Hate Speech Detection Using DNN(Deep Neural Network)

by

Muhammad Mustaqim
16301174
Mehedi Hasan
16201016

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
October 2020

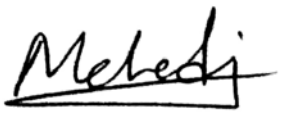
Declaration

It is hereby declared that

1. The thesis submitted is my/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:

Muhammad Mustaqim
16301174



Mehedi Hasan
16201016

Approval

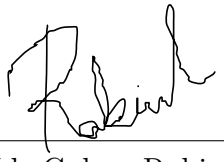
The thesis/project titled "Hate Speech Detection Using DNN(Deep Neural Network" submitted by

1. Muhammad Mustaqim (16301174)
2. Mehedi Hasan (16201016)

Of Summer, 2020 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of B.Sc. in Computer Science on October 5, 2020.

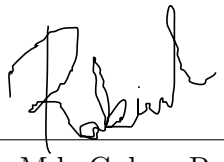
Examining Committee:

Supervisor: (Member)



Md. Golam Rabiul Alam
Associate Professor
Department of Computer Science and Engineering
Brac University

Program Coordinator: (Member)



Md. Golam Rabiul Alam
Associate Professor
Department of Computer Science and Engineering
Brac University

Head of Department: (Chair)

Mahbubul Alam Majumdar
Professor and Chairperson
Department of Computer Science and Engineering
Brac University

Abstract

Hate Speech is defined as offensive or threatening speech that expresses prejudice against a particular group of people on the basis of race, stereotype, epithet, threat, gender, sexual orientation, religion, organization, country. Through the availability of the Internet and Social media, anonymity has made hate speech hard to detect. For the detection of hate speech, the DNN (Deep Neural Network) model can be very effective. Also, there is no such research available that gives the best result when it comes to detecting hate speech from audio speech. In this paper, feature extraction of audio has been done with the help of many audio feature extraction methods-MFCCs, ZCR etc. The DNN (Deep Neural Network) deep learning method is used as it gives better accuracy in sequential data like audio. Moreover, for classification purposes, two different modern classifiers are used to classify the dataset that has been made. SVM(Support Vector Machine) and XGboost classification models are used in our dataset to compare results. From these three models, DNN(Deep Neural Network) performs the best applying the dataset. Lastly, after applying these three different kinds of approaches, the research has been completed by doing analysis and by predicting whether it is hate speech or not.

Keywords: Hate speech; DNN(Deep Neural network); SVM(Support Vector Machine); Prediction; Classification; Accuracy.

Acknowledgement

Initially, we are all thankful to Allah for the wonderful opportunity to learn something different. It was a wonderful experience to us, and we were thankful for overcoming many difficulties. By the grace of Allah, we will make every effort and complete it successfully.

The second thing we want to thank our supervisor Dr. Md. Golam Rabiul Alam

for his outstanding support and diligent commitment throughout the entire process of our study . He provided us with many opportunities and a lot of support from the very beginning of the journey to the end and got us to our desired target. In addition, it encouraged us a lot to progress to our target when we lost our way and felt dissolved.

In the end, we will always be grateful for the support of our parents , friends and

all of the people who gave us audio voice data during the research. We are still really motivated by them and are willing to go through all the obstacles. We also recognize all the support on the internet , particularly from related works, from which we have innumerable benefits and learn a lot from it.

And it could not be feasible for our parents eventually without their full support.

We are now on the verge of our graduation with their kind encouragement and prayer.

Table of Contents

List of Figures

List of Tables

Nomenclature

The next list describes several symbols & abbreviation that will be later used within the body of the document

CSV Comma Separated Value

DFT Direct Fourier Transformation

DNN Deep Neural Network

FFN Feed Forward Network

FFT Forward Fourier Transformation

$MFCC$ Mel-frequency Cepstrum Coefficient

$ReLU$ Rectified Linear Unit

SVC Support vector Classifier

SVM Support Vector Machine

$XGBoost$ Extrme Gradiant Boost

ZCR Zero Crossing Rate

Chapter 1

Introduction

1.1 Background

Voice phishing, bullying, threatening people using these anonymity's has become easier and quicker than expected. The people in general often fall into this trap as there is no filter or verification available in real-time to identify. Often, they fear the circumstances, some become depressed, suicidal, and anti-social which not only holds off their upbringing but also leaves a scar for generations. Criminal activities such as murder, robbing, rape cases are sometimes causing by hate speech. As a result, crimes are increasing day by day. So, detecting hate speech from different voice inputs is a must now. Here we presented a hate speech recognition system which is a part of automatic speech recognition. Keeping all these issues in mind, we will be using many modern approaches to deal with hate speech recognition. We will try detecting with DNN (Deep neural network), SVM, and XGboost classifier for hate speech in real-time.

1.2 Introduction

Due to the massive increase of social interactions and anonymity in online social networks, hateful activities through the online means of communication has become a major issue in the world. Nowadays it is causing massive impacts on the everyday life of people. A lot of criminal activities such as murder, rape, robbing is causing just because of hate speech. Mainly, the areas hate speech covers are social media platforms such as Facebook, Twitter, Reddit, etc. Already the social media companies have taken the necessary initiative against hate speech. Most of them applied a different approach to detect hate speech from posts, text messages, etc. which are in written form. So, there are several techniques available for text to hate speech detection. But the hate speech detection remains ineffective as we are still unable to detect it properly from the audio message or real-time voice calls. Because hate speech detection from audio or real-time voice call is still a challenging task. So, in our research, we are presenting hate speech detection from audio records or real-time voice using digital signal processing and deep neural networks.

Firstly, we will create our dataset. There will be two types of audio data. Hate data and non-hate data. In the beginning, we will use around 500 audio data to test our

system. After creating the dataset, we will process our data, remove the duplicates, and finish cleaning data. r

Secondly, we will use feature engineering techniques to extract features such as zero-crossing rate, the energy of audio, spectral centroid, spectral entropy, spectral roll-off, MFCCs, chroma vector from the audio data. Then we set the classes as hate and non-hate.

Finally, a classification algorithm will be used to predict the accuracy of our dataset. With the help of the classification algorithm, it will be predicted whether it's 'hate' data or "non-hate". For testing purposes, at first, we will perform SVM (Support Vector Machine) for the classification. After that XGboost classifier will be used with more audio data. And finally, DNN(Deep Neural Network) will be used to get better results and accuracy.

1.3 Problem Statement

Hate speech are those that contain abusive words, that express hate or encourages violence towards a specific group of people on the basis of race, threat, gender, sexual orientation, religion, organization, country. The areas hate speech covers are- Social media (such as Facebook, twitter, tumbler, Reddit), audio-video contents, cellular communication, and interaction methods (phone calls, text messages, MMS, voice mail, etc.). Nowadays, hate Speech is causing massive impacts in real life and even in the social life of people. Criminal activities such as murder, robbing, rape are sometimes causing by hate speech. Not only that, the suicide attempts issue, increasing social stigmas are also the impacts of the problem.

Hate speech is not usually occurred gender-biased. It affects equally to both males and females. So, the victims are both male and female. Students, classification of professions, religious persons, specific organizations are mainly affected through hate-related spoken or verbal words. For instance- In 2015, students were most likely to report hate-related words which are according to their race [3.2 percent], followed by ethnicity [1.8 percent]. Rates of students being targeted for their religion, disability, gender, or sexual orientation were around 1 percent each.

Furthermore- In Germany, a correlation was found between anti-refugee Facebook posts by the far-right Alternative for Germany party and attacks on refugees.

Hate related issues usually occur on different occasions. People generally use hate speech at the time of being angry at activities, conversations, different topics, general ideas, beliefs, stereotypes, issues, and many more. Hate speech can also be a product of a lack of knowledge, literature, experience, or social reflection. If the speaker comes from an environment where slang and other hatred views are common, it

can also result in hateful speech towards other people. Speakers may be unfamiliar with members of a targeted group, and not realize they are stereotyping or using language that could be considered hateful.

In this generation, anyone can be anonymous, anything can be spoken anonymously from anywhere. Communicating, interacting, sharing with each other has become so much easier than before that we sometimes do not recognize who is on the other side of the communication. If anything related to hateful audio or text content delivered to anyone is detected and further analyzed then the spreading of hateful speech can be nullified. Also, by detecting the hateful speech from vast text and audio messages we can remove the spread of any threat or violence towards any specific group of people. It is not a very easy task.

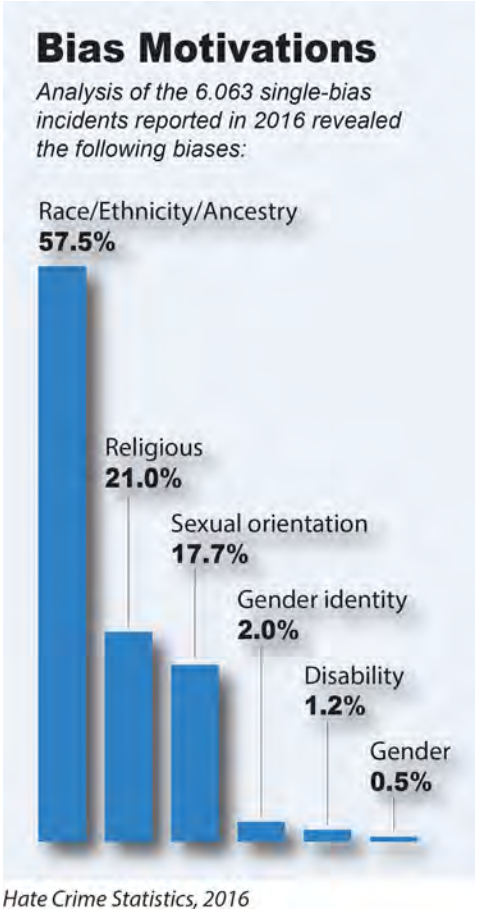


Figure 1.1: Hate crime statistics, 2016

From the figure 1.1, we can see that,
57.5 percent were motivated by a race, ethnicity, or ancestry bias; 21.0 percent were motivated by a religious bias; 17.7 percent were motivated by a sexual orientation bias; The remaining incidents were motivated by gender identity, disability, or gender bias.

People use to identify these manners through text or messages from social-interactions, but it has become more challenging to detect these offensive languages while exchanging information verbally. Some social media platforms such as Facebook and Twitter using artificial intelligence approaches to detect many bad contents. In this technique, hate speech can be detected when it is only in written form, but what about verbal words? It is still a challenge to identify the isolated words accurately from a verbal communication or normal audio file. Because different people have different accents, speaking style which depends on the speaker. Besides background noise is also responsible. Due to these problems, speech detection from a voice or any record is still less efficient than speech detection from the text in terms of accuracy. If it is possible to recognize speech directly from call records or voice calls, a lot of opportunities will be created. Nowadays most of the criminal activities and their plans are conducted via phone calls. So that we can observe them and will be able to stop their activity. In addition to this, we will also be able to identify any kind of conspiracy against the government or the plan of the terrorist groups. Apart from this, in terms of creating a society without hate towards each other, a proper approach to this site is the key.

1.4 Objectives and Contribution

In this world, nowadays where social media is that much available that even children have access to screen touch phones. While communicating with friends and society even children are doing hate crimes. These hate-related crimes that we are all talking about are not going in a straight line. Often people do not care what they say about any human being. Usually, people do not always care what the person behind the screen is who, where, and whom. So they make a stance of words or speech which even hurt someone's emotions, feeling and even that can turn something into revenge, suicide, criminalism, etc. Through our thesis work, we wanted to make that stop. Our main goal is to stop bullying, threatening, harassing, torturing, criminal attempts, racism toward a particular group of the human race, and hatefulness towards each other online. We have the opportunity to do something for the rest of the world with the help of modern technology. By the grace of machine learning, AI, deep learning anything can be possible.

Our main objective of this research is to explore social media platforms, communication platforms, and even gaming platforms where we promote no tolerance towards hate crimes. For this hate propaganda, we also promote obligation towards social media governance where we identify the gaps.

With the help of our collected audio speech data from a huge number of people, we have done the feature extraction methods to extract features from audio speeches. Because we cannot work with this raw audio wav files. So after doing the feature extraction, we come to a decision to classify our dataset. We are applying two machine learning approaches and one deep learning approach to achieve our objective.

We contributed our work for the sake of making the online world beautiful to watch and sound. So for that, we have done some valuable works that are as followed-

- collecting real audio hate speech from going home to home in search of different kinds of voice, with various kinds of voice tones.
- making feature extraction possible with many modern feature extraction methods.
- making a dataset that accomplishes the need for comparing, predicting, evaluating.
- classifying data with some machine learning and deep learning algorithms.
- getting the perfect accuracy to detect the hatefulness of audio data.

1.5 Work Plan

To do a huge research one should walk forward blindly. So, before moving with research we did much planning depending on the dependencies we have to cover up throughout the thesis work. We therefore might be in huge complacent and losing our minds if anything wrong happens. So for this long research, we prepare ourselves by going through many research papers regarding our research topic. So, by reading all these research papers, journals, and conference papers, we have gathered much knowledge that helped us throughout the whole process. However, we had to also study audio signal processing at first. We gathered knowledge about how the audio signals are processed. We also come to the decision of gathering a lot of audio real audio voice samples from the people around us. These audio voices are mixed with both hateful and non-hateful speeches. We emphasized in general speech people communicate with each other and use aggression between human to human. So that we make our data more lively and reasonable for our research.

Furthermore, we proceed to the dataset creation step. To begin with, after collecting the audio data. The data was not ready to be used for research purposes, so we had to choose among the audio files and evaluate the audio files one after one. If the audio contained too long speech we removed them. If the audio contained many noises we made them noiseless. So, basically, if the audio quality is not up to mark we set it to be, we remove it or classify it vice-versa. After that, we have seen that

the audio file is in different formats. We take the audio files and make them wav files so that the audio files can be read through codes.

After that, we make the dataset from the audio files we collected. But before doing that we did the feature extraction part. We executed many feature extraction methods and implemented them with our audio. For each and every audio we get seven features that are further combined into a CSV file. After making the feature extraction done we had to normalize the data because as we knew that if we had to fit the dataset into a deep learning or machine learning model, we have to normalize the data and make them between 0 to 1 and in fractional value.

After the normalization part, we were stuck in the middle of the research. We were confused about which model or classifier we will choose for our dataset to be classified and predicted. After going through a lot of research and studies among various neural network models and classification methods we finally came to a decision to classify our dataset through the machine learning model as long as a deep learning approach. However, we did apply these models and classifiers in our model through learning coding implementation of machine learning and deep learning.

Finally, after implementing all the machine learning and deep learning approaches we came to compare the results of the outcomes we found. We found extraordinary results from our dataset and our model. Later, we come to a decision of final evaluation where we went to an end to our research.

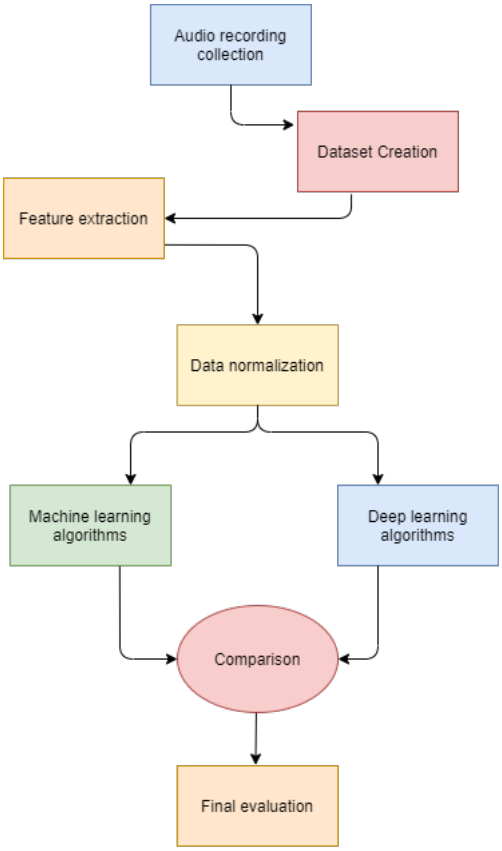


Figure 1.2: Workflow diagram

Chapter 2

Related Work

In this paper, [8] they used LSTM as a classifier and LPC (Linear Predictive Coding) along with MFCC(Mel-frequency cepstrum) feature extraction was used to recognize Indonesian speech digits. They used LPC which takes the coefficient value based on the prediction of the previous value and it takes 12 coefficients. Also, MFCC extracts features based on the sound spectrum and it is the same as LPC. They used two types of LSTM processes- training process and classification process. In this paper, they examined both LPC and MFCC differently along with LSTM. The LSTM model produced high-level accuracy and it can handle voice data so well.edge discovery system and data mining framework for National Basketball Association (NBA). It was aimed to discover several interesting patterns in basketball games. This and related system have been used by several basketball teams over the past decades. Such solutions designed for offline usage and no in game effects were taken care of. There has been some recent works (20) about in-game decision making to find how much time remaining in the game without making any prior prediction model.

In this paper, [9]they proposed a singer recognition system that has been possible by applying deep learning and Feed-Forward neural network approaches. They also used the MLP (multilayer perceptron) network compared with the SVM classifier to classify the gender of the singer. For feature extraction, they used MFCC, spectral flatness, spectral flux, spectral crest, energy, and LFPC. in this paper they also used a GMM classifier to detect vocal and non-vocal parts. The dataset they used (MIR-1K) consists of a huge amount of song clips with maximum information and data. They centered each signal to mean, then they started the training and testing process. Along with an appropriate feature vector, the results they achieved proves the efficiency of their work.

This paper [6]adds improvements in detecting hate-speech from English-Hindi mixed tweets. They used three deep learning models, these are- CNN-1D, LSTM, BiLSTM to remove violent massage. They mainly used CNN-1D as a base in which they basically used globalMAxPooling. Also, to find the final result they applied the sigmoid activation function. They used the other two classifiers to compare their result. They used a large dataset Bohra et al. which consisted of huge tweets full

of Hindi and English tweets. In this paper, they mentioned they used Keras neural network API. They claimed that their approach would outperform the statistical approach by so far.

In this paper, [5] to detect online hate speech they used two methods - Logistic Regression model with context features and A neural model with learning context. It is different as other methods do not contain contextual hate speech recognition. The results found from testing cases are identical. In this paper, they used three basic models these are- character-based neural network, deep neural network, and logistic regression model. The logical regression model shows a 3 percent better result and the neural model shows a 4 percent better result. Both showed 7 percent better results to recognize hate speech. So, what they gave better accuracy's to predict hate speech.

Abusive speech targeting- cyberbullying, Gender, religion, and LGBT are classified and filtered as racist, sexiest, or neither in this paper. Various DNN methods are used (CNN, LSTM, Fast-Text) to identify which one is the best. In this paper, they experimented with multiple classifiers like logistic regression, SVM, random forest, and deep neural networks. They used CNN for sentiment classification, LSTM for capturing long-range dependencies, Fast-Text for averaging word vector, and back-propagation for tuning. They used 'Adam' for CNN and LSTM and 'RMS-props' for fast-text for optimization. They did many tests with many methods but LSTM along with Random embedding learned from deep neural network model combined with gradient boosted decision trees led to the best accuracy values. On the other hand, Combinations of CNN, LSTM, Fast-Text embeddings as features for GBDTs did not lead to better results. [4]

The aim of this paper [3] is to find out how NLP [Natural Language Process] can contribute to the detection of hate speech. Following five major steps of the Natural Language Process, it obtains 91 percent accuracy to detect hate speeches.This paper works on isolated Hindi words recognition where the speech recognition system understands a voice by microphone and converts it into text using KNN [K-Nearest-Neighbour]. A ninety-point zero nine percent precision, night-eight-point three eight percent recall, and ninety-four-point three one percent accuracy prove that this system can be used for speech recognition.

The goal of this paper is to classify online hate speech in general and also a particularly racist and sexist speech on deep learning methods using pre-trained word embeddings and max/mean from simple, fully connected transformations of embeddings. The three most widely used datasets were used. These datasets were named as Sexist/Racist(SR) collected by Waseem and Hovy(2016), HATE collected by Davidson et al(2017) , HAR collected by Golbeck et al(2017). After that These datasets were labeled in the following terms- sexist racist, neither; hate speech or not; harassment or not. The training set consisted of N examples X, Y=1 where

X is the input and Y is the numerical class of the hate speech. In addition to that SWEM-concat was used to handle infrequent and unknown words and capture nonlinear word combinations. They used 300-dimensional embeddings, 1-layer Multi-Layer perceptron Wt with a Relu activation to form an updated embedding space z1: T. Two pooling methods were used, one is max pooling to capture salient word feature from input noted as m and another is to average embeddings z1:T to capture overall meaning of the sentences, denoted as a. As for Preprocessing, they experimented with different approaches while tokenizing the data using Spacy. Using 300-dimensional Glove Common Crawl embeddings and fine-tuning the data. The data was padded each input into 50 words. They used RMSprop with a learning rate of 0.001 and batch size 512. The results are produced from 10- fold cross-validation to allow comparison of previous data. The baseline for the HAR dataset was taken out using a Logistic Regression model. The result showed more efficiency than many complex methods which in total is F1 12 points. They suggested studying the occurrence of such linguistic phenomena in the existing dataset and construct a new dataset that has a better representation of the subtle form of hate speech.[7]

Audio segmentation is used to preprocessing the data. In this paper, they introduced different well-known segmentation methods. They focused on the Log-linear model to determine segmentation from the audio stream. There is also comparison of different speech recognition techniques. From segmentation, the non-speech are detected and removed (music and other unused sounds). Overlapping speeches are separated to reduce the error rate. They used three methods- 1. decoder guided segmentation which is used to silent regions recognized, 2. model-based approach which is used to classify regions divide audio stream, 3. Metric based approach –the distance between the audio stream used. They used the ASR model along with matric based approach along with a different segmentation approach. The front line consists of MFCC and VTLN(Vocal Tract Length Normalization). To recognize unsegmented data, data wrapping factors processed with the speaker-independent acoustic model. CMLLR feature transformations and MLLR mean transformations are estimated using the segment clusters. They did mention the use of the sum of negative logarithm for implementation. They analyzed some feature- segment length, words, boundary length, boundary confidence, wrapping factor variance, BIC score, signal type, sentence end. In training, they used GIS(Generalization Iterative Scaling) for optimization. To detect speech they used 3-state HMMs for the signal type speech, non-speech, pure music, and silence. They used Gaussian mixtures as an emission model. The mixture models were trained for 9.5hrs audio material labeled with 5 classes. For individual signals, they estimated a bigram language model. For the data set, they used American English broadcast news data. They presented a MAP decoder framework for audio segmentation. Among 15s segments, NIST, ASR-based, and MAP, MAP is the best for the audio signal they found. So, in our paper, we must use audio segmentation in the preprocessing step for these reasons. [1]

We investigated a review paper and we get to know about the techniques that were used in various stages of automatic speech recognition. Among these, basic building

blocks is the primary step of speech and language process. In this step natural speech in converted into modified speech and symbol sequence. Then features and parameters are extracted via feature extractor. After that features turn into labeled segments via segmenter and labeler. The next stage is the classification of the speech recognition systems.Words can be classified into several categories such as isolated, connected, and continuous. Speech recognition systems can be classified as discrete or continuous that asr speaker dependent or independent. In a discrete speech system speaker needs to stop between the words, but a continuous speech system faces difficulties finding the start and the endpoint of the word. That's why discrete speech is quite easier than the other one. Furthermore, the size of vocabulary can be small, medium, or large which can affect the complexity of the system. They also mentioned three different approaches for speech recognition, they are acoustic-phonetic approach, pattern recognition, and Artificial intelligence. After all, the paper provided a comprehensive survey of research in the speech recognition system. [2]

For detecting hate speech from the text on social media, some challenges have to be faced like nuance and subtleties in language, differing definitions of what constitutes hate speech, limitations of data availability, interpretability problems. As a review paper, they have reviewed some other papers regarding hate speech from the text on social media. Therefore, they have proposed the mSVM [Multi-View Support Vector Machines] model for this purpose, having some other models. The only motivation of their work is the automatic detection of hate speech that can reduce the spread of hateful content and speech. They have indicated some popular datasets such as HateBase Twitter, WaseemA, WaseemB, Stormfront, TRAC, HatEval, Kaggle. After all, they have reviewed the keyword-based approach for hate speech detection. For the keyword-based approach, this approach is fast and straightforward but having some false alarming problems such as trash, swine, etc. Additionally, they have reviewed the Source Metadata approach which needs additional information like demographics, location, timestamp, or social engagement on that platform which can give further understanding regarding recognition hatred. But this approach has the big issue of unavailability of this kind of data from the authority for external researchers. Another approach is Machine Learning Classifiers, Neuman for feature extraction, Naive Bayes, SVM, Logistic regression for text categorization, Davidson for feature based classification in the form of TF-IDF, Neural ensemble, FastText, BERT, C-GRU. Finally, the proposed model mSVM where they have used case folding, tokenization, punctuation removal for preprocessing. For feature extraction, they simply extract word TF-IDF from unigram to 5-gram and character N-gram from unigram to 5-gram. They used Stamfron, TRAC, HatEval HateBase Twitter datasets. Finally, the evaluation result showing for the StormFront dataset BERT having 0.8201 percent accuracy and mSVM having 0.8033 percent accuracy, for the TRAC dataset BERT having 0.5809 percent accuracy and mSVM having 0.6121 percent accuracy. As having some challenges, there is a need for further research on this problem, including both technical and practical matters. [10]

Chapter 3

Proposed model

3.1 System Model

First of all, we are collecting voice records from people of different gender and ages, which will be used to create our data-set. As we are going to detect hate words from real-time voice or audio records, we need to make the machine learn which word is hate and which is non-hate. That's why we classified all the audio data into three different classes. One is direct hate-data which will contain hateful words. Another one is non-hate. In non-hate data, there are two classes, one is directly non-hate, which will contain only non-hate words, and another one will also be considered as non-hate, but it might have hateful words. Now here's the question. Why an audio data will be considered as a non-hate data even if it contains any hateful, cruel or nasty word. let's ask a question to yourself, in which situation and why a person uses hateful words while they are talking with other persons? The answer is relatively easy. On the one hand, people use hateful words when they become angry, fierce, at the time of threatening others, or while planning about any conspiracy. On the other hand, sometimes people use slang words (e.g. while playing video games, chatting in social media) at his or her friends to make jokes or fun which cannot be considered as hateful behavior at all. So how will a machine be able to differentiate the dissimilarity between the hateful words and non-hateful words? By reading the wave frequency which will be generated from the input audio data. When people become angry, fierce at others, they usually shout and speak loudly. If we convert it into an audio wave frequency, we will find it at a high pitch. Besides this, if a person speaks in a calm, funny, or friendly manner, he or she will not speak in a fierce or loud voice which will give a lower pitch in the output wave frequency. This is the technique that we will use to detect hateful and non-hateful words properly.

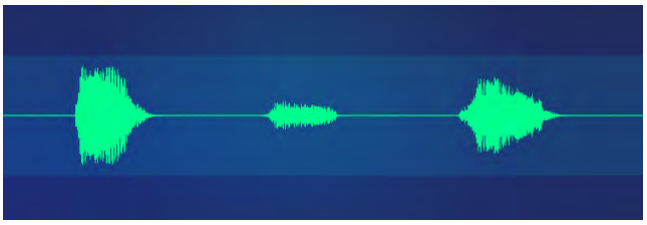


Figure 3.1: Hate data in a louder voice

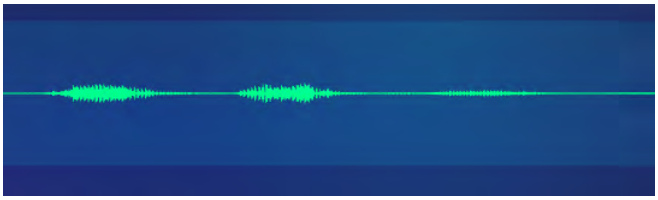


Figure 3.2: Non-hate data in a calm voice

Figure 3.3: A bad example of recorded audio data

Second of all, we are preparing our data-set by following some criteria to acquire the best accuracy from the model which we are going to use in our system. Creating the data-set based on these criteria will help our model to find the pattern in between the audio in our data-set. Firstly, all the audio data of our data-set has a maximum length of 3 seconds containing a minimum of one word or a maximum a sentence of four words. Secondly, we maintained a short duration between each of the words, so that the individual words will be recognizable easily by the system model. Thirdly, as we are classifying the data-set into two classes, the audio data from the hate class contain a hateful word or sentence of the hateful word along with a loud and fierce voice. On the contrary, the non-hate class contains a sentence or a word that does not contain any hate words along with a calm and friendly voice. In addition to this, some of the audio data from the non-hate class contain hateful words. Because, we are classifying the hateful words with a friendly and calm voice as a joke or fun, just like people do to their friends when playing video games, making fun of each other, or chatting.

In our data-set, there are around 500 audio data which is the voice recording of different people. Till now we get a voice recording of 30 different people. Moreover, we collect voice recordings of both hate and non-hate class from each individual. Challenges during the pandemic. We strictly maintain the criteria while recording voice. Initially, we get the MP3 file from each recorder audio. Then we convert

those audio data into WAV format, as this format is more favorable for extracting audio features in python. We put all the audio data (.wav files) into a CSV file and label as per the class they belong to.

9	b- f ashole.wav	1
10	b- s o a b.wav	1
11	Babu - U nigga.wav	1
12	bastard.wav	1
13	bBad.wav	0
14	b-bstrd.wav	1
15	b-C U AGAIN.wav	0
16	b-c u soon.wav	0
17	b-COD.wav	0
18	b-come here.wav	0
19	beautiful.wav	0
20	b-f off.wav	1

Figure 3.4: Labeled data as per the class

In the feature extraction part, we extract several audio features particularly, Zero Crossing Rate, Energy of audio, Spectral centroid, Spectral Roll-off, MFCC (Mel-frequency cepstral coefficients), and chroma. Using these feature extraction methods, we get the corresponding coefficient value from each audio. After that, we get a complete CSV file that contains all the audio data along with seven co-efficient feature values for each audio file.

	A	B	C	D	E	F	G
1	zcr	energy	spectral_centro	rolloff	spectral_rolloff	mfcc	chroma
2	[0.11743800951086956]	[1.73362260246418	[2057.04715388	[3428.460427989	[438.6219853940	[[-382.7268371	[[1.0]]
3	[0.148046875]	[11.5422857342794	[2454.85335078	[4572.844848632	[585.703125]	[[-272.6997680	[[1.0]]
4	[0.134140625]	[1.35590093222492	[2004.45728042	[3945.744140625	[334.5183105468	[[-411.1415710	[[1.0]]
5	[0.1623270425451807]	[6.93237881510699	[2499.52014192	[4537.019954819	[403.2935452748	[[-303.5448913	[[1.0]]
6	[0.11948029891304347]	[0.63979288333743	[1763.15972045	[3351.502632472	[299.0306555706	[[-399.6028137	[[1.0]]
7	[0.14389400158898305]	[5.11996817930912	[2338.40009578	[4499.891998808	[542.5272278866	[[-329.6027221	[[1.0]]
8	[0.12591171264648438]	[6.01351286592623	[1503.73475077	[2693.837356567	[385.9994888305	[[-386.9741210	[[0.890625]]
9	[0.13141610689252337]	[4.52656240133772	[1584.01473747	[2978.927469699	[356.1028311185	[[-383.9568176	[[0.91588783264160
10	[0.18614459859913793]	[6.11877684817845	[2310.70785290	[3716.705111799	[698.4368686018	[[-378.1065368	[[0.93965518474578
11	[0.1444091796875]	[3.02982637092168	[1502.36029098	[2885.045471191	[271.5875244140	[[-460.9257812	[[0.88749998807907
12	[0.13812331211419754]	[17.7074554060634	[2043.92212680	[3930.607096354	[373.1092664930	[[-317.0487670	[[1.0]]
13	[0.11157831869834711]	[0.89814866555755	[2023.77588753	[3475.743720945	[406.4614540289	[[-405.4538879	[[1.0]]
14	[0.11588541666666667]	[3.00078939498764	[1656.90885111	[3049.687114514	[350.2923262746	[[-397.9020080	[[0.92105263471603
15	[0.0846728515625]	[3.20275091850787	[1411.63044512	[2700.479003906	[314.8154296875	[[-428.0217895	[[0.86000001430511
16	[0.11060491597877359]	[2.23215146151259	[1554.82730180	[2836.390081441	[350.3208376326	[[-444.4442443	[[0.85849058628082
17	[0.06986490885416667]	[2.00448989254668	[1264.26386298	[2710.491943359	[248.9776611328	[[-473.9007263	[[0.90625]]
18	[0.187969970703125]	[0.73411808640661	[1790.88697950	[3002.805175781	[670.7592773437	[[-530.4840087	[[0.89999997615814
19	[0.1128479863556338]	[1.54598742165745	[2052.47707901	[3662.160954005	[276.5955105633	[[-368.0093078	[[1.0]]
20	[0.0953626130756579]	[3.17845343670571	[1422.45144823	[2588.092683490	[328.5230134662	[[-428.1631774	[[0.88157892227172

Figure 3.5: Data set with coefficient values

Before implementing our system model, we normalize all the coefficient values of

our data-set. Because there are several types of values such as whole, float, positive, negative, etc. In addition to this, the value range of the features is different from each other, for example, MFCC (value range -500 to +500), Zero crossing rate (value range -1 to 1), Energy of audio (value range -50 to 50), etc. Data normalization turns each and every value within a small range. Here we use the Lambda function to normalize our data-set. After proceeding all the tasks with the data-set, we get a complete CSV where all the seven features are extracted with normalization.

The deep neural network, a classifier, or a machine learning algorithm has been used in our research. Basically, a deep neural network is an artificial network consisting of multiple layers, including an input layer, a hidden layer, and an output layer. Each layer contains multiple neurons. The hidden layer belongs in between the input and output layers. The input data passes through the hidden layer. Since we have seven features per audio, the input dimension of our Deep neural network model will be 7. Furthermore, Deep neural network models give us the accuracy of both training and testing, along with the total amount of loss.

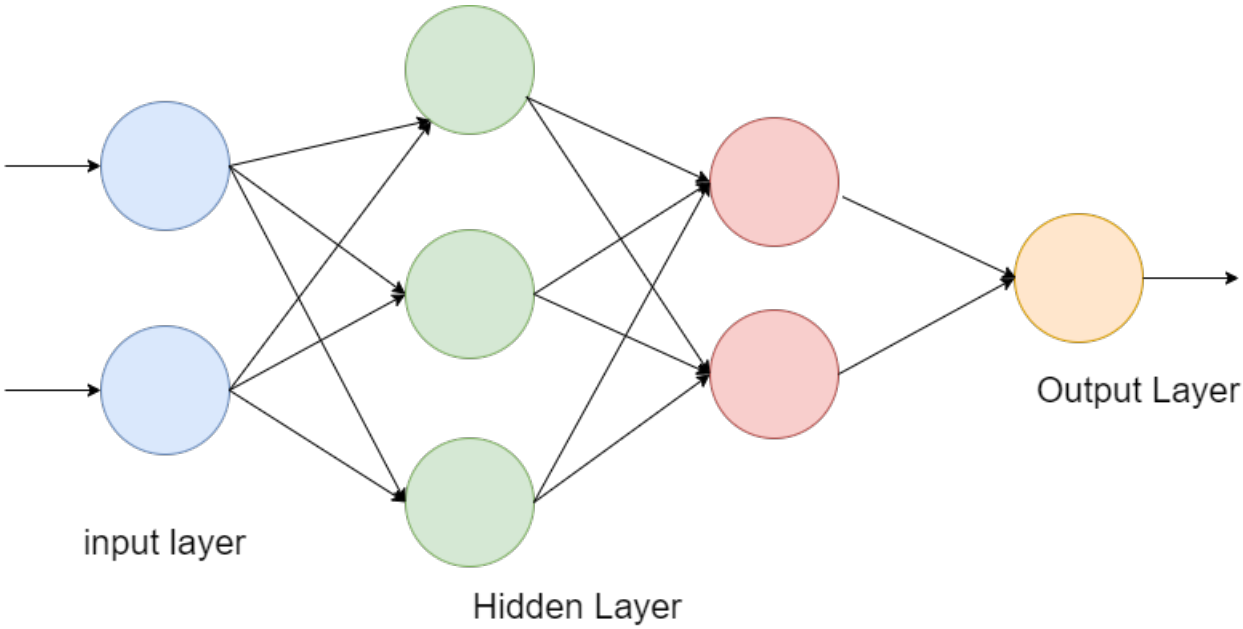


Figure 3.6: Deep neural network

In our research, another machine learning algorithm has been used, which is called the XGBoost classifier. The term XGBoost refers to "Extreme Gradient Boosting" This is one of the most popular classifying algorithms for small to the medium data-set. This classifier uses decision trees to predict the maximum outcome. Besides this, the XGBoost classifier portrays the feature which dominates the others in terms of predicting the better result.

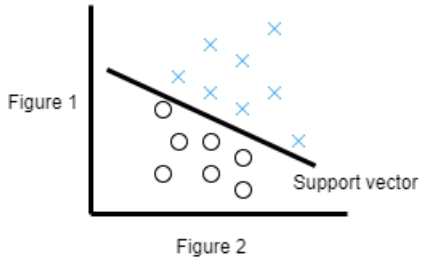


Figure 3.7: Support vector machine

In our research, another machine learning algorithm has been used, which is called the XGBoost classifier. The term XGBoost refers to "Extreme Gradient Boosting" This is one of the most popular classifying algorithms for small to the medium dataset. This classifier uses decision trees to predict the maximum outcome. Besides this, the XGBoost classifier portrays the feature which dominates the others in terms of predicting the better result.

In order to evaluate the performance of each model, training-testing accuracy and the amount of loss have been monitored. We implemented Precision, Recall, and F1 Score in our algorithm. Additionally, the confusion matrix has been executed to get a visual representation of how the algorithms perform.

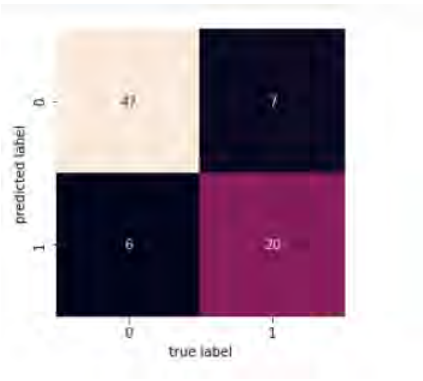


Figure 3.8: Confusion matrix

Finally, we will get the accuracy of the predicted output from the Deep neural network model, Support vector machine, and Extreme gradient boost. We compare

the accuracy between all these algorithms, and we get to know which algorithm provides the best accuracy for our audio data-set.

3.2 Feature Extraction

Feature extraction is an essential part of our research because our system model will learn the similarities and differences among the audio data by analyzing these features. To extract features from our audio data, we use seven different feature extraction techniques. We used Zero crossing rate, Energy of the audio, Roll-off, Spectral Roll-off, Spectral centroid, Mel frequency cepstral coefficient(MFCC), and Chroma vector. At first, we prepared a CSV which contains all the audio that we recorded along with the class(hate or non-hate). After that, we extract all the seven features for each audio data. We put all the coefficient feature values, respectively, to the audio data. After completing the feature extraction operation for each data, we get a complete CSV dataset which we use in our algorithm for evaluation.

3.2.1 MFCCs

In our research Mel frequency cepstral coefficient(MFCC) is one of the most vital features to extract. Mel frequency cepstral coefficient(MFCC) is highly recommended to use in audio speech data. We implemented MFCC in our audio data. The Mel-frequency cepstral coefficients (MFCCs) of a signal are a small set of features that concisely describe a spectral envelope's overall shape. MFCC(Mel frequency cepstral coefficient) works on the Fourier transform of a signal and maps the spectrum's powers above onto the Mel scale using overlapping triangular windows. It takes the value of the log of the forces at each of the Mel frequencies to take the discrete cosine transform of the list of Mel log powers as if it were a signal. The MFCCs(Mel frequency cepstral coefficient) are the amplitudes of the resulting spectrum. MFCCs(Mel frequency cepstral coefficient) is a feature widely used in automatic speech and speaker recognition. Before introducing MFCCs, different feature extraction was used, but they had different sets of problems with noise reduction and accuracy, but the Mel frequency cepstral coefficient(MFCC) was better than all of them.

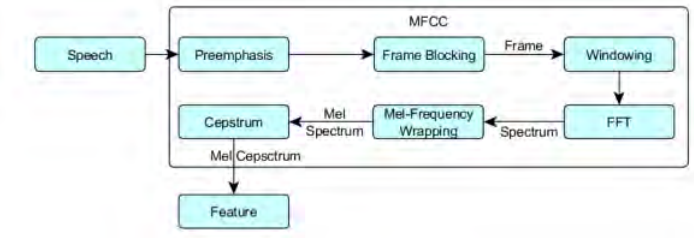


Figure 3.9: How MFCC works

Firstly, we collected the hate speech audio, and then it was passed to preemphasis. Pre-emphasis refers to filtering which emphasizes the higher frequencies of audio or speech data. After the preemphasis was done then it was passed to frame blocking and windowing. The input speech signal is segmented into frames of 20 30 ms with an optional overlap of 1/3 1/2 of the frame size. In terms of sample points, the frame is equal to the power of two in order to facilitate the use of FFT(Forward Furrier Transformation). Then, each windowed frame is converted into a magnitude spectrum by applying DFT(Direct furrier Transformation). After that, it comes to the Mel-frequency wrapping term. A Mel is a unit of measure based on the human ears perceived frequency. Mel-Spectrum is computed by passing the Fourier transformed signal and a set of band-pass filters known as Mel-filter bank. Since they only contain information from a given frame. So, the cepstral coefficients are usually referred to as static features. After doing so, we get the MFCC feature.

At the first step, we get 12 valued 2d arrays from the first feature extraction process. From each audio, we get that 12 values and 2d array of it. Most of the values are in the range of -500 to 500. As there are 12 values in one row and that array is 2d array, we cannot take that value for our model. So, we make that 12 values into 1 corresponding value with the help of nmfcc parameter. Basically, nmfcc takes that 12 valued array and gives 1 value which is well fitted for our classifier. Moreover, MFCC features have negative values and long fractional values. After that, we have to remove the brackets before and after the values for each value. We designed a code that makes automatic cleaning of the data.

The frequency measure can be converted to Mel-frequency by this following forluma-

$$Mel(f) = 2595 \log(1 + f/700) \tag{3.1}$$

We implemented this feature extraction method in our audio data and here is one of the audio plot images that we have plotted to view the data in image form.

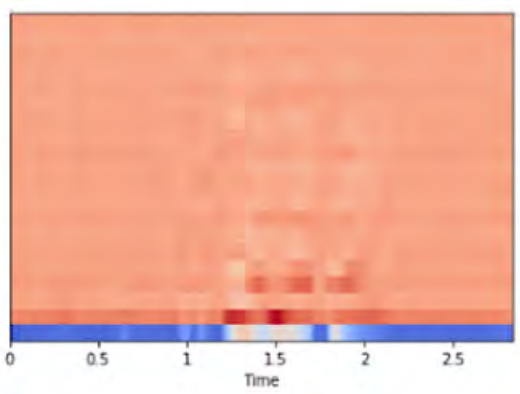


Figure 3.10: Mel-frequency cepstral coefficients

3.2.2 Chroma Vector

Chroma vector is an important feature in case we take audio data and extract features from it. we implemented a chroma vector feature extraction method and found some features of it. Firstly, we input each audio data to the chroma vector and the chroma vector gives us 17 values in a 2d array. These values are in the range of [0-1]. It was a huge array to provide for our models. So, we had to make it as one valued. We made the 17 valued array into single value with the help of nchroma. Basically, this nchroma parameter takes the whole array and gives only one single value which is easy to give to the model. After having that single-valued array, we clean the data by removing the brackets of it. Chroma vector is a 12-element representation of the spectral energy. The chroma vector is computed by grouping the DFT(Direct Fourier Transformation) coefficients of a short-term window into 12 bins. Each bin represents one of the 12 equal-tempered pitch classes of the input data. Each bin produces the mean of log-magnitudes of the respective DFT(Direct Fourier Transformation) coefficients. Here is the equation-

$$V_k = \sum_{n \epsilon S_k} \frac{X_i(n)}{N_k}, \tag{3.2}$$

where k element of 0,.....11,

Here, Sk is a subset of the frequencies that correspond to the DFT coefficients and Nk is the cardinality of Sk. The chroma vector, vk, is usually computed on a short-term frame basis. Here, k indicates the pitch class and i indicates the frame number. So, this final V is a matrix representation of the sequence of chroma vectors.

Here is a histogram that has time in x-axis and pitch class in y-axis.

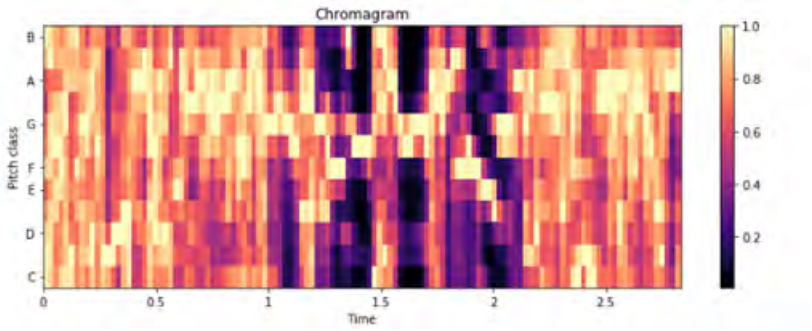


Figure 3.11: Chroma vector

3.2.3 ZCR

We also implemented ZCR(zero-crossing rate) feature extraction method for our audio data to extract features from them. We implemented ZCR in all the data. These features are the most important feature to look at in our research. For each of the audio data we get single-valued data. These data are from [0-1] and mostly fractional values.

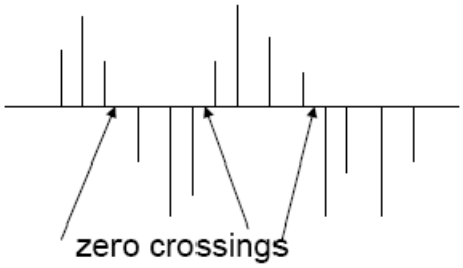


Figure 3.12: measurement of Zero Crossings.

Basically, Zero crossing rate(ZCR) represents how many times any signal crosses the horizontal axis. The ZCR(zero Crossing rate) does a shift in the performance of audio frames from a positive value to a negative value. So, ZCR(Zero Crossing rate) gives a positive value if there is a lot of noise. On the other hand, it gives a negative value when there is a silent audio frame. However, Zero cross rate(ZCR) is the number of times the signal changes value from positive to negative or negative to positive, divided by the length of the frame. It can be interpreted as a measure of the noisiness of a signal. Here is a histogram of a hate audio. We take the audio file and make a plot of ZCR(Zero Crossing Rate) implemented into it.The ZCR is defined according to the following equation:

$$Z(i) = \frac{1}{2W_L} \sum_{n=1}^{W_L} |sgn[x_i(n)] - sgn[x_i(n-1)]|, \tag{3.3}$$

where sgn(.) is the sign function, i.e.

Here is a histogram of a hate audio. We take the audio file and make a plot of ZCR(Zero Crossing Rate) implemented into it. In the horizontal axis there is measured time(t). we can see from plot that, though there were lots of noise we have the peaks.

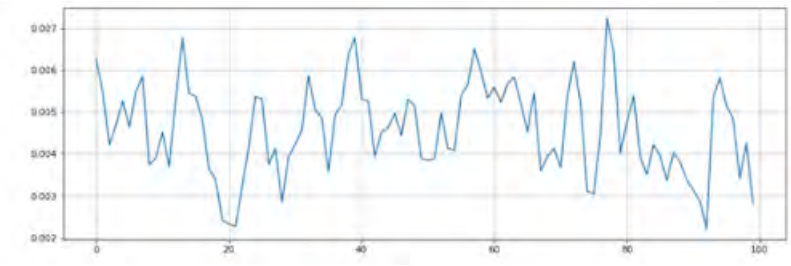


Figure 3.13: Zero Cross rate

We have computed the presence of the Zero Crossing Rate(ZCR) feature for each of the audio data. That means, we have computed how many times the signal of an audio file has changed the horizontal axis. For this, we have made our data in a way that each and every audio does not exceed three seconds. So, for each hate speech and non-hate speech, the zero-crossover rate is calculated.

3.2.4 Spectral Centroid

We also implemented the Spectral Centroid feature extraction method in our audio files. It is also an important feature extraction method for our implementation. Basically, it indicates where the center of mass" for a sound is located and is calculated as the weighted mean of the frequencies present in the sound. If the frequencies in an audio are the same throughout then spectral centroid would be around a center and if there are high frequencies at the end of sound then the centroid would be towards its end. Also, higher values correspond to brighter sounds.

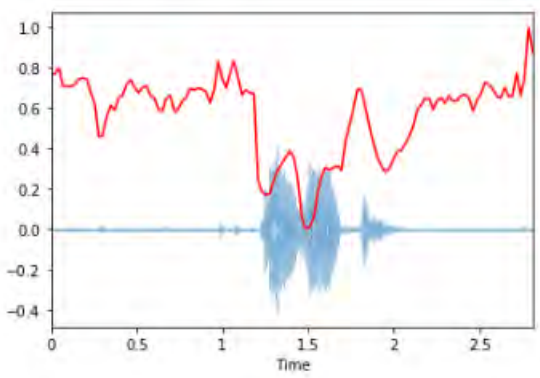


Figure 3.14: Spectral Centroid

For implementing Spectral Centroid, we take the audio data and extract features from it. It gives a single value for each audio data. The values are fractional value and all the values are greater than two thousand. So, furthermore, we implemented normalization For the values of the spectral centroid, Ci, of the ith audio frame is defined as:

$$C_i = \frac{\sum_{k=1}^{Wf_L} kX_i(k)}{\sum_{k=1}^{Wf_L} X_i(k)} \tag{3.4}$$

3.2.5 Spectral Roll-off

We also extracted Spectral Roll-off features. Spectral roll-off measures the frequency below which is a specified (usually 90) percentage of the total spectral energy. Basically, we take each and every audio file and input in our spectral roll-off code. Then it gives an array for each audio. Finally, we pre-process the data with data cleaning (removing parenthesis from the array) and normalize the data afterward. We applied spectral roll-off in our audio.

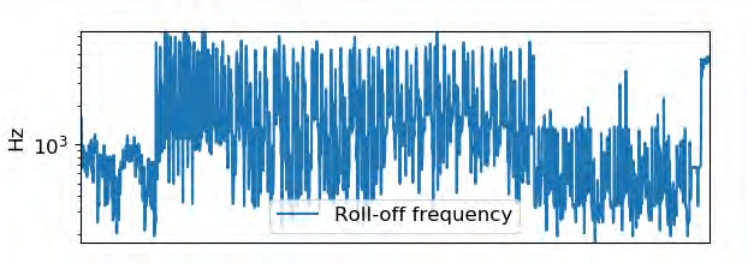


Figure 3.15: Spectral Roll off

3.2.6 Energy

We applied the Energy feature extraction method from Librosa also. Energy can be described as the power of the signal. A signal strength can be measured by its energy. Energy is measured by the total of squares of signal values, normalized by the frame length of the signal. The energy envelope alternates rapidly between low and high energy states. Because speech signals contain weak phonemes and short periods of silence between words. We extracted Energy features from our audio data. Actually, what we did was, we gave the audio data in our code as wav file and it gave us an array for each and every audio, we gave in. later we clean the feature by removing parenthesis. Furthermore, we did normalization also.

3.3 Data-set Description

3.3.1 Audio data collection

As the purpose of our research is to detect hateful words from real-time audio or voice records, our first task is collecting voice records from the people. We were able to reach around 30 people and collect a total number of 500 audio data for the data-set. Initially, collecting audio data from 30 people is not that much good, but we have plans to reach out to more people and collect more audio data to enhance our data-set.

The main goal of our thesis or research is to make the system or the model understand the difference between hateful words and non-hateful words. While collecting audio records for our data-set, we followed several criteria so that the system model will be able to find the similarities and dissimilarities between the audio records, which will lead to better accuracy.

We used a decent quality headset with a microphone to record voice from the people. As we were in a pandemic situation, we were not able to collect much voice recordings. Somehow we managed around 30 people, including males, females, and children. We tried to collect at least 20 audio data from each person and maintain a ratio of 40-60 between hate data and non-hate data. An online voice recorder has been used to record the voices. Due to the availability of the internet everywhere, this was the only way that is affordable for us.

Some techniques were strictly followed while recording the audio and collecting the data. Firstly, we have collected all the audio data where each of the audio got a maximum time length of three seconds. Although the maximum time length is 3 seconds, there is some audio data which has a length of one or two seconds. The majority of the data contains a complete sentence with the highest three to four words. For example “Happy Birthday to you”, “I will kill you (as a hate

word)" Besides some of the data contains a single word like "Hello", "Morning", "nigga(as a hate word)" etc. Secondly, our main concern was to make the system model recognize the difference between hate and non-hate words. So we kept a little duration between each word in the voice record so that each word will be visible individually after plotting it into a wave frequency. But the question is how do we manage a short duration inside the sentences? We convinced the people to keep a small break after speaking each word. In this regard, we had to remove a lot of audio data due to. Thirdly, here comes the vital part, data classifying. As we have two classes, one is hate, and another is non-hate. The data from hate class may contain only a single hate word or a sentence with at least one hate word inside it. For example "I will kill you.", "Bustard" etc. In addition to this, all the sentences were spoken in a fierce, loud, and angry voice which sounds like he or she is threatening someone. Moreover, a loud voice gives us a higher pitch in the spectrogram image.

3.3.2 Creating CSV

After collecting the audio recording, we got a total number of 500 data. The data were in mp3 format. So we convert all the mp3 files into .wav format. Because this format is convenient for creating the dataset and extracting features from the audio in python. We insert all the audio data into a CSV file and label it as per their class. Here we use two types of classes, Zero and one. "Zero" indicates the data is non-hate data. On the other side "One" indicates, the data contains a hate word and it's hate data. Though our data set has only two classes, there are three variations in it. First of all, if any audio data contains any hateful word which is in a fierce, arrogant, and loud voice, it will be considered as hate-data. Second of all, the audio data will be considered as a non-hate data if it doesn't contain any hateful word. In addition to this, a hateful word can be considered as a non-hate data, only if it is spoken in a normal and calm tone. In that case, we are assuming it as a joke. For example "I will kill your man", "Are you f**king kidding me?", "Yoo Nigga". These are the types of sentences people generally use with their friends while they make fun of each other. Besides this, it doesn't sound threatening, as they are speaking it in a calm or funny voice.

After labeling all the audio files according to their class, we get a proper CSV file that contains all our data.

	A	B	C
1	name	class	
2	aniversary.wav	0	
3	are you a gigolo.wav	1	
4	are you fine.wav	0	
5	are you fkin brain dead.wav	1	
6	are you from dhaka.wav	0	
7	asshole.wav	1	
8	b- bloody mf.wav	1	
9	b- f ashole.wav	1	
10	b- s o a b.wav	1	
11	Babu - U nigga.wav	1	
12	bastard.wav	1	
13	bBad.wav	0	

Figure 3.16: Dataset after labeling as per the class

3.3.3 Feature extraction

Feature extraction is one of the important tasks which covers a major portion of our research. After preparing our labeled data-set we apply feature engineering on it. Here we are using some time-domain features along with some spectral features. The features are Zero crossing rate, Energy of the audio, Roll-off, Spectral Roll-off, Spectral centroid, Mel frequency cepstral coefficient(MFCC), and Chroma vector. The library that we have used for extracting features from the audio is "Librosa" from python. Using the feature extraction technique we get seven coefficient values for seven different features. There are several varieties on the coefficient value of each feature such as positive, negative, and float numbers. Furthermore some of the features [Mel frequency cepstral coefficient(MFCC) and Chroma vector] return a two-dimensional array, which is not convenient for applying in our algorithm.

```
[[-430.7644857014281, 89.98317679028692, 4.162358663488669, 56.48934024016838,
34.575788416305905, 8.991319649414393, 18.05629705859214, 24.052047270760628,
5.046660511873654, 16.499267236721554, 5.396182054219969, 8.07113784159692,
4.844292975748025, -1.8838166300123582, 5.280726313920531, 10.73315918487876,
7.945656739021225, 5.466332989581041, 4.331157610615083, 6.533153226622665245]]
```

Figure 3.17: Coefficient values from MFCC

3.3.4 Data normalization

To implement our data-set in the algorithm in an appropriate way, data normalizing is a must. As some feature like [Mel frequency cepstral coefficient(MFCC) and Chroma vector] returns a whole two-dimensional array as a coefficient value, we have to normalize these values in a proper way. Basically, the Mel frequency cepstral coefficient(MFCC) and Chroma vector return around twelve and seventeen values respectively. Using an extra parameter in the feature method we bring out one value instead of multiple values. In addition to this, almost all the coefficient values

come with a second parenthesis around it which can be a reason for the error in the system code. So, we use a simple python code to remove all the parentheses from the coefficient values. Moreover, there are several types of values such as integer, float, positive, negative. Furthermore, the range of the coefficient feature values is quite different from each other. For example, Mel frequency cepstral coefficient(MFCC) returns the value from range of -500 to +500, Zero crossing range returns values from the range of 0 to 1. That is why it is necessary to simplify all the coefficient feature values and convert them into a common scale.

zcr	energy	spectral_cent	rolloff	spectral_rollc	mfcc	chroma	class
0.11743801	1.733622602	2057.047154	3428.460428	438.6219854	-382.72684	1	0
0.148046875	11.54228573	2454.853351	4572.844849	585.703125	-272.69977	1	0
0.134140625	1.355900932	2004.45728	3945.744141	334.5183105	-411.14157	1	0
0.162327043	6.932378815	2499.520142	4537.019955	403.2935453	-303.54489	1	1
0.119480299	0.639792883	1763.15972	3351.502632	299.0306556	-399.60281	1	0
0.143894002	5.119968179	2338.400096	4499.891999	542.5272279	-329.60272	1	1
0.125911713	6.013512866	1503.734751	2693.837357	385.9994888	-386.97412	0.890625	1
0.131416107	4.526562401	1584.014737	2978.92747	356.1028311	-383.95682	0.91588783	1
0.186144599	6.118776848	2310.707853	3716.705112	698.4368686	-378.10654	0.93965519	1
0.14440918	3.029826371	1502.360291	2885.045471	271.5875244	-460.92578	0.88749999	1
0.138123312	17.70745541	2043.922127	3930.607096	373.1092665	-317.04877	1	1
0.111578319	0.898148666	2023.775888	3475.743721	406.461454	-405.45389	1	0
0.115885417	3.000789395	1656.908851	3049.687115	350.2923263	-397.90201	0.92105264	1
0.084672852	3.202750919	1411.630445	2700.479004	314.8154297	-428.02179	0.86000001	0
0.110604916	2.232151462	1554.827302	2836.390081	350.3208376	-444.44424	0.85849059	0
0.069864909	2.004489893	1264.263863	2710.491943	248.9776611	-473.90073	0.90625	0
0.187969971	0.734118086	1790.88698	3002.805176	670.7592773	-530.48401	0.89999998	0
0.112847986	1.545987422	2052.477079	3662.160954	276.5955106	-368.00931	1	0
0.095362613	3.178453437	1422.451448	2588.092683	328.5230135	-428.16318	0.88157892	1

Figure 3.18: Coefficient values before normalization

	zcr	energy	spectral_centi	rolloff	spectral_rolloff	mfcc	chroma
0	0.002523851	0.00085861	0.002683369	0.002433279	0.003070398	0.00267826	0.002545
1	0.003181663	0.005716543	0.003202298	0.003245482	0.004099981	0.00190831	0.002545
2	0.002882805	0.000671536	0.002614767	0.002800411	0.002341662	0.0028771	0.002545
3	0.003488557	0.003433396	0.003260565	0.003220056	0.002823096	0.00212416	0.002545
4	0.002567741	0.00031687	0.0023	0.00237866	0.002093245	0.00279635	0.002545
5	0.003092414	0.002535764	0.003050388	0.003193706	0.003797745	0.00230651	0.002545
6	0.002705958	0.00297831	0.001961586	0.001911896	0.002702035	0.00270798	0.002267
7	0.002824253	0.002241869	0.00206631	0.002114232	0.002492756	0.00268687	0.002331
8	0.004000419	0.003030444	0.003014264	0.002637855	0.004889129	0.00264593	0.002392
9	0.003103486	0.001500581	0.001959793	0.002047602	0.00190114	0.00322548	0.002259
10	0.002968397	0.008769964	0.002666248	0.002789667	0.002611803	0.00221866	0.002545
11	0.002397921	0.000444826	0.002639968	0.002466837	0.002845271	0.0028373	0.002545
12	0.002490484	0.0014862	0.002161398	0.002164453	0.002452082	0.00278445	0.002344
13	0.001819697	0.001586225	0.001841438	0.001916609	0.00220374	0.00299523	0.002189
14	0.002377001	0.001105517	0.002028235	0.002013069	0.002452281	0.00311015	0.002185
15	0.001501461	0.000992763	0.001649202	0.001923716	0.001742869	0.00331628	0.002307
16	0.004039647	0.000363586	0.00233617	0.002131179	0.004695383	0.00371224	0.002291

Figure 3.19: Coefficient values after normalization

3.3.5 Final Dataset

After completing all the steps mentioned above, we get the final CSV of our dataset.

Attributes	value
Total data	500
Hate data	200
Non-Hate data	300
Male	410
female	90

Table 3.1: Dataset overview

3.4 Methodology

The deep neural network, a classifier, or a machine learning algorithm has been used in our research. Basically, a deep neural network is an artificial network consisting of multiple layers, including an input layer, a hidden layer, and an output layer. Deep neural networks(DNN) are Feed Forward Network(FFN) where the data flows from the input layer to the output layer. Basically, a deep neural network is an artificial network consisting of multiple layers, including an input layer, a hidden layer, and an output layer. The input, output, and hidden layers are consisting of many neurons igniting each other. The DNN(Deep Neural Network) uses back propagation to obtain perfect results according to the data-set we provide. Back propagation is an algorithm based on the gradient descent of artificial neural networks. It means, when the neurons ignite each other through weights in layers if the training process somehow drops data or loses data, it collects those errors. Through back propagation, the so-called lost data is again run in the neurons to find their pattern. So, partial computations from one layer are reused in the computation of the gradient for the previous layer also. Since we have seven features per audio, the input dimension of our Deep neural network model will be 7. In machine learning, the layers are basically called dense layers as they densely exist in the model. Every time we add layers, we have to add activation functions as one of the parameters along with the neuron numbers in a particular layer. Here, we used the RelU activation function in

the starting layer, in the hidden layer we used the Tanh function and RelU function both, and lastly for the output layer we used the sigmoid activation function. When compiling the model, we defined the loss function as binary crossecntropy and the optimizer we used is adam. We used binary cross-entropy as we are doing binary classification. The loss function stands for how we can evaluate how good (or bad) are the predicted probabilities.

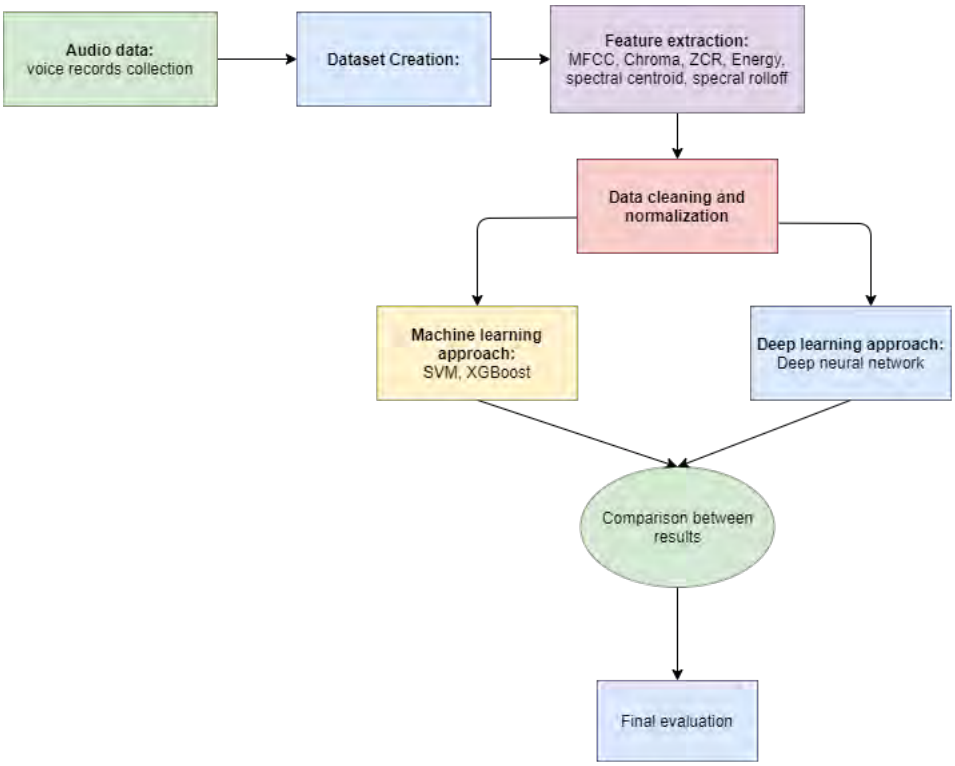


Figure 3.20: Methodology diagram

We started our research by collecting real-time audio records from different people to create our dataset. After that, we apply several processes to make our dataset more reliable such as data cleaning, data preprocessing, data normalization, etc. In addition to this, we have implemented machine learning algorithms along with deep learning algorithms. In this portion, there will be a description of all the processes or methods that have been used in our research.

3.4.1 Data Prediction Through Deep Neural Network(DNN)

In the DNN [deep neural network] model, each layer contains multiple neurons which functionality works as like human brains. Basically, a neuron works as per the activation function used in it. So, a neuron in the neural networks uses an activation function to provide output from the input. There are several activation functions such as RelU Function [The Rectified linear Unit], Sigmoid Function, Tanh Function. All these functions are mainly used to make the output range on a scale.

For example, the sigmoid function returns an output value between 0 to 1, the Tanh function returns between -1 to 1. Besides, the RelU function returns either 0 or 1. This is why the RelU activation function is used for the purpose of classifying data. In our model, we also used the RelU activation function in our DNN [deep neural network] model.

Equation for RelU activation finction

$$1if z > 00if z < 0 \tag{3.5}$$

Equation for Sigmoid activation finction

$$Sigmoid : 1/(1 + e^{-z}) \tag{3.6}$$

Equation for Tanh activation finction

$$Tanh : 1 - e^{-2z}/1 + e^{-2z} \tag{3.7}$$

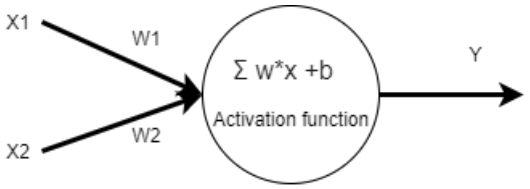


Figure 3.21: Functionality of a neuron

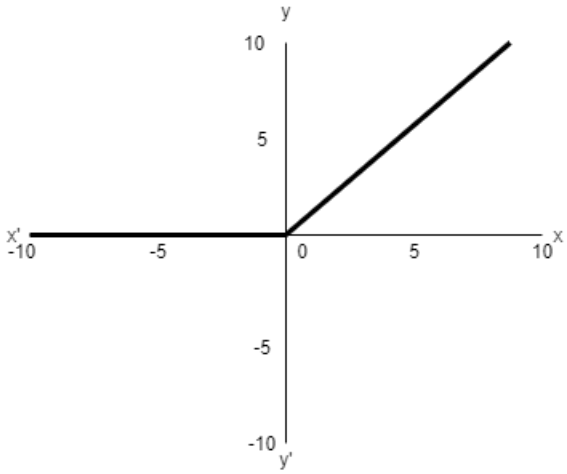


Figure 3.22: RelU activation function

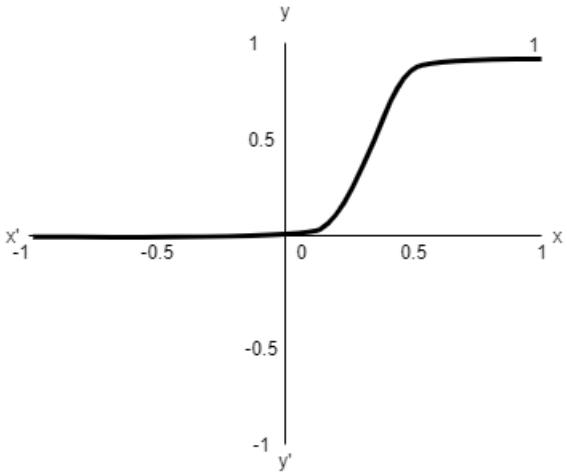


Figure 3.23: Sigmoid activation function

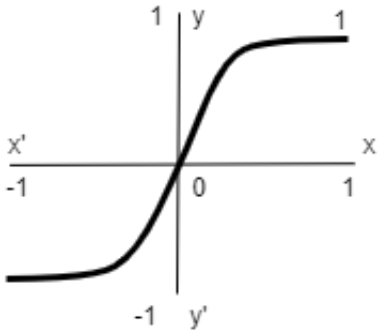


Figure 3.24: TanH activation ffunction

The word “Deep” in Deep neural networks refers to the hidden layers in it. There can be multiple hidden layers. Each layer is fully connected to the other layers. The output of the neurons of the previous layer is sent as an input to the next layer. In the figure, the leftmost layer is called the input layer. In the input layer, the number of neurons depends on the dimension of the input. In our research, we have extracted 7 different audio features. That’s why we used an input dimension of 7 to proceed with all the features into the Deep neural network. The Rightmost layer from the figure is called the output layer. Our output layer consists of 1 neuron, as we are heading to a binary classification just like yes or no.

We created a DNN [deep neural network] model to fit our dataset and evaluate it. We optimized our deep neural network model as we prepared our dataset. So that the algorithm can predict the dataset more precisely. That is why we strictly followed some criteria while preparing our dataset. We keep the length of each audio data maximum of 3 seconds. We used a decent quality headset to record voices from the people. Besides we tried to make sure there is no background noise in the voice recording. In addition to this, we did data normalization which is quite favorable for a machine learning model in terms of data training and predicting.

As we are using Deep learning techniques, our deep neural network has more than one hidden layer. We have extracted 7 features from each audio data. That’s why the input dimension of our DNN model is 7. This means coefficient values from 7 features will be sent as an input value to the 7 neurons of the input layer. Our model has 4 hidden layers. Each layer has 12, 15, 8, 10 neurons respectively. RelU activation Function [The Rectified linear Unit] has been used in all the hidden layers. Here we have used RelU activation function to predict whether it’s hate data or not. RelU activation function is used to make the output value either 0 or 1. On the output layer, we have only one neuron along with an activation function named Sigmoid. Sigmoid activation function returns the value in between 0 and 1. For example 0.44, 0.92, 0.84 etc. That is why the Sigmoid activation function is used in the output layer to calculate accuracy.

As the input dimension of our DNN(Deep Neural network) model is 7, it will take 7 feature values in each neuron of the input layer. Then each neuron calculates the output value and passes it to the next layer. the figure shows the process of calculating the output value. Generally, in a dense neural network, all the neurons inside a layer are connected to each and every neuron of the other layer. Which is also called as fully connected. As a result, we get output value or the predicted from the output layer. If the predicted value does not match with the actual value then the model calculates the amount of the loss using a loss function. The loss function calculates the difference between the actual value and the predicted value and then calculate the square of it. the figure shows the process of calculating the loss value. After calculating the loss, the weights of all the neurons get updated using an optimizer. An optimizer is mainly used to change or update the attributes inside an algorithm. Although there are different types of optimizer, the “ADAM”

optimizer is the most popular one. The process of calculating new weight is called backward propagation. the figure shows the process of calculating the new weights. Through the forward propagation and the backward propagation, the DNN(Deep neural Network) algorithm reduces the amount of loss. As the loss reduces, the accuracy of the system model increases gradually. Sometimes dropout layer is used to handle the over fitting problems in the model.

$$(output)y : [x1 * w1 + x2 * w2 + ... + xn * wn] + bias \tag{3.8}$$

$$(Loss)z : (y(actual) - y(predicted))^2 \tag{3.9}$$

$$(Weightupdate)W_{new} : W_{old} - \eta * \delta h/\delta w_4 \tag{3.10}$$

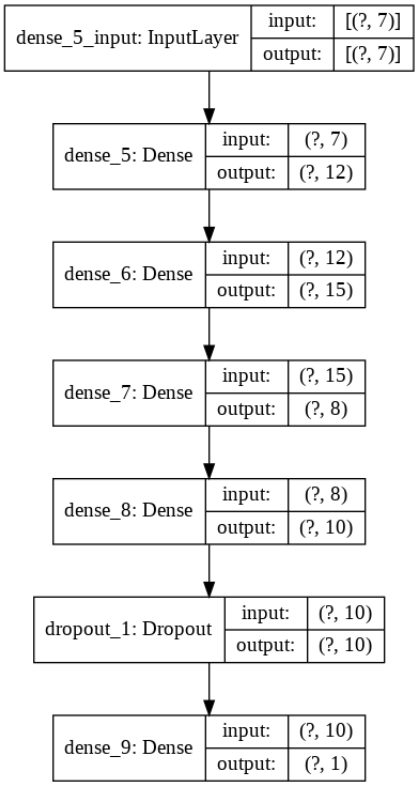


Figure 3.25: Sequential diagram of our model

After creating the model and compiling it, we split our dataset into training and testing data. We kept the testing ratio 0.2, which means only 20 percent of data will be tested while 80 percent will be trained. Adam optimizer is used in the model. A dropout layer has been implemented to reduce the over fitting problem. We set Epoch to 1000 and batch size of 50. So the whole process of forward-propagation and backward-propagation will repeat 1000 times in training.

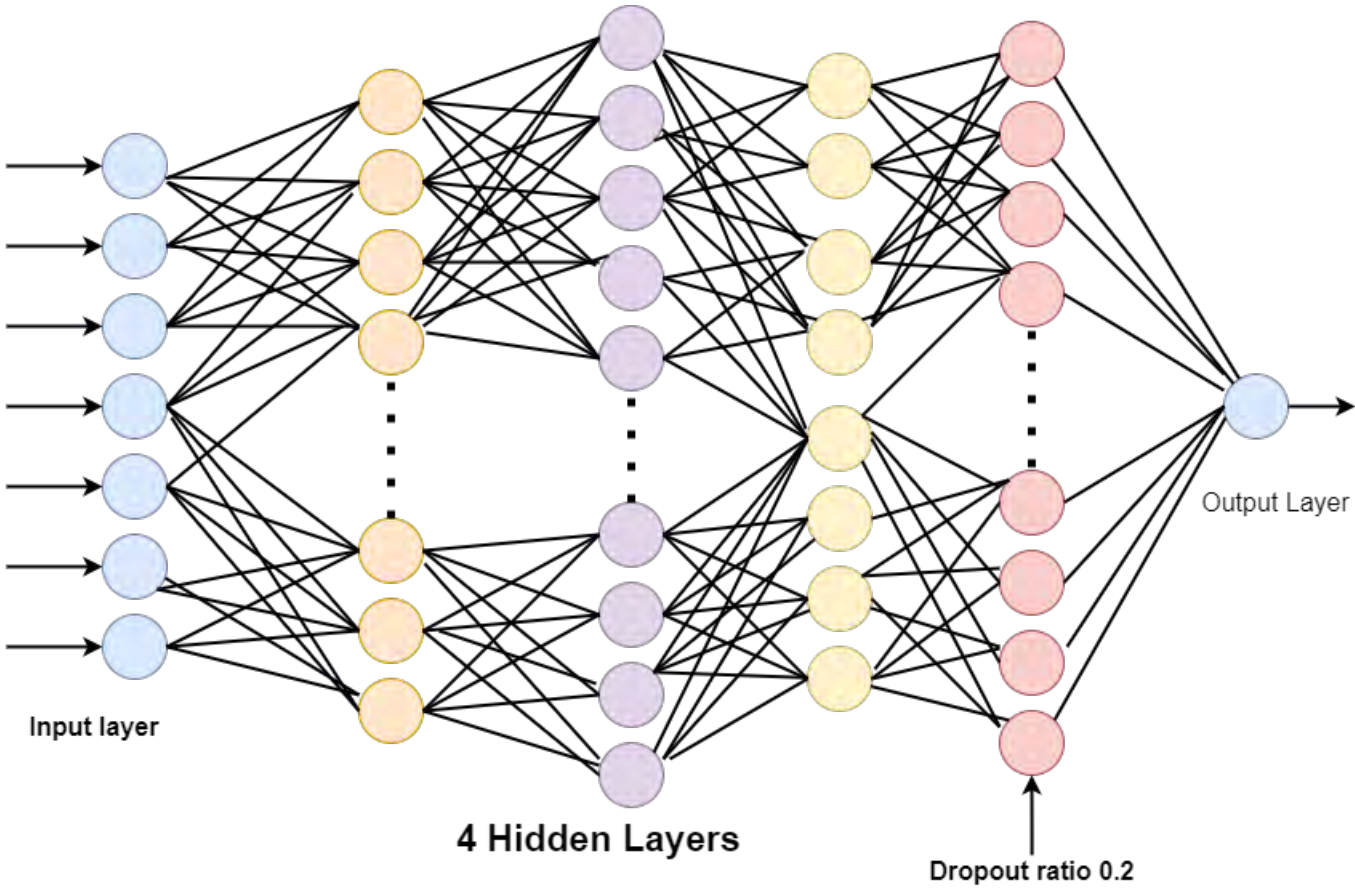


Figure 3.26: Dnn model we used in our research

3.4.2 SVM(Support Vector Machine

For testing our data-set to be working perfectly we applied Support Vector Machines. Supervised learning which analyzes data used for classification and regression analysis. SVM(Support Vector Machine) differs from the other classification algorithms in the way that it chooses the decision boundary that maximizes the distance from the nearest data points of all the classes. An SVM(Support Vector Machine) doesn't merely find a decision boundary; it finds the most optimal decision boundary. According to the SVM(Support Vector Machine) algorithm, we find the points closest to the line from both the classes. These points are called support vectors. Now, we compute the distance between the line and the support vectors. This distance is called the margin. Our goal is to maximize the margin. The hyper plane for which the margin is maximum is the optimal hyper plane.

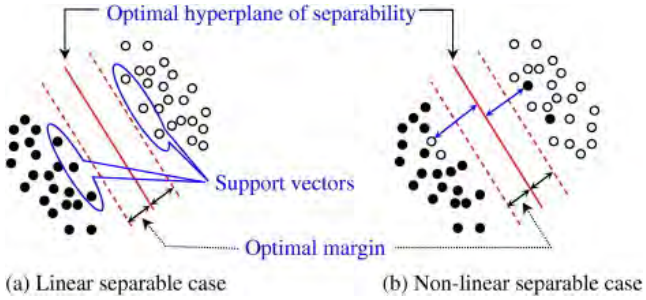


Figure 3.27: How support vector works

SVM is a supervised learning model and it is built to make predictions measuring the data we give it. Actually, SVM is a modern and powerful classifier and it is more powerful if we give it a binary classification- true or false. It has an SVC classifier that is a really strong algorithm in the labeled dataset. After the audio feature extraction, from audio wav files, is done, we did some necessary cleaning to our data. After that, we found the final dataset which is csv file which we imported into our SVM classification model. All the columns of the csv file are being stored in a variable except the class column. Because the class column is a labeled column which consists only two classes 0 for non-hate 1 for hate identification. So, in another variable we stored the class columns data. Here, the first variable contains attributes when the second variable contains the corresponding libeling of the dataset. When dividing the data into labels is completed now, we move onto next step which is dividing the dataset into two portions. One is training data and another one is testing data. One more thing, here we firstly imported many libraries. One of them is Scikit-learn. If provides model selection library which contains built in train test split method that allows us to divide the data into training and testing data sets seamlessly. Finally, we move on to training the algorithm. So, we have to train our SVM on training data. Scikit-learn provides built in classes for SVM which helps in the classification of data. The SVM is written as SVC in the SVM library. It is a class that takes kernel type parameter. Here, we used linear kernel for our data. We also tried polynomial kernel, Gaussian kernel, sigmoid kernel but these are not

suitable with our dataset as we did binary classification. We trained the 80 percent of our data and testing them with 20 percent of our data in purpose of classification. The fit method of SVC is responsible for training the algorithm on training data, which is passed as a parameter. After doing so, we make predictions of the result we found. This predict method is also from SVC class provided by SVM.

For successful SVM classification, we needed to use confusion matrix, precision, recall and f1 measure metrics. The Sci-kit learn's metrics library contains the two most useful methods which are widely used in report making. These are- classification report and confusion matrix built in methods, that provides the information of the result of classification.

$$(W*, b*) = min\frac{||W||}{2} + C\sum_{i=1}^{n} Z_i \tag{3.11}$$

where C is how many errors; Z is called zeta and it is i=1

3.4.3 XGBoost Classifier

XGBoost is very much popular and powerful in machine learning for giving higher performance when it comes for classifying data. XGboost stands for eXtreme Gradient Boosting. Basically, XGboost is popular for many reasons. This classifier is faster than any other classifier as it is written is C++. The core XGboost algorithm uses multicores which does powerful computations easily on very large datasets also. Also, XGBoost has some internal important parameters such as, cross-validation, regularization, user-defined object function, missing values, tree parameters, scikit-learn compatible API etc. The XGboost uses a gradient boosting framework as its main feature. XGBoost is also popular for classifying small to the medium dataset.

In our research we implemented the XGBoost classifier and it gave us a tremendous classification report. Basically, the XGBoost generates a decision tree every time when combining the weak learning data and deliver and improvement of accuracy in the data. But, after the combination of weak Learning data, there also exists some weak data which are misclassified. So, for second time, the classifier gives more weights to the tree with the data that are misclassified previously. Again, after the second classifier, there also exists some of the minor weak data. So again, and again, it increases the weight of the tree and adds up the rest weak learning data that were not added previously. By doing this, the XGBost gradually takes the weak entities and make them classified by strong approach.

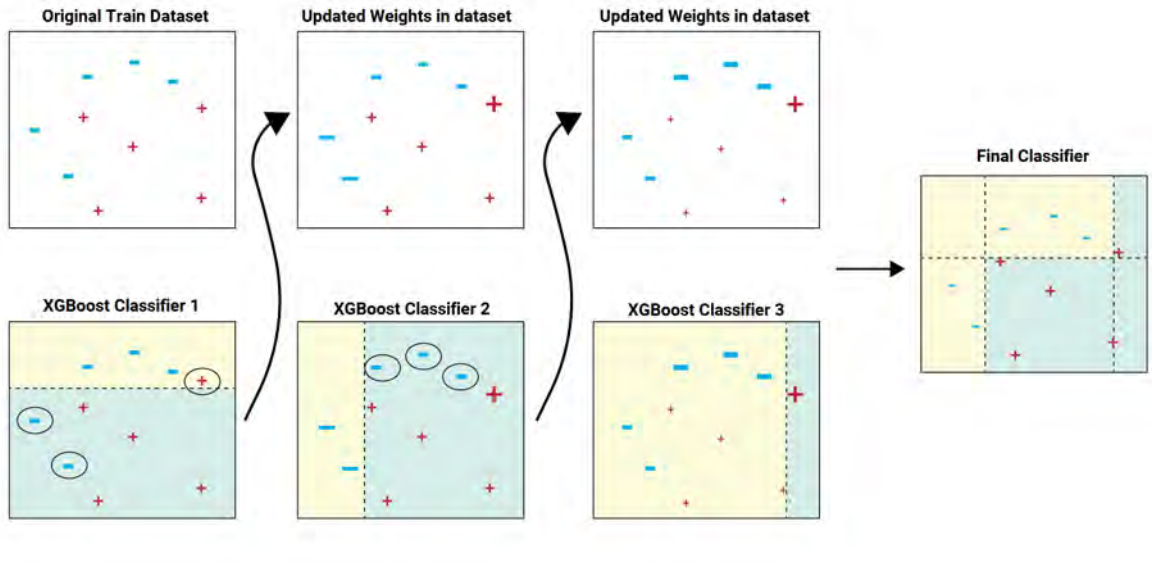


Figure 3.28: How XGBoost works

The XGBoost classification model has XGBClassifier by default and we created a variable and fit the classifier into the dataset of ours. Firstly, we imported many libraries that are needed to do our implementation. Then we imported our dataset into the model as csv file. After that, for classification purposes, we had to split the data into train and test data separate and put them in training variable and testing variable. For instance, we take 80 percent of our data for training purpose and 20 percent of the data as testing purposes. We also make separate the class column and put it in a variable and for the other columns we put them in another variable. The training portion is used for making XGBoost model. On the other hand, testing data are used to make the prediction from the dataset. We can also evaluate the performance of the classification model from it. Lastly, to make the prediction successful we used the default function provided by sci-kit learn named model-predict function. After doing so, we can evaluate the prediction performance when we compare them with predicted values. Here also, we will use the function from sci-kit learn named accuracy score function.

3.4.4 Classification Report of SVM and XGBoost

Confusion Matrix

The confusion matrix is a performance measurement for machine learning classification problem solution. In this classification, report output can be two or more classes. The confusion matrix is a table with 4 different combinations of predicted and actual values.

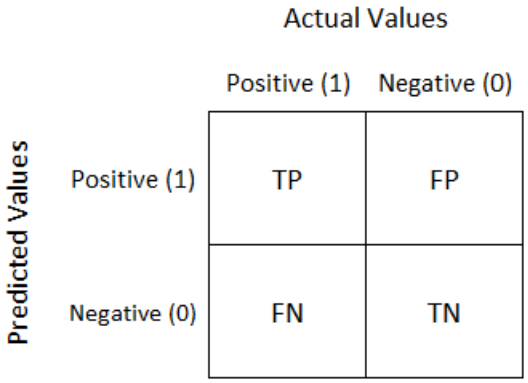


Figure 3.29: Confusion matrix

True positive(TP): what we predicted is positive and it's true.

True Negative(TN): what we predicted is negative and it's true.

False Positive(FP): what we predicted is positive and it's false.

False Negative(FN): what we predicted is negative and it's false.

Accuracy

The accuracy of a model depends on the true positive, true negative, positive samples, and negative samples altogether.

$$ACC = \frac{TP + TN}{TP + TN + FN + FP} \tag{3.12}$$

Precision

Precision is the mathematical equation of true positive values divided by the total number of true positive values along with false-positive values. Precision value works with the TP and FP values. So we can see that the precision is the ratio of

the exactly positive predictions out of all the positive prediction which is made by the classification method. Precision is applicable when there is cases of minimizing the false positives.

$$PRC = \frac{TP}{(TP + FP)} \tag{3.13}$$

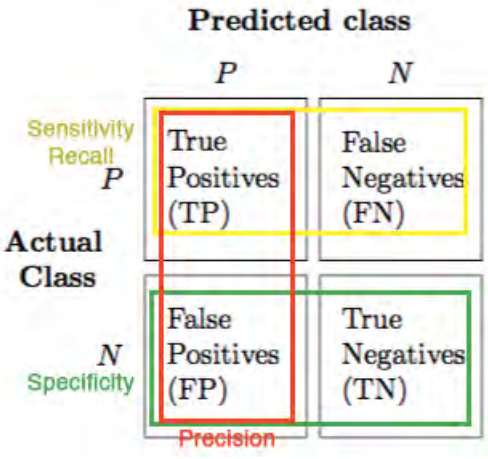


Figure 3.30: How precision is calculated

Recall

Recall is the out put of the ratio between the number of true positive values and the combined number of true positive values along with false-positive values. basically, recall stands for the number of the actual positive predictions made from all the positive predictions. It is really helpful finding the imbalance of a classification model. Recall is applicable when there is minimizing false negative values are necessary.

$$RCL = \frac{TP}{(TP + FP)} \tag{3.14}$$

F1 Score

F1 score is also known as f-measures. It is a way to combine the precision and recall operation into only one measure that can have both the functionalities from them. Also, it does not matter what is the precision or recall give, it can be that there is imbalance and disorder in both the operation but f1 score provides excellent result in one score.

$$F1Score = \frac{(2 * PRC * RCL)}{PRC + RCL)} \tag{3.15}$$

Chapter 4

Implementation and result analysis

This chapter describes the full implementation of our proposed model and portrays how accurate our system model is. As the purpose of our research is to detect hate words from the real-time audio records or voice. We Tried several approaches and compared them to bring out the best model for our system. First of all, we collect Real-time voice recordings from the people. Then we create our Data set. We extracted audio features from each audio data. Here we used Zero crossing rate, Energy of the audio, Roll-off, Spectral Roll-off, Spectral centroid, Mel frequency cepstral coefficient(MFCC), and Chroma vector. After finishing the feature engineering process, we normalized the dataset. Data normalization made our dataset ideal for implementing it into the machine learning and deep learning algorithms. We used DNN [deep neural network], SVM [Suport vector machine], XGBOOST [Extreme gradient Boosting] in our research. Here DNN [deep neural network] is the deep learning algorithm for prediction and SVM [Suport vector machine] classifier and XGBoost [Extreme gradient Boosting] classifier is the machine learning algorithms for data classification. These models are some of the best algorithms for data science. We get results output from each model individually, and we compare them to sort out which algorithms give the best output for our dataset. Besides, confusion matrix, f1, precision, and recall have also been used to evaluate the output result.

4.1 Data pre-processing

The dataset we have used in our research contains real-time voice records from people of different ages and gender. It holds around 500 audio data which have been collected from 30 peoples. After implementing feature extraction methods, we get a dataset containing coefficient values from 7 features for each audio data. But the values we get are not suitable for applying to any machine learning algorithms. We need to follow some steps to make our dataset prepare for the algorithms because most of the features return coefficients values which are different from each other in terms of the value range and type. For example, some of the features return value range from 0 to 1, some return only negative values, some return a two-dimensional array. On the other hand, almost all the coefficient values come with a "[ ]" (third parenthesis) with it. As a result, the machine learning model considers that integer

value as a string input. So we applied some lines of python code that removes all the brackets from our dataset. Furthermore, we used the "lambda" function to convert all the coefficient features into a common scale of 0 to 1. The figure shows the functionality of how the "lambda" function calculates each of the values. This process is also called as data normalization.

$$\lambda_x = \frac{X - X.Min}{X.Max - X.Min} \tag{4.1}$$

4.2 Results

4.2.1 Deep Neural Network

The DNN [Deep neural network] model has been used in this research as our deep learning method. We used the DNN [Deep neural network] algorithm to predict hate and non-hate data from our dataset. To run this deep learning algorithm, we followed all the steps that should be done. We extracted features from audio, did some pre-processing, and we also normalize the dataset. Besides, we also removed all the unnecessary columns from the dataset.

	zcr	energy	spectral_centroids	...	spectral_rolloff	mfcc	chroma
0	0.002524	0.000859	0.002683	...	0.003070	0.002678	0.002545
1	0.003182	0.005717	0.003202	...	0.004100	0.001908	0.002545
2	0.002883	0.000672	0.002615	...	0.002342	0.002877	0.002545
3	0.003489	0.003433	0.003261	...	0.002823	0.002124	0.002545
4	0.002568	0.000317	0.002300	...	0.002093	0.002796	0.002545
5	0.003092	0.002536	0.003050	...	0.003798	0.002307	0.002545
6	0.002706	0.002978	0.001962	...	0.002702	0.002708	0.002267
7	0.002824	0.002242	0.002066	...	0.002493	0.002687	0.002331
8	0.004000	0.003030	0.003014	...	0.004889	0.002646	0.002392
9	0.003103	0.001501	0.001960	...	0.001901	0.003225	0.002259

[10 rows x 7 columns]

Figure 4.1: Dataset with normalized coefficient values

After the normalization part, we created a feature column that contains only the columns that are necessary to fit into the model. We also split our dataset into train data and test data. Here our test data size is 0.2. That means 20 percent of data out of the total data has been used for testing, and the rest of the data has been used for training.

After splitting data into train and test, we created our DNN [deep neural network] model. Here we have used a total of 5 layers: one input and one output layer and

three hidden layers. One dropout layer has been used, and the dropout ratio has been set to 0.2. Tanh function, RelU Function [The Rectified linear Unit], and Sigmoid function are used as the activation function in each layer. To calculate the amount of loss, "binary crossentropy" has been used as a loss function. We used the "Adam" optimizer to change the attribute throughout the neural network.

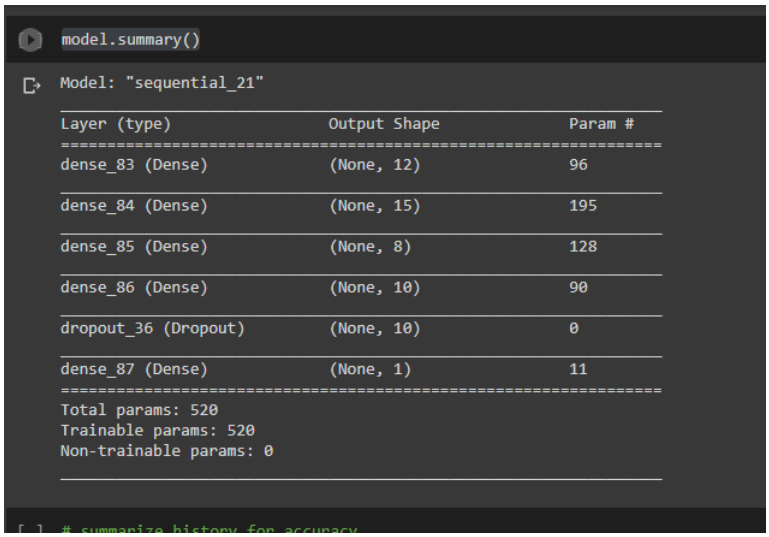


Figure 4.2: Sequential of dnn model

Finally, we fit our train and test data into our model. Here the epoch has set to 1000, and the batch size has set to 50. It needs to be mentioned that there are a total number of 500 data in our dataset.

```
Code  + Text
Epoch 986/1000
6/6 [==============================] - 0s 5ms/step - loss: 0.4110 - accuracy: 0.8381 - val_loss: 0.4445 - val_accuracy: 0.7917
Epoch 987/1000
6/6 [==============================] - 0s 5ms/step - loss: 0.4291 - accuracy: 0.8489 - val_loss: 0.4328 - val_accuracy: 0.8250
Epoch 988/1000
6/6 [==============================] - 0s 5ms/step - loss: 0.4014 - accuracy: 0.8381 - val_loss: 0.4409 - val_accuracy: 0.7917
Epoch 989/1000
6/6 [==============================] - 0s 5ms/step - loss: 0.4199 - accuracy: 0.8381 - val_loss: 0.4380 - val_accuracy: 0.8000
Epoch 990/1000
6/6 [==============================] - 0s 5ms/step - loss: 0.4065 - accuracy: 0.8381 - val_loss: 0.4328 - val_accuracy: 0.8250
Epoch 991/1000
6/6 [==============================] - 0s 5ms/step - loss: 0.3895 - accuracy: 0.8525 - val_loss: 0.4465 - val_accuracy: 0.7833
Epoch 992/1000
6/6 [==============================] - 0s 5ms/step - loss: 0.4060 - accuracy: 0.8345 - val_loss: 0.4334 - val_accuracy: 0.8167
Epoch 993/1000
6/6 [==============================] - 0s 5ms/step - loss: 0.4062 - accuracy: 0.8525 - val_loss: 0.4330 - val_accuracy: 0.8250
Epoch 994/1000
6/6 [==============================] - 0s 5ms/step - loss: 0.4080 - accuracy: 0.8417 - val_loss: 0.4449 - val_accuracy: 0.7917
Epoch 995/1000
6/6 [==============================] - 0s 4ms/step - loss: 0.4125 - accuracy: 0.8489 - val_loss: 0.4330 - val_accuracy: 0.8250
Epoch 996/1000
6/6 [==============================] - 0s 5ms/step - loss: 0.3961 - accuracy: 0.8453 - val_loss: 0.4370 - val_accuracy: 0.8083
Epoch 997/1000
6/6 [==============================] - 0s 5ms/step - loss: 0.4024 - accuracy: 0.8381 - val_loss: 0.4329 - val_accuracy: 0.8250
Epoch 998/1000
6/6 [==============================] - 0s 5ms/step - loss: 0.4214 - accuracy: 0.8381 - val_loss: 0.4407 - val_accuracy: 0.7917
Epoch 999/1000
6/6 [==============================] - 0s 5ms/step - loss: 0.4259 - accuracy: 0.8381 - val_loss: 0.4326 - val_accuracy: 0.8250
Epoch 1000/1000
6/6 [==============================] - 0s 5ms/step - loss: 0.4217 - accuracy: 0.8309 - val_loss: 0.4368 - val_accuracy: 0.8083
```

Figure 4.3: Training and testing of the dnn model

From our DNN [deep neural network] model, we achieved 83 percent training accuracy and 42 percent loss. On the other hand, we achieved around 80 percentage testing accuracy along with a 43 percentage loss. The difference between the training and testing accuracy proves the consistency of our dataset.

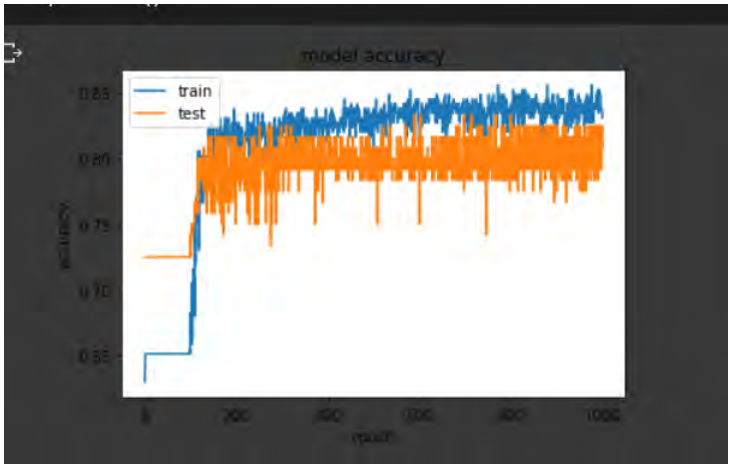


Figure 4.4: Model accuracy

The figure shows that, the accuracy of both the training data and testing data. Here the blue line indicates the training accuracy and the orange line indicates the testing accuracy. This curve line describes how the algorithm achieves better accuracy gradually.

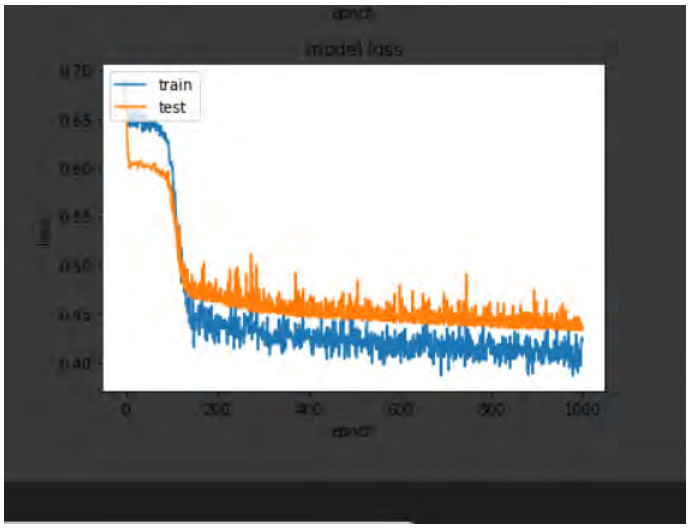


Figure 4.5: Model loss

The figure shows the rate of loss of both the training and testing data. Here the blue line indicates the training loss and the orange line indicates the testing loss. This curve line describes how the algorithm learns and gradually the loss is getting lower.

Attributes	value
training accuracy	84
Testing accuracy	80
trainig loss	43
testing	44

Table 4.1: DNN result overview

4.2.2 XGBoost

XGBoost classifier is also used to classify our data. From this classification, we got the precision, recall, f1 measure and support terms for both hate-speech and non-hate speech data. We also get the accuracy of 86.25 percentage from our feature extracted audio dataset. From XGBoost we get the best score as we have small dataset. XGBoost uses XGBclassifier named a built in classification algorithm to

classify the data. This algorithm has many parameters. Some of them are 'Booster', 'gamma', 'learning rate', 'max depth', 'n jobs', 'random state', 'verbostiy' and many other parameters. These parameters alltogather make the XGBoost classification successful.

```
XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
              colsample_bynode=1, colsample_bytree=1, gamma=0,
              learning_rate=0.1, max_delta_step=0, max_depth=3,
              min_child_weight=1, missing=None, n_estimators=100, n_jobs=1,
              nthread=None, objective='binary:logistic', random_state=0,
              reg_alpha=0, reg_lambda=1, scale_pos_weight=1, seed=None,
              silent=None, subsample=1, verbosity=1)
```

Figure 4.6: XGBClassifier

After fitting the training data, we went to the next step which was making prediction for the test data. We predicted the testing data using a default function named predict, which takes as a parameter called X test, and we stored it into y pred value. We rounded the values we found in y pred and put them in a variable called predictions. So. Now it comes to evaluate our predictions. We predicted the score of the XGBoost classification method using the accuracy score function. This function takes input from two parameters. One is the y test, and another one is the variable we made earlier. We printed the accuracy, and before that, we multiplied the value with 100, so it doesnot comes with fractional value. Accuracy of our XGBoost classifier we get is 86.25 percentage.

After the feature measurement, we can have a look at the scores we get from XG-Boost. We found out that, for hate-speech the precision, which is also a positive predictive value, is 0.74, which means 74 percentage . For non-hate speech the precision score is 0.92 or 92 percent. We found another term which is recall. Recall stands for the total relevant values or instances that were actually extracted by doing the classification. Both of these precision and recall terms are based on the measurement of how much the values or data are relevant to each other. Then there comes the f1 measure term. If we want to learn about the measurement of test accuracy, there comes the f1 score. Basically we computed the f1 score from precision and recall formula.

	precision	recall	f1-score	support
0.0	0.92	0.88	0.90	56
1.0	0.74	0.83	0.78	24
accuracy			0.86	80
macro avg	0.83	0.85	0.84	80
weighted avg	0.87	0.86	0.86	80

Figure 4.7: Result of XGBoost

We implemented a confusion matrix to evaluate the result we have got from the XGBoost [Extreme Gradient Boosting]. The figure shows the overview of how accurately the XGBoost classifier predicted the actual value. According to this confusion matrix, This classifier successfully predicted 49 data as 0 and 20 data as 1. On the other hand, the classifier wrongly 11 values where 7 values as 0 and 4 values as 1. So, the total number of 69 data was successfully predicted and 11 data were mispredicted.

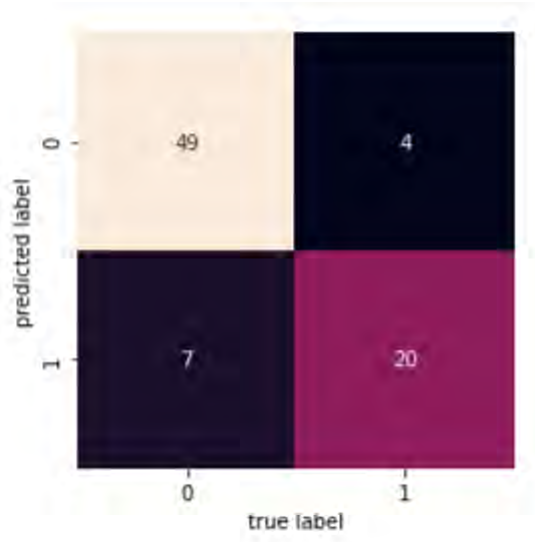


Figure 4.8: The confusion matrix

From classifying our dataset with XGBoost we also get the decision tree we mentioned in the methodology part. Xgboost works on a set of class cation and regression trees just like decision trees. Firstly, we classify the features and according to this node we divide it into two different leaves and assign them the score of the corresponding leaf and it contains decision value. From one node to another the decision tree forms just like a binary tree to make the prediction. The tree takes only two decision – 'yes, missing' and another one is 'no'. If the decision it made is 'yes, missing' then it goes to the left-hand node and if it gives 'no', then it goes to the right node. Then for the left most node, it makes the decision and does just like the previous one. Thus, the tree goes to every node and checks if there is any weak entity. If it found any weak entity, it goes to the previous one and runs with the weight again. It was possible by calling the function provided by xgboost. We plotted the tree using xgb.plot tree function which has parameter called number of trees and we make that four.

Figure 4.9: The decision tree

While executing the XGBoost classifier, we can also get which feature was the most valuable feature in our dataset. We can clearly see that by plotting feature importance from XGboost classification result. In this plot, we plotted the F score in horizontal axis and features(f0-f6) in vertical axis. We can see that the f1, which is the energy feature, scores the best in our model, and it is the most responsible performance scorer from all the features. To execute our result, the energy feature plays the most vital role, according to XGBoost. After that, we can see that the MFCC(Mel-Frequency cepstral coefficient) scores the second highest in importance of features. It scores 86. The f6 is a chroma vector, and it scored 71, which is the third most important feature. Lastly, f3,f0,f2 and f4 are respectively, roll off , ZCR(Zero crossing rate), spectral centroid and spectral roll off.

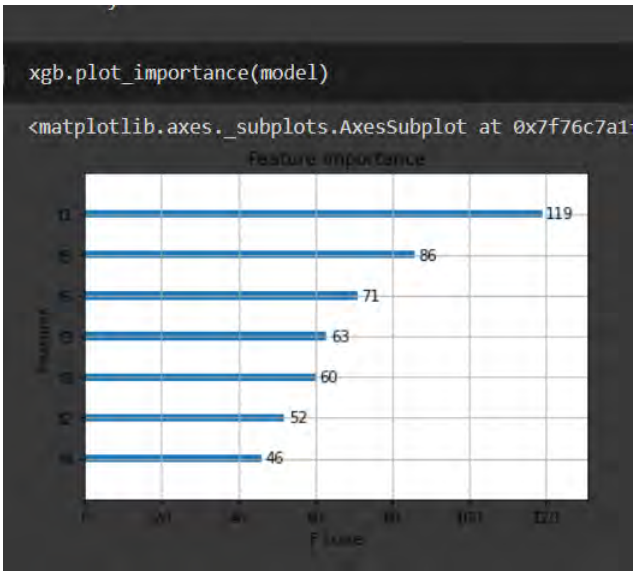


Figure 4.10: Feature dominance diagram

4.2.3 SVM(Support Vector Machine)

When we finished making the feature extracted audio dataset, we found a correlation between them to find a definite pattern and solve our problem by applying different classification methods. The correlation we found from the features are given-

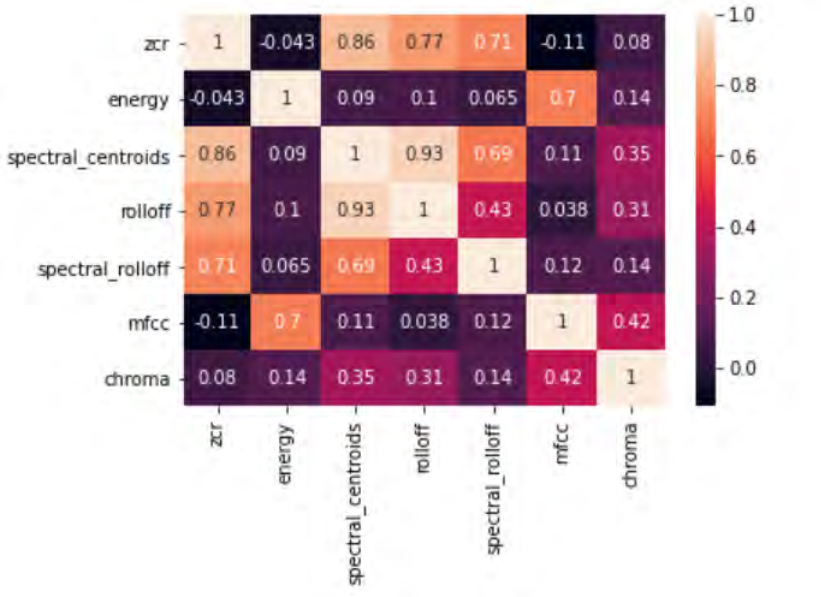


Figure 4.11: Confusion Heatmap of SVM

So when we apply our dataset to SVM classifier, it read the dataset and divided the dataset into train and test data. We split our dataset as a way that there could be 20 percentage test data to train with 80 percent of the train dataset. After that,

we applied the classifier provided by SVM from sci-kit learn. The kernel we used is linear. Then we fit the classifier with fit function and it was called by SVClassifier. It has many parameters that includes- 'cache size', 'class weight', 'coef0', 'decision function shape', 'random state', and 'verbose' etc. Then we predict the test classes using the predicted default function from the SVC Classifier. As a parameter, it takes input as X test. We put it in a variable called y pred.

```
SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='linear',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

Figure 4.12: SVCClassifier

After the classification, there comes the classification report. From the report we can see that for hate-speech, the precision, which is also a positive predictive value, is 0.84 that means 84percentage. For non-hate speech the precision score is 0.80 or 80 percentage. We found another term which is recall. Here we find that the recall for hate speech is 0.57 and for non-hate speech data it is 0.94. The Recall stands for the total relevant values or instances that were actually extracted by doing the classification. Both of these precision and recall terms are based on the measurement of how much the values or data are relevant to each other. Then there comes the f1 measure term. If we want to learn about the measurement of test accuracy there comes the f1 score. Basically, we computed the f1 score from precision and recall formula. The f1 score for hate speech data is 0.68 and for non-hate speech the score is 0.87. We also get support term. From table we find that, for hate speech support is 28 and for non-hate speech the support is 52. And lastly, we get the accuracy from the classification report, which is 0.81, that is 80 percentage.

```
[[49  3]
 [12 16]]
              precision    recall  f1-score   support

           0       0.80      0.94      0.87        52
           1       0.84      0.57      0.68        28

    accuracy                           0.81        80
   macro avg       0.82      0.76      0.77        80
weighted avg       0.82      0.81      0.80        80
```

Figure 4.13: SVM result

We implemented a confusion matrix to evaluate the result we have got from the SVM [support vector classifier]. the figure shows the overview of how accurately the SVM classifier predicted the actual value. According to this confusion matrix, This classifier successfully predicted 47 data as 0 and 20 data as 1. On the other hand, the classifier wrongly 13 values where 6 values as 0 and 7 values as 1. So, the total number of 67 data was successfully predicted, and 13 data were mispredicted.

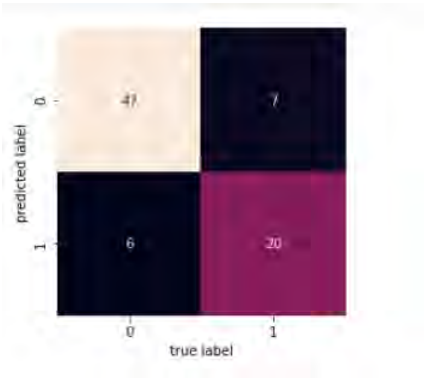


Figure 4.14: The confusion matrix

4.2.4 Final Comparison

Attributes	Accuracy
Deep neural network [DNN]	84
Support vector machine [SVM]	82
Extreme Gradient Boosting [XGBoost]	86

Table 4.2: Comparison between the results

Chapter 5

Conclusion and Future Work

5.1 Conclusion

Social networks and many web channels use features, such as web calling and message delivery, voice-controlled systems. Like all popular websites, video and audio calls are also included. The contact has become a new medium. Just when we use voice commands to work is a matter of time. Those features are used in Facebook, What's app, Skype, Zoom, Google and others all over the world. Since websites and social media sites are anonymous, abusive remarks can easily be left behind and menaced or scammed. On the other hand, online video sharing sites such as YouTube feature a vast number of hateful and inappropriate children's material and comments. These acts trigger issues like fear, assaults, depression , social anxiety and much more. Many systems only use texts yet create confusion when people use offensive words and hateful words as a joke or as a part of friendly expression.

Our system would detect online hate speech such threat, hateful comments, harassments, violated audio massages, online terrorism, racism towards ethnicity and aggressive voices which can turn a human being emotionally harassed, suicidal, revengeful. As an input, we will using real audio voice from human beings. The audio data are then filtered and manipulated in a way to accomplish our objective. For functionality extraction, audio data has been used and separately labeled into hate and non-hate speech. After labeling them, we applied many train-test functionalities and finally After applying machine learning along with Deep learning approaches, we successfully came to verifying the results we found. With the help of modern Deep learning technology, we eventually made the system to decided whether or not it is a hate speech. It is fast and reliable as it uses all functionalities in a sequential and compact way. Furthermore, it can discern more consistency between friendly language and hate speech. Benefits of hateful remarks and menaces can be identified automatically since the technology is configured to operate in any audio and for further research can be used in video contents. We plan to use this framework on social media sites like Facebook, Twitter and YouTube along with online multiplayer game making industry to detect aggressive, intimidating, bullying, and abusive content.

5.2 Future Work

A large number of data is required for the training process, which is among the system 's great limitations. We still have some noise, so we are going to use more filters for processing data. We want to incorporate more details in the future to boost the machine accuracy. The device can be configured to eliminate aggressive or unsafe threats from children and more effectively detect hate speech. So, then, w e can filter videos by tweaking hate material or violent material contents. Moreover, to compare our result and create a more accurate method, we will use some more classification algorithms and develop our system. We can also introduce our system into any kind of online gaming or communicating technology. It will be more efficient if we can detect hate speech from people in real-time.

Bibliography

[1] D. Rybach, C. Gollan, R. Schluter, and H. Ney, "Audio segmentation for speech recognition using segment features", in *2009 IEEE International Conference on Acoustics, Speech and Signal Processing*, IEEE, 2009, pp. 4197–4200.

[2] S. J. Arora and R. P. Singh, "Automatic speech recognition: A review", *International Journal of Computer Applications*, vol. 60, no. 9, 2012.

[3] U. Patil, S. Shirbahadurkar, and A. Paithane, "Automatic speech recognition of isolated words in hindi language using mfcc", in *2016 International Conference on Computing, Analytics and Security Trends (CAST)*, IEEE, 2016, pp. 433–438.

[4] P. Badjatiya, S. Gupta, M. Gupta, and V. Varma, "Deep learning for hate speech detection in tweets", in *Proceedings of the 26th International Conference on World Wide Web Companion*, 2017, pp. 759–760.

[5] L. Gao and R. Huang, "Detecting online hate speech using context aware models", *arXiv preprint arXiv:1710.07395*, 2017.

[6] S. Kamble and A. Joshi, "Hate speech detection from code-mixed hindi-english tweets using deep learning models", *arXiv preprint arXiv:1811.05145*, 2018.

[7] R. Kshirsagar, T. Cukuvac, K. McKeown, and S. McGregor, "Predictive embeddings for hate speech detection on twitter", *arXiv preprint arXiv:1809.10644*, 2018.

[8] E. R. Swedia, A. B. Mutiara, M. Subali, *et al.*, "Deep learning long-short term memory (lstm) for indonesian speech digit recognition using lpc and mfcc feature", in *2018 Third International Conference on Informatics and Computing (ICIC)*, IEEE, 2018, pp. 1–5.

[9] S. Kooshan, H. Fard, and R. M. Toroghi, "Singer identification by vocal parts detection and singer classification using lstm neural networks", in *2019 4th International Conference on Pattern Recognition and Image Analysis (IPRIA)*, IEEE, 2019, pp. 246–250.

[10] S. MacAvaney, H.-R. Yao, E. Yang, K. Russell, N. Goharian, and O. Frieder, "Hate speech detection: Challenges and solutions", *PloS one*, vol. 14, no. 8, e0221152, 2019.