which are 0.480, 0.349 and 0.330 respectively. And the lowest precision score we get from Bi-LSTM which is 0.191, which means it gets the lowest actual positive values among the predictable positive values. We get recall values from Logistic regression, Random forest and Multinomial NB are 0.593, 0.590 and 0.532 respectively. These classifiers get a better rate more than 50% which means the rate of actual positive value among all the other values. Again, there are AdaBoost and Decision tree classifiers and GaussianNB which get recall values of 0.474, 0.464 and 0.456 respectively. And the lowest recall value we get from Bi-LSTM which is 0.438. This means Bi-LSTM gives the lowest actual positive value rate among all the other values in the dataset. We get an F1 score from the Random Forest and Logistic Regression classifier which is 0.546 and 0.530 respectively, that is more than 50%. The f1 score represents the average value of precision and recall value. That is why we can say the Random forest and Logistic Regression model are good models. After that, we get F1 scores from GaussianNB and Multinomial NB which are 0.465 and 0.429 respectively. And there we get f1 scores from AdaBoost and Decision tree classifiers which are 0.343 and 0.311 respectively and thus these are not good models. And the lowest F1 scores we get from Bi-LSTM which is 0.266, this means this model is not a good model for our dataset. Finally we can say that SVM is the best classifier for TF-IDF method in multiclass dataset and the worst is Bi-LSTM for TF-IDF in multiclass dataset.

5.2.2 Bag of Word tokenized Multi-Class Dataset

Classifiers	Accuracy	Precision	Recall	F1 score
Multinomial NB	0.600	0.579	0.602	0.574
Gaussian NB	0.460	0.499	0.460	0.475
Decision Tree	0.460	0.329	0.463	0.310
AdaBoost	0.475	0.360	0.475	0.346
Random Forest	0.556	0.539	0.556	0.529
Logistic Regression	0.615	0.601	0.615	0.587
SVM	0.566	0.556	0.566	0.555
BiLSTM	0.442	0.283	0.442	0.276

Table 5.2: Multiclass BOW

The data in a multiclass dataset is tokenized using the Bag of Words approach. For tokenization, we used an n-gram range of 2. We can observe that the Logistic Regression model has a high accuracy of 0.615. For a multiclass dataset, that's a great outcome. Multinomial Naive Bayes came in second with a score of 0.600. SVM and Random Forest, on the other hand, had an accuracy of 0.566 and 0.556, respectively. Gaussian Naïve Bayes, on the other hand, has the accuracy of 0.460. The accuracy of the Decision tree classifier and the AdaBoost Classifier are below 50%, at 0.460. So, if the hate comments are tokenized via Bag of Words, we may conclude that Logistic Regression is the best classifier for a multiclass dataset

With the help of Bag of Words here we are seeing the result of binary classification of our hate speech detection part. In here, our ngram of Bag of Word tokenization algorithm is 2, which means it will take two words together in a data at a time. We used 8 types of classification algorithms. Now we will discuss every model's performance with help of Precision, Recall and F1 Score.
Firstly, Logistic Regression has a higher Recall value, which means that this classifier predicts more positive values correctly. So, this classifier can predict more perfectly that this data is hate speech or not. Along with it Multinomial NB and SVM give us close results. Among these, Multinomial NB has the Recall value which is 0.602. Logistic Regression(0.615) is highest and SVM(0.566). The other four classifiers have low Recall value. Recall values from Random Forest, Gaussian Naive Bayes,Decision tree Classifier, AdaBoost Classifier and Bi-LSTM are 0.566, 0.460, 0.643, 0.475 and 0.442 respectively.
On the other hand, Precision values from Multinomial Naive Bayes,Logistic Regression, SVM, Random Forest, Gaussian Naive Bayes,Decision tree Classifier, AdaBoost Classifier and Bi-LSTM are 0.579, 0.601,0.556, 0.539, 0.499, 0.329, 0.360

and 0.283 respectively. Higher precision value means how these three classification algorithms are predicting correctly from positive values.
In case of F1 Score value, Logistic Regression has highest value which is 0.587, Followed by, Multinomial NB(0.574), SVM(0.555),Random Forest(0.529), Gaussian Naive Bayes(0.475), Decision Tree Classifier(0.310) and AdaBoost Classifier(0.346). Among these classifiers, Logistic Regression is the best because it has the highest F1 Score. After that, Random Forest and Gaussian Naive Bayes are in the 0.52-0.47 range. Lastly ,among all the classifiers Decision Tree, AdaBoost Classifier and Bi-LSTM performance is lowest which are 0.310,0.346 and 0.276 respectively.

5.2.3 TF-IDF tokenized Binary Dataset

Classifiers	Accuracy	Precision	Recall	F1 score
Multinomial NB	0.740	0.743	0.740	0.734
Gaussian NB	0.680	0.687	0.681	0.682
Decision Tree	0.600	0.630	0.599	0.526
AdaBoost	0.574	0.598	0.574	0.468
Random Forest	0.692	0.691	0.692	0.687
Logistic Regression	0.740	0.740	0.740	0.737
SVM	0.743	0.742	0.743	0.741
BiLSTM	0.576	0.332	0.576	0.422

Table 5.3: Binary TF-IDF

In the table we can see the accuracy, precision, recall and F1 score for each methodologies. By running the methodologies we observed that SVM provided the highest accuracy of 0.743 following that Logistic Regression 0.74, Multinomial NB 0.74, Random Forest 0.692, Gaussian NB 0.68, Decision Tree Classifier 0.6, AdaBoost Classifier 0.574 and last Bi-LSTM is 0.576.
Multinomial NB provided the highest precision of 0.743. SVM gives almost the same score as Multinomial NB ,which is 0.742. Logistic Regression, Random Forest Classifier, Gaussian NB, Decision Tree Classifier provides 0.740, 0.691, 0.687 and 0.63 respectively. From the table we can see that SVM has accuracy of 0.743 but its precision rate is slightly lower than Multinomial NB. Logistic Regression, Random Forest Classifier, Gaussian NB, Decision Tree Classifier provides 0.740, 0.691, 0.687 and 0.63. These low scores suggest a substantial number of courses that were incorrectly identified. AdaBoost Classifier provided precision of 0.574 and BI-LSTM provided 0.332. As for the recall score SVM, Logistic Regression and Multinomial NB provides 0.743, 740 and 0.740 respectively. Next to them Random Forest Classifier provides a 0.692 score. This suggests that the most relevant results were produced by SVM, Logistic Regression and Multinomial NB. Gaussian NB, Decision Tree Classifier provides 0.681, 0.599 respectively and Like accuracy and precision score, Bi-LSTM provided the lowest score on recall, which is 0.576. Bi-LSTM has also provided the lowest F1 score of 0.422. On the other hand SVM, Logistic Regression, Multinomial NB, Random Forest Classifier, Gaussian NB, Decision Tree Classifier provides 0.741, 0.737, 0.734; 0.687; 0.682 and 0.526 respectively. AdaBoost provides F1 score of 0.468. As for the summary we can say that, for this Binary TF-IDF the

best classifiers are SVM, Logistic Regression and Multinomial NB. Each of these classifiers provided a very good score in accuracy, precision, recall and F1 score.

5.2.4 Bag of Word tokenized Binary Dataset

Classifiers	Accuracy	Precision	Recall	F1 score
Multinomial NB	0.760	0.760	0.761	0.760
Gaussian NB	0.690	0.698	0.688	0.689
Decision Tree	0.600	0.625	0.597	0.525
AdaBoost	0.586	0.641	0.586	0.484
Random Forest	0.683	0.681	0.683	0.682
Logistic Regression	0.734	0.733	0.734	0.732
SVM	0.718	0.717	0.718	0.717
BiLSTM	0.576	0.332	0.576	0.423

Table 5.4: Binary BOW

With the help of Bag of Words here we are seeing the result of binary classification of our hate speech detection part. In here, our ngram of Bag of Word tokenization algorithm is 2, which means it will take two words together in a data at a time. We used 8 types of classification algorithms. Now we will discuss every model's performance with the help of Accuracy, Precision, Recall and F1 Score.

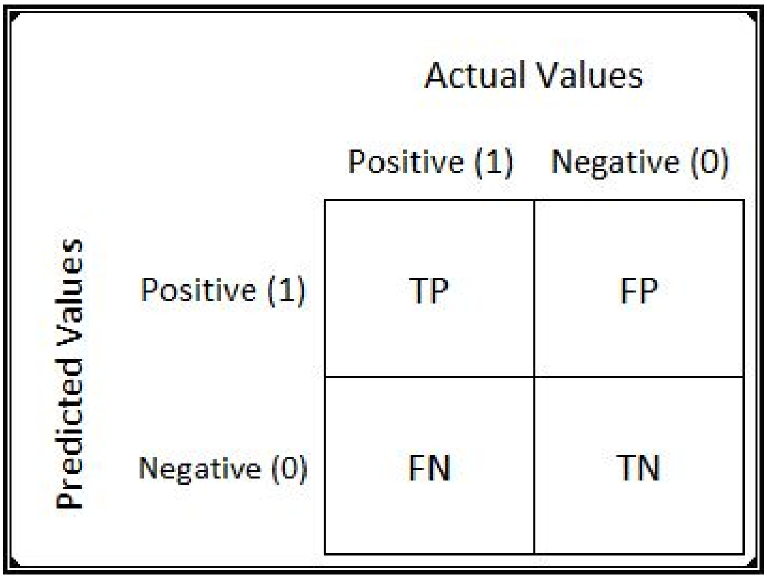
Firstly, Multinomial Naive Bayes has the highest accuracy of 0.760. Again, Logistic Regression and SVM give accuracy of 0.734 and 0718 respectively. And the lowest accuracy comes from Adaboost and Bi-LSTM which are 0.586 and 0.576 respectively. On the other hand, Precision values from Multinomial Naive Bayes, Logistic Regression, SVM, Gaussian Naive Bayes, Random Forest, AdaBoost Classifier, Decision tree Classifier and Bi-LSTM are 0.760, 0.733, 0.717, 0.698, 0.681, 0.641, 0.625 and 0.332 respectively. Multinomial Naive Bayes, Logistic Regression and SVM has Higher precision value which refers to how these three classification algorithms are predicting more correctly from positive values.

Multinomial Naive Bayes has a higher Recall value, which is 0.761, that means that this classifier predicts more positive values correctly. So, this classifier can predict more perfectly that this data is hate speech or not. Along with it, Logistic Regression and SVM give recall values of 0.734 and 0.718 respectively. Recall values from Gaussian Naive Bayes and Random Forest 0.688 and 0.683 respectively. Moreover, we get recall values from Decision Tree and Adaboost are 0.597 and 0.586 respectively. And the lowest recall we get is from Bi-LSTM which is 0.576.

In the case of F1 Score value, Multinomial Naive Bayes has the highest value which is 0.760. Again, we get F1 scores from Logistic Regression(0.732), SVM(0.717), Gaussian Naive Bayes(0.689), Random Forest(0.682) and Decision Tree Classifier(0.525). Here, we can see that the first three classifiers give F1 scores of around 0.71-0.76. These three are performing better than others. Among these three, Multinomial Naive Bayes is the best because it has the highest F1 Score. Lastly, among all the classifiers AdaBoost Classifier and Bi-LSTM performance is lowest which are 0.484 and 0.423 respectively. In conclusion, in the Binary Bag of Words Multinomial Naive Bayes performed best and after that Logistic Regression and SVM.

5.3 Confusion Matrix

In a classification problem, Confusion matrix shows us how predicted values are perfectly predicted. To be specific, it shows us how our used model is performing. We can use this in both binary and multiclass problems. Using a n*n matrix, it shows us how much error we have and what type of error. Here, n depends how many classes we have in our problem. This is also very effective to calculate Recall, Precision and Accuracy.



		Actual Values	
		Positive (1)	Negative (0)
Predicted Values	Positive (1)	TP	FP
	Negative (0)	FN	TN

Figure 5.1: Confusion Matrix

The confusion matrix of Multiclass Dataset and Binary Dataset using TF-IDF and Bag of Words performed on different classifiers such as MultinomialNB, Gaussian NB,Decision Tree,AdaBoost,Random Forest,Logistic Regression,SVM are shown in the next page.

5.3.1 Confusion Matrix Based Result

From the confusion matrix we can determine how the used algorithms detected hate, non-hate, sexual, slang, racial, religious, appearance and other type hate comment. It compares between prediction and true labels. Basically, it shows the comparisons between true positives, false positives, true negatives and false negatives or which class is being misclassified. In our case we have two types of datasets. One is binary labeled (Hate, non-hate) and multi-labeled (non-hate, sexual, slang, religious, appearance and others). Also, as we have tokenized the comments using both bag of words and TF-IDF. So, there are 14 confusion matrixes for each type of datasets.

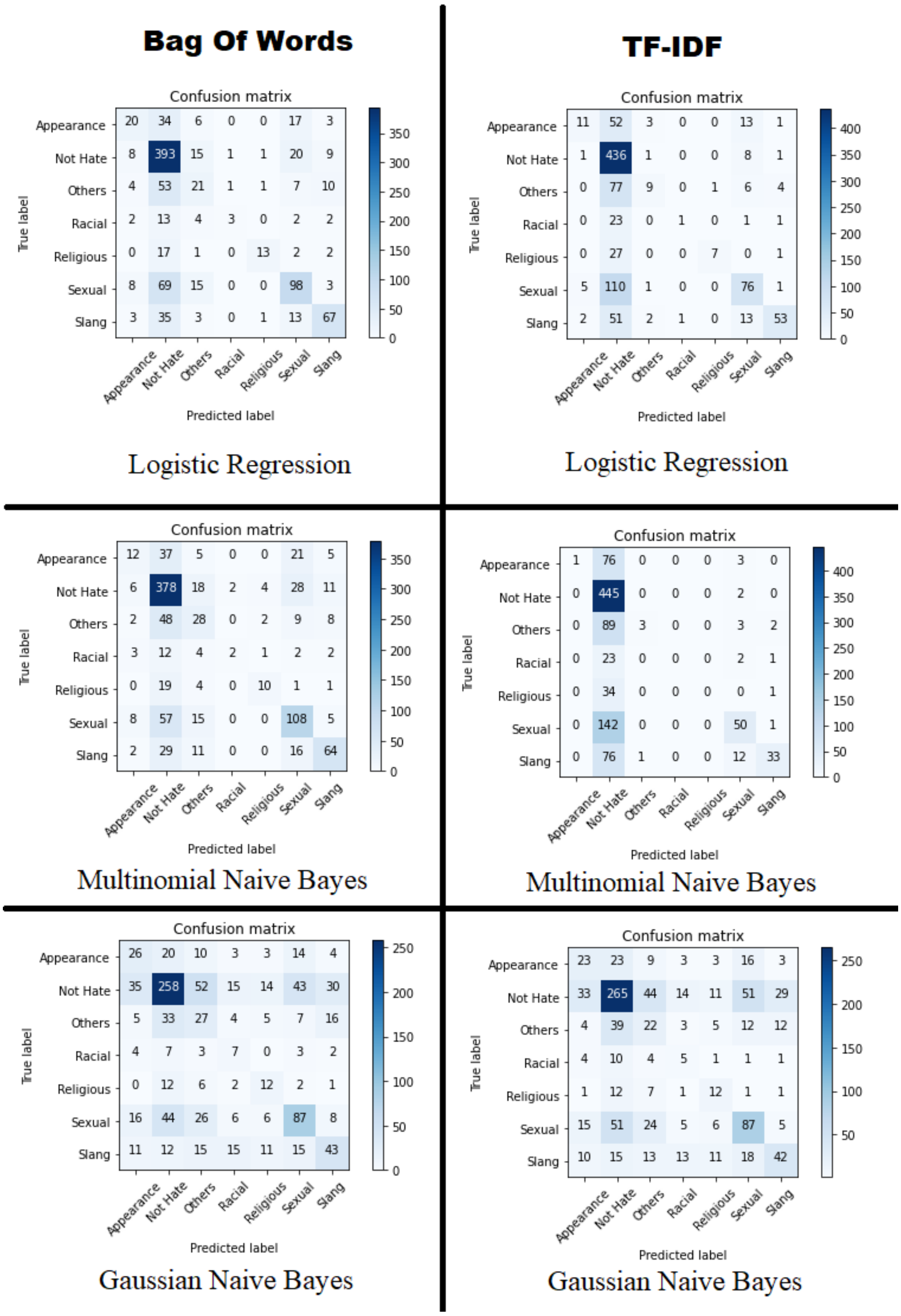
5.3.2 Binary Dataset

Out of all algorithms Multinomial Naive Bayes and SVM had highest accuracy and F1 score. From the confusion matrix we can see Multinomial detected 312 non-hate comments correctly among 443 training non-hate comments. But the number of correctly predicted hate comments was satisfying. Among 558 hate comments it predicted 450 comments correctly. This was the result of the Bag of words tokenization process. In the case of TF-IDF the result is almost the same. Only Hate predictions got better. Correctly predicted hate comments are 477 out of 558. But the number of wrongly predicted hate comments decreased slightly. Which is 81 out of 558. For the Bag of Word tokenized SVM confusion matrix we can see 293 correctly predicted non-hate comments out of 443 and for TF-IDF tokenized data 298 correctly predicted non-hate comments. On the other hand correctly predicted hate comments for Bag of Word it is 426 and TF-IDF 446 out of 558. From all the confusion matrixes we can see comparatively the number of correctly predicted hate comments is higher than correctly predicted non-hate comments. However, the AdaBoost classifier showed outstanding results in correctly predicted hate comments but it predicted 396 (for Bag of Word) non-hate comments as hate. For this reason, the F1 score of AdaBoost was not that high. On the other hand, the Gaussian Naïve Bayes classifier showed better results in predicting non-hate comments. About 325 non-hate comments were correctly predicted for bag of words tokenized data and 310 non-hate comments were correctly predicted for TF-IDF tokenized Gaussian Naïve Bayes classified data.

5.3.3 Multi-Class Dataset

As our dataset contains comparatively higher non-hate comments among all the hate comments, all the confusion matrix has higher correctly predicted non-hate comments. Also, many other hate comments are also predicted as non-hate comments. For instance, in AdaBoost and Decision Tree Classifier a large number of racial, religious, sexual, slang, others and appearance hate comments are classified as non-hate comments. Among all the classifiers Decision Tree classifier has predicted all the non-hate comments correctly. But overall, we saw SVM and Logistic Regression had the highest F1 score. From confusion matrix we can see that these two classifier predicted non-hate comments as well as slang and sexual based hate comments correctly. SVM predicted 88 sexual hate comments correctly among 193 comments and 64 correctly predicted slangs among 122 slang comments.In Bag of word, Logistic Regression predicted correctly 98 sexual hate comment out of 193. As well as it predicted correctly 67 slang hate comments out of 122.

5.3.4 Confusion Matrix of Multi-Class Dataset



Bag Of Words — Confusion matrix

True label \ Predicted label	Appearance	Not Hate	Others	Racial	Religious	Sexual	Slang
Appearance	20	34	6	0	0	17	3
Not Hate	8	393	15	1	1	20	9
Others	4	53	21	1	1	7	10
Racial	2	13	4	3	0	2	2
Religious	0	17	1	0	13	2	2
Sexual	8	69	15	0	0	98	3
Slang	3	35	3	0	1	13	67

Logistic Regression

TF-IDF — Confusion matrix

True label \ Predicted label	Appearance	Not Hate	Others	Racial	Religious	Sexual	Slang
Appearance	11	52	3	0	0	13	1
Not Hate	1	436	1	0	0	8	1
Others	0	77	9	0	1	6	4
Racial	0	23	0	1	0	1	1
Religious	0	27	0	0	7	0	1
Sexual	5	110	1	0	0	76	1
Slang	2	51	2	1	0	13	53

Logistic Regression

Confusion matrix

True label \ Predicted label	Appearance	Not Hate	Others	Racial	Religious	Sexual	Slang
Appearance	12	37	5	0	0	21	5
Not Hate	6	378	18	2	4	28	11
Others	2	48	28	0	2	9	8
Racial	3	12	4	2	1	2	2
Religious	0	19	4	0	10	1	1
Sexual	8	57	15	0	0	108	5
Slang	2	29	11	0	0	16	64

Multinomial Naive Bayes

Confusion matrix

True label \ Predicted label	Appearance	Not Hate	Others	Racial	Religious	Sexual	Slang
Appearance	1	76	0	0	0	3	0
Not Hate	0	445	0	0	0	2	0
Others	0	89	3	0	0	3	2
Racial	0	23	0	0	0	2	1
Religious	0	34	0	0	0	0	1
Sexual	0	142	0	0	0	50	1
Slang	0	76	1	0	0	12	33

Multinomial Naive Bayes

Confusion matrix

True label \ Predicted label	Appearance	Not Hate	Others	Racial	Religious	Sexual	Slang
Appearance	26	20	10	3	3	14	4
Not Hate	35	258	52	15	14	43	30
Others	5	33	27	4	5	7	16
Racial	4	7	3	7	0	3	2
Religious	0	12	6	2	12	2	1
Sexual	16	44	26	6	6	87	8
Slang	11	12	15	15	11	15	43

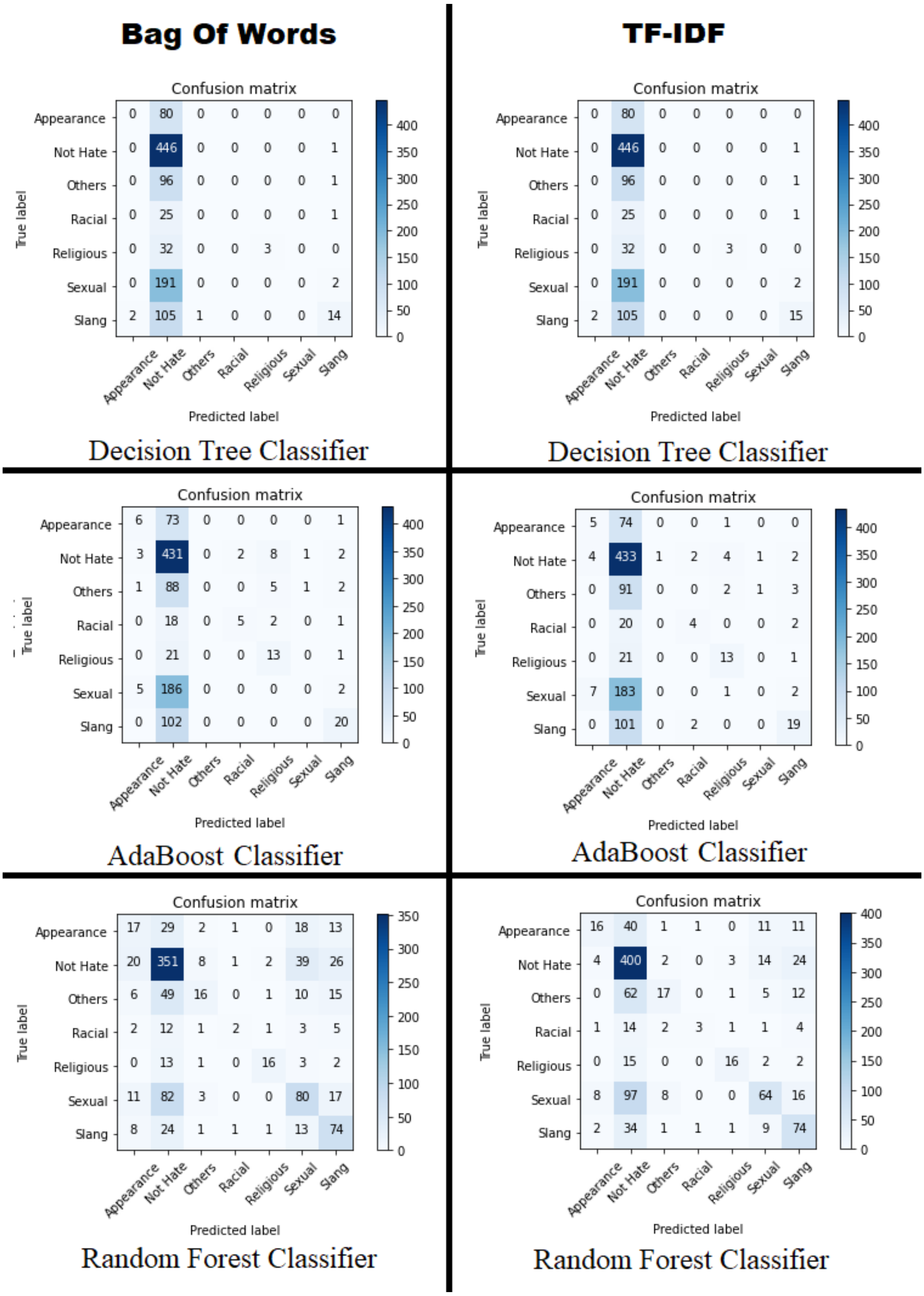
Gaussian Naive Bayes

Confusion matrix

True label \ Predicted label	Appearance	Not Hate	Others	Racial	Religious	Sexual	Slang
Appearance	23	23	9	3	3	16	3
Not Hate	33	265	44	14	11	51	29
Others	4	39	22	3	5	12	12
Racial	4	10	4	5	1	1	1
Religious	1	12	7	1	12	1	1
Sexual	15	51	24	5	6	87	5
Slang	10	15	13	13	11	18	42

Gaussian Naive Bayes

Figure 5.2: Confusion Matrix of Multi-Class Dataset



Bag Of Words — Decision Tree Classifier

True label \ Predicted label	Appearance	Not Hate	Others	Racial	Religious	Sexual	Slang
Appearance	0	80	0	0	0	0	0
Not Hate	0	446	0	0	0	0	1
Others	0	96	0	0	0	0	1
Racial	0	25	0	0	0	0	1
Religious	0	32	0	0	3	0	0
Sexual	0	191	0	0	0	0	2
Slang	2	105	1	0	0	0	14

TF-IDF — Decision Tree Classifier

True label \ Predicted label	Appearance	Not Hate	Others	Racial	Religious	Sexual	Slang
Appearance	0	80	0	0	0	0	0
Not Hate	0	446	0	0	0	0	1
Others	0	96	0	0	0	0	1
Racial	0	25	0	0	0	0	1
Religious	0	32	0	0	3	0	0
Sexual	0	191	0	0	0	0	2
Slang	2	105	0	0	0	0	15

Bag Of Words — AdaBoost Classifier

True label \ Predicted label	Appearance	Not Hate	Others	Racial	Religious	Sexual	Slang
Appearance	6	73	0	0	0	0	1
Not Hate	3	431	0	2	8	1	2
Others	1	88	0	0	5	1	2
Racial	0	18	0	5	2	0	1
Religious	0	21	0	0	13	0	1
Sexual	5	186	0	0	0	0	2
Slang	0	102	0	0	0	0	20

TF-IDF — AdaBoost Classifier

True label \ Predicted label	Appearance	Not Hate	Others	Racial	Religious	Sexual	Slang
Appearance	5	74	0	0	1	0	0
Not Hate	4	433	1	2	4	1	2
Others	0	91	0	0	2	1	3
Racial	0	20	0	4	0	0	2
Religious	0	21	0	0	13	0	1
Sexual	7	183	0	0	1	0	2
Slang	0	101	0	2	0	0	19

Bag Of Words — Random Forest Classifier

True label \ Predicted label	Appearance	Not Hate	Others	Racial	Religious	Sexual	Slang
Appearance	17	29	2	1	0	18	13
Not Hate	20	351	8	1	2	39	26
Others	6	49	16	0	1	10	15
Racial	2	12	1	2	1	3	5
Religious	0	13	1	0	16	3	2
Sexual	11	82	3	0	0	80	17
Slang	8	24	1	1	1	13	74

TF-IDF — Random Forest Classifier

True label \ Predicted label	Appearance	Not Hate	Others	Racial	Religious	Sexual	Slang
Appearance	16	40	1	1	0	11	11
Not Hate	4	400	2	0	3	14	24
Others	0	62	17	0	1	5	12
Racial	1	14	2	3	1	1	4
Religious	0	15	0	0	16	2	2
Sexual	8	97	8	0	0	64	16
Slang	2	34	1	1	1	9	74

Figure 5.3: Confusion Matrix of Multi-Class Dataset

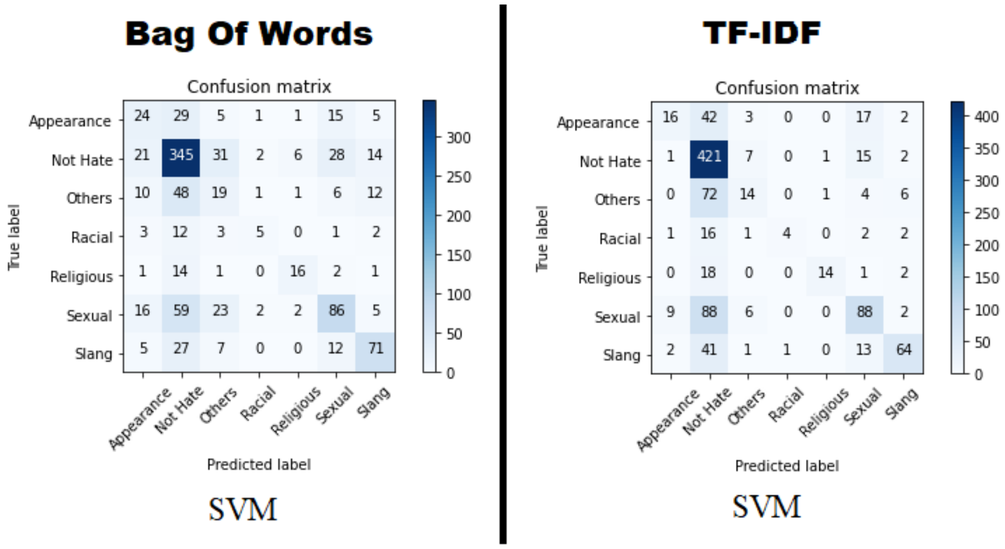


Figure 5.4: Confusion Matrix of Multi-Class Dataset

5.3.5 Confusion Matrix of Binary Dataset

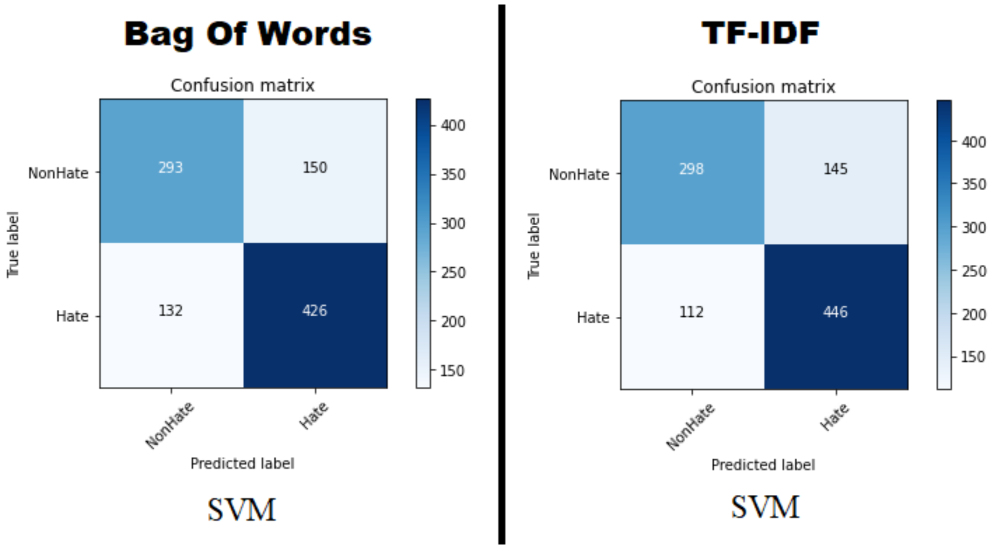


Figure 5.5: Confusion Matrix of Binary Dataset

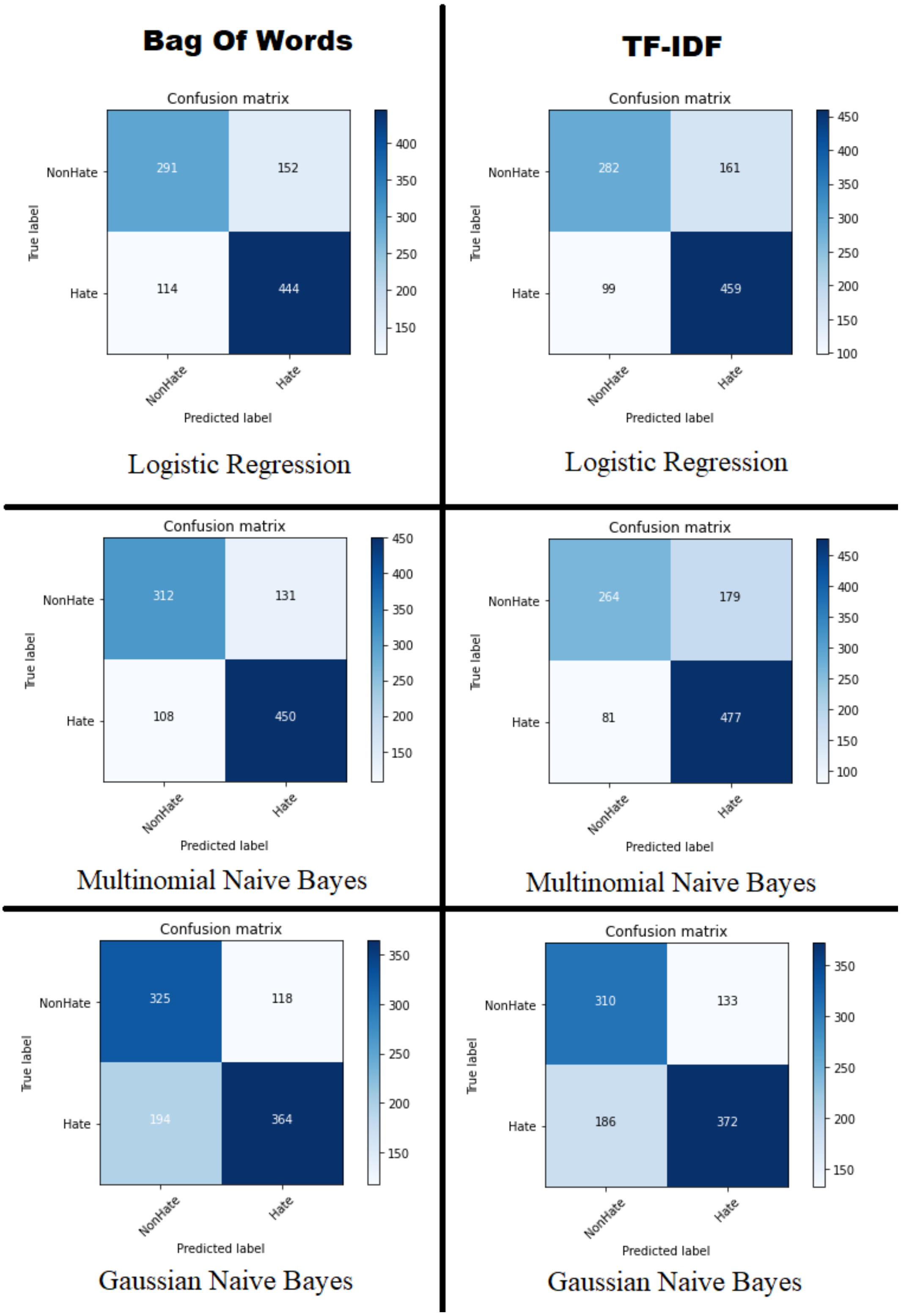


Figure 5.6: Confusion Matrix of Binary Dataset

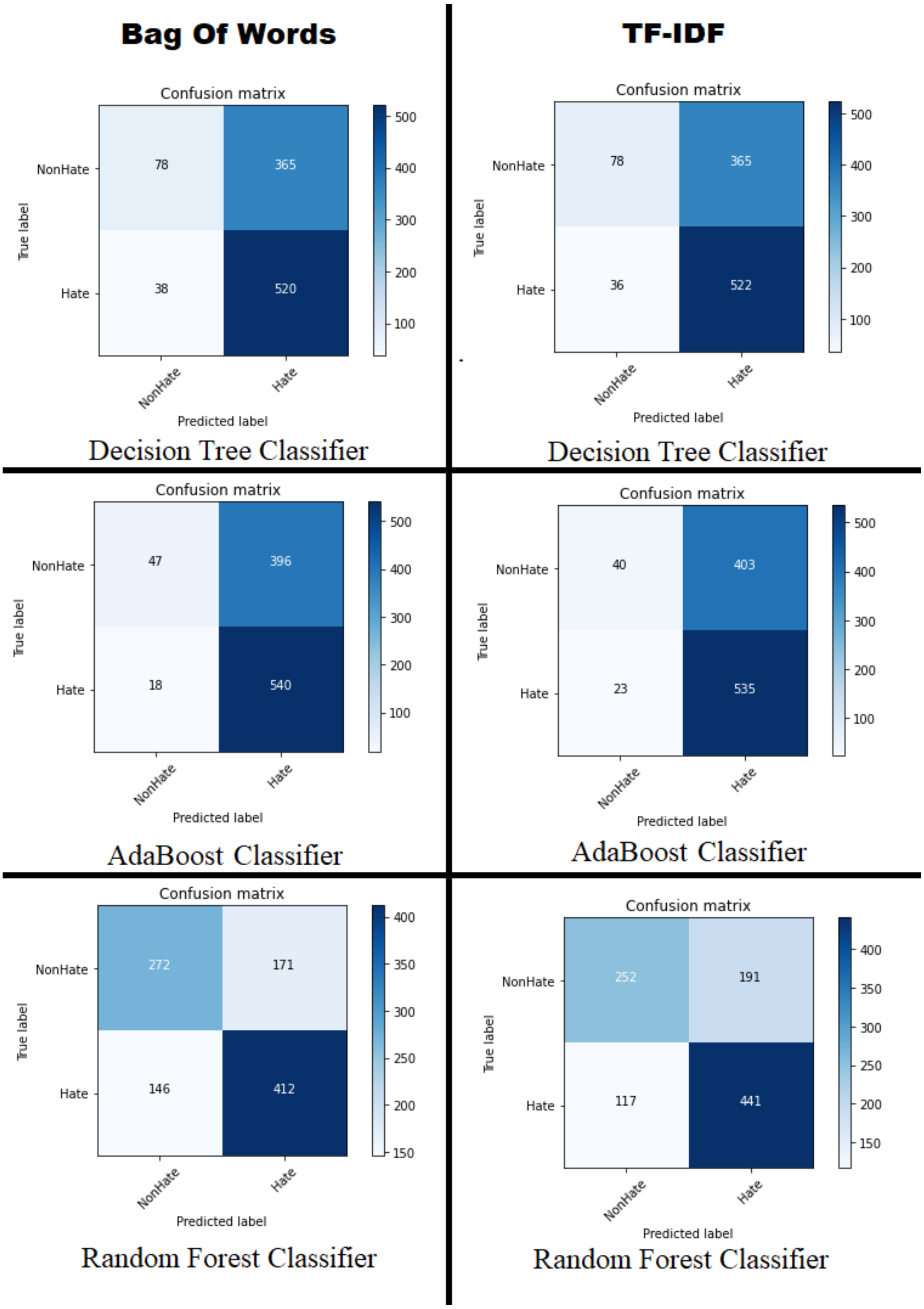


Figure 5.7: Confusion Matrix of Binary Dataset

5.4 Result Comparison

A similar type of research on “Abusive content detection in transliterated Bengali-English social media corpus” was conducted by an author named Sazzed in 2021[17]. In his paper he stated that he also made a transliterated bengali hate speech binary dataset which contains equal number of hate and non hate comments and he got the following results. He used SVM, Logistic Regression, Random Forest and BiLSTM. The score results were as follows. SVM Recall score was (0.79), precision(0.86) and f1(0.827) where else we got Recall (0.743),precision (0.742) and F1(0.741). For Logistic Regression the score was Recall(0.77),precision(0.876) and f1(0.823). Wherelese, we got recall(0.74),precision(0.74) and f1(0.737). However comparison included the best result for both of the works is given below.

Models Used	Dataset Used	Data Size	Best Recall	Best F1	Best Precison
4 (Sazzed.S)	Binary	3000	0.79	0.827	0.876
8 (Our)	Binary&Multiclass	5000	0.761	0.760	0.760

Table 5.5: Comparison Table

5.5 Proposed Model Analysis

5.5.1 SVM

SVM is a supervised machine learning technique that may be used to solve issues like classification and regression. It transforms the data using a method known as the kernel trick, and then calculates an ideal boundary between the potential outputs based on these alterations. When there is a clear margin of distinction between classes, SVM performs effectively. In high-dimensional spaces, SVM is more effective. SVMs minimize the requirement for feature selection because to their ability to generalize effectively in high-dimensional feature spaces, making text classification much simpler to apply. SVMs also have the benefit of being more resilient than traditional approaches. When the number of dimensions is more than the number of samples, SVM is successful. For the reasons stated above, SVM outperformed several other algorithms in terms of accuracy.

5.5.2 Multinomial NB

The multinomial nave Bayes algorithm is often used to classify texts based on statistical study of their contents. It offers an alternative to ”heavy” AI-based semantic analysis and greatly simplifies the categorization of textual data.

Multinomial Naive Bayes is simple to use since all we have to do is compute probability. This approach works with both continuous and discrete data. It’s very scalable and can handle enormous datasets with ease. It doesn’t need as much data for training. It is capable of dealing with both continuous and discrete data. It can handle a large number of predictors and data points. For multi-class prediction issues, Naive Bayes is a good choice. If the premise of feature independence remains true, it has the potential to outperform other models. Categorical input variables are more suited to Naive Bayes than numerical input variables. It also establishes a link between all of the characteristics. As a result, Multinomial Naive Bayes had the greatest accuracy for binary dataset, which was 76%

5.5.3 Logisic Regression

Logistic regression is simpler to construct and analyze, and it trains extremely well with a relatively little amount of data. It works fairly quickly to categorize records with unknown origins. When the dataset can be separated linearly, it works rather well. It is able to interpret the model coefficients as indications of the significance of the features. Logistic regression is a great tool for this since it outputs a probability that is between 0 and 1, and it does so by using a sigmoid function. Because of this, our project ended up having one of the finest classifiers possible.

5.6 Overall Discussion

For both Multi-Class and Binary dataset TF-IDF worked better than Bag of Words for SVM. For multi-class accuracy of TF-IDF was 0.621 on the other hand for Bag of Words it was 0.566. So TF-IDF is much preferable than BOW in the case of SVM. Also for TF-IDF tokenized binary dataset SVM gave an accuracy of 0.742, whereas for BOW tokenized data showed 0.718. Another dominating algorithm in our case is Multinomial Naive Bayes. In the case of Multinomial Naive Bayes we can see Bag of Word is much more efficient than TF-IDF. For example: the accuracy of Bag of words is 0.60 on the other hand the accuracy of TF-IDF is 0.53 which is at least 7% better than TF-IDF. In the binary Bag of Word dataset for Multinomial Naive Bayes algorithm showed the highest accuracy which is 0.76 and for TF-IDF tokenized data it was 0.74. This is the highest accuracy we measured from our conducted research. For another dominating algorithm of our research, Logistic Regression the accuracy in Binary class both Bag of Words and TF-IDF gave almost same accuracy. For Bag of Words it was 0.734 and for TF-IDF it was0.74. But the Multi-class Bag of Words gave better accuracy than TF-IDF. The score of Bag of Word is 0.615. But TF-IDF has less accuracy by less than 1%. So we can see for binary dataset there is not that much drastic change between TF-IDF and Bag of Words. We got almost the same accuracy for both of the tokenizer algorithms. But the change is noticeable for multi-class datasets. Among the eight algorithms for three algorithms, Bag of Word worked better than TF-IDF. And only for two algorithms TF-IDF was much more efficient than Bag of Word. For the rest of the algorithm we got the same accuracy. The highest accuracy recorded from the multi-class dataset was tokenized by Bag of Word (Logistic Regression, accuracy: 0.615). Also TF-IDF tokenized data was able to almost touch the peak accuracy (SVM, accuracy:0.621). However, for Bi-LSTM we didn't see any change in accuracy for both TF-IDF and BOW. From the infos above we can come to a decision that Bag of Words works better than TF-IDF for multi-class datasets. On the contrary TF-IDF is more efficient than Bag of Words in case of binary datasets.For Bag of Word tokenized multi-class dataset we can see Logistic Regression has a higher precision, recall and F1 score than other algorithms. Along with that, in Multi-class TF-IDF SVM performed better than any other algorithm in precision, recall and F1 score.For Binary Bag of Words, by comparing all the algorithms, Multinomial Naive Bayes has a good score in all of the three performance measurements(precision, recall and F1 score).

Logistic regression outperformed other methods on the TF-IDF tokenized binary dataset in terms of precision, recall, and F1 score.So, tokenizer algorithms have an impact on precision, recall and F1 score as well as accuracy. Finally, we may infer that the best algorithms for multi-class datasets are Bag of Word tokenizer and Logistic Regression, whereas the best algorithm for binary datasets is Multinomial Naive Bayes and both tokenizers are preferable. Bi-LSTM works more efficiently in case of long sentences for sentence classification. In our case, our dataset consists of sentences with word length of 1-5 about 32.86% as per the comments we collected from facebook. Also length of sentences from 6-10 was about 42.96%. As about 75.82% of data is short length sentences we got such accuracy from Bi-LSTM which was not expected from recurrent neural networks.

Classifiers	TF-IDF (MC)	Bag of Words(MC)	TF-IDF(B)	Bag of Words(B)
Multinomial NB	0.530	0.600	0.740	0.760
Gaussian NB	0.460	0.460	0.680	0.690
Decision Tree	0.460	0.460	0.600	0.600
AdaBoost	0.474	0.475	0.574	0.586
Random Forest	0.590	0.556	0.692	0.683
Logistic Regression	0.593	0.615	0.740	0.734
SVM	0.621	0.566	0.743	0.718
BiLSTM	0.438	0.442	0.576	0.576

Table 5.6: Summary of Accuracy

Here MC stands for Multi-Class and B stands for Binary. From table we can see highest accuracy was observed for Multinomial NB for Binary dataset and the lowest accuracy was recorded in case of BiLSTM for multiclass dataset.

Chapter 6

Conclusion

On a variety of social media platforms, there are a few different mechanisms that can be used to identify instances of hate speech.In most cases, they are able to comprehend fundamental languages including the English language.It's possible that the algorithms that detect hate speech won't pay any attention to people who use mixed language.However, throughout the entirety of our experiment, we focused on identifying hate speech that contained elements of both Bangla and English.Not only were we able to determine whether or not the comment constitutes hate speech, but we were also capable of classifying the type of hate speech that it was, such as (racial, appearance, religious, slang, sexual and other).We had sufficient data to carry out such a research, but due to the length of the comments that were gathered, we had some difficulties while running it on Bi-LSTM. However, the weaknesses of this research can be enhanced in future work.

Bibliography

[1] Graham, C. (2019, March 14). Christchurch Shooting Live Updates: 49 Are Dead After 2 Mosques Hit. The New York Times. https://www.nytimes.com/2019/03/14/world/asia/new-zealand-shooting-updates-christchurch.html

[2] Hate Speech ABA Legal Fact Check—American Bar Association; Available from: https://abalegalfactcheck.com/articles/hate-speech.html.

[3] Hatzipanagos, R. (2018, November 30). How online hate turns into real-life violence. https://www.washingtonpost.com/nation/2018/11/30/how-online-hate-speech-is-fueling-real-life-violence/

[4] People should be able to make statements that are offensive to minority groups publicly. (2019, June 7). [Graph]. https://www.cfr.org/backgrounder/hate-speech-social-media-global-comparisons

[5] Clint, S. (2021, July 31). Increasing hate speech cases on Twitter [Photograph]. https://www.safehome.org/resources/hate-on-social-media/

[6] The EU Code of conduct on countering illegal hate speech online. (n.d.). The EU Code of Conduct. Retrieved September 20, 2021,from https://ec.europa.eu/info/policies/ justice-and-fundamental-rights/combatting-discrimination/racism-and-xenophobia/eu-code-conduct-countering-illegal-hate-speech-online_en

[7] Proactive Detection Rate for Hate Speech. (2020, November 19). [Histogram]. Proactive Detection Rate for Hate Speech. https://about.fb.com/news/2020/11/measuring-progress-combating-hate-speech/

[8] .Abro, S., Shaikh, S., Ali, Z., Khan, S., Mujtaba, G., Khand, Z. H. (2020). Automatic Hate Speech Detection using Machine Learning: A Comparative Study. International Journal of Advanced Computer Science and Applications, 11(8).

[9] Gomez, Raul Gibert, Jaume Gomez, Lluis Karatzas, Dimosthenis. (2019). Exploring Hate Speech Detection in Multimodal Publications. [1910.03814] Exploring Hate Speech Detection in Multimodal Publications (arxiv.org

[10] Gitari, N. D., Zuping, Z., Long, J., Damien, H. (2015). A Lexicon-based Approach for Hate Speech Detection. International Journal of Multimedia and Ubiquitous Engineering, 10(4). http://dx.doi.org/10.14257/ijmue.2015.10.4.21

[11] Das, A.K., Asif, A.A., Paul, A., Hossain, M.N. (2021). Bangla hate speech detection on social media using attention-based recurrent neural network. Journal of Intelligent Systems, 30(1). https://www.degruyter.com/document/doi/10.1515/jisys-2020-0060/html

[12] Mathew, Binny Saha, Punyajoy Muhie, Seid Biemann, Chris Goyal, Pawan Mukherjee, Animesh. (2020). HateXplain: A Benchmark Dataset for Explainable Hate Speech Detection.Retrieved from - https://www.researchgate.net/publication/347515513_HateXplain_A_Benchmark_Dataset_for_Explainable_Hate_Speech_Detection

[13] Raizada, R. (2013, July 12). Illustration of how a Gaussian Naive Bayes (GNB) classifier work [Illustration].https://www.researchgate.net/figure/Illustration-of-how-a-Gaussian-Naive-Bayes-GNB-classifier-works-For-each-data-point_fig1_255695722

[14] Multinomial Logistic Regression Example. (n.d.). [Graph]. htps://www.statstest.com/ multinomial-logistic-regression/

[15] Agnihotri, N. (n.d.). How Random Forest Selects Decision [Photograph]. https://www.engineersgarage.com/machine-learning-algorithms-classification

[16] Zvornicanin,E.(2021, February5). BiLSTM [Photograph]. https://www.baeldung.com/cs/bidirectional-vs-unidirectional-lstm

[17] Sazzed, S. (2021, June). Abusive content detection in transliterated Bengali-English social media corpus. In Proceedings of the Fifth Workshop on Computational Approaches to Linguistic Code-Switching (pp. 125-130).