Rumor Detection In Bangla
Using Machine Learning & Deep learning

by

Md Istiaq Ahmed
18101191
Shoumik Malakar Anoy
18101139
Elham Nusrat
18301265
Risat Ahmed
18101713

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

Department of Computer Science and Engineering
Brac University
May 2022

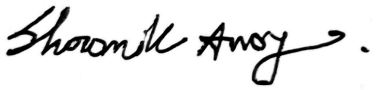
Declaration

It is hereby declared that

1. The thesis submitted is 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:

Md Istiaq Ahmed
18101191



Shoumik Malakar Anoy
18101139

Elham Nusrat
18301265

Risat Ahmed
18101713

Approval

The thesis/project titled "Rumor Detection In Bangla Using Machine Learning & Deep learning" submitted by

1. Md Istiaq Ahmed (18101191)
2. Shoumik Malakar Anoy (18101139)
3. Elham Nusrat (18301265)
4. Risat Ahmed (18101713)

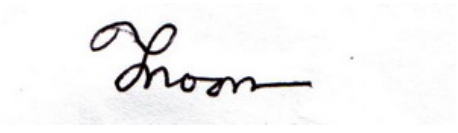
Of Spring, 2022 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of B.Sc. in Computer Science on May 29, 2022.

Examining Committee:

Supervisor:
(Member)

Mr. Annajiat Alim Rasel
Senior Lecturer
Department of Computer Science and Engineering
Brac University

Co-Supervisor:
(Member)



Ms. Jannatun Noor Mukta
Lecturer
Department of Computer Science and Engineering
Brac University

Co-Supervisor:
(Member)

Dr. Farig Sadeque
Assistant Professor
Department of Computer Science and Engineering
Brac University

Head of Department:
(Chair)

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

Abstract

Over the last decade, social media networks have become a vital platform for sharing and obtaining information. Nevertheless, one of the dark sides of this social media fever is rumors and fake news. Because misinformation has caused destruction around the globe, it can also be a powerful weapon in cyber warfare. Due to its easy accessibility in Bangladesh, cybercriminals and malicious forces target social media platforms to spread misinformation in the native Bangla language. However, there is no effective research or efficient tool to detect rumors in the Bangla language on social media. So, this research works on Bangla rumor detection using Machine Learning and Deep learning algorithms. This work explores different types of Machine Learning and Deep Learning techniques and relevant datasets to develop an effective technique that will help detect rumors from trending and sensitive Bangla social media posts, detect their authenticity, and efficiently provide an accurate result.

Keywords: Rumor; Hoax; Misinformation; Fake News; Artificial Intelligence; Machine Learning; Classification Models; Natural Language Processing; CNN; RNN

Acknowledgement

In the beginning, we are extremely grateful to our creator, for whom we are able to complete our thesis work flawlessly. Next, we would like to express our deepest gratitude towards our parents. Without their sacrifices, support and prayer, we would not have been able to come this far. Then, to our respective supervisor Annajiat Alim Rasel for his great mentorship. Additionally, co-supervisors Jannatul Noor Mukta and Dr. Farig Sadeque for their kind support and advice. They were always there whenever we needed them throughout the whole thesis process. This would not have been possible without their help, we are truly grateful for that. Finally, huge thanks to Google for the crucial breakthrough to completing our work.

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

API Application Programming Interface

CNN Convolutional Neural Network

FN False Negative

FP False Positive

GRU Gated Recurrent Unit

KNN k-nearest neighbors algorithm

LDA Linear Discriminant Analysis

$LSTM$ Long short-term memory

$MM - MTL$ Multi Modal Meta Multi Task

NLP Natural Language Processing

RNN Recurrent Neural Network

SMO Sequential Minimal Optimization

SVM Support Vector Machine

TN True Negative

TP True Positive

Chapter 1

Introduction

1.1 Introduction

In Bangladesh, social media plays a vital role in people's lives. With these social networking platforms, people communicate, create bonds, do online shopping and share critical information about their daily lives and the country.

According to a 2021 report, around 48.66 million active social media users [1]. This number is continuously increasing. However, there is a catch as immoral people abuse social media to create chaos by spreading misinformation.

Misinformation can exist in multiple forms, such as fake news, rumors, propaganda, etc. As subsets of misinformation, these are varied by their motivation and deception sources. False news streams faster than actual news. Inconsistency over the wayward and facts checking through social media posts is the major reason for spreading these hybrid threats. The difference between fake news and rumors is that fake news is untrue and deceitful information delivered as news by online newspapers that publish to make a profit. On the other hand, rumors are misinformation spread through random social media users by circulating an unverified or intentionally false story.

Among these forms of misinformation, rumor is one of the most vicious ones. Because rumors are spread in a manner that creates panic among the mass. As a result, rumors can spread like wildfire. Moreover, by the time rumors get detected and removed, they have already created havoc.

Holding on to information and obtaining the right direction in a crisis is essential. We have seen several critical situations regarding social media involvement in sharing information over the past years. Generally, people tend to believe the things that belong to their prejudices, no matter how unexpected they are. The ongoing COVID-19 situation has made this susceptibility very noticeable. A couple of gossips related to covid-19 targeted people through social media, such as the government will pardon rents during lockdowns, the Coronavirus stays in the throat for four days gargling a cure for COVID-19, use vegetable oil to stop this disease from affecting the body, eating garlic can be beneficial to get rid of covid, due to the pandemic tax refunds.

In 2020, rumors were spreading parallelly with Corona's fear. Various misleading information spread on social media about what to do and what not during Covid-19. Comparing that news with actual information provided by the World Health Organization (WHO) to clarify whether the news is a myth or real.

During the Covid-19 situation, many misleading incidents happened for believing rumors and not being cautious about the circumstances. For instance, a hand dryer or ultraviolet light kills CoronaVirus, a thermal scanner can predict Corona positivity, and the Pneumonia vaccine can resist Coronavirus.

Gradually, social media is becoming a reproduction of unverified and groundless information or rumors. These rumors can even threaten people's mental health, are risky economic-wise, and attack a country's solidity. Hence, the World Health Organization (WHO) invented a page on their website called "myth buster" to reduce some of the irresponsible declarations and other fact-checking websites. Fact-checking is a journalistic practice that protests trusted figures from a maiden claim. It requires them to recognize evidence from trustworthy resources, understand the factors, and have rational thinking over the evidence. Until now, without a large dataset generating an automatic machine learning method has been a considerable lack at bottleneck.com for fact-checking. A misleading narrative explanation of an incident generates a cycle of rumors from person to person. The rumors circulate in several ways, such as like, comment and share. Rumors flow very fast over social media. Basically, like, comment, and resharing the origin post results in a diffusion cascade or tree.

Websites such as POLITIFACT, FactCheck, FULL FACT, and FactWatch BD are dedicated to fact-checking. For example, POLITIFACT.com separates the contexts based on validation, mostly false, true, and half true. Accordingly, without an automatic rumor detection system, social media is a cradle for rumors in Bangladesh. Additionally, in 2017 over a social media post, a rumor came out about the anger of the Muslims from the nearest villages against the Hindu community in Rangpur. [2] Moreover, 53 were arrested from the Hindu houses during the mass violence. [3] Once again, in 2021, following another rumor from anti-Islam-related social media status, more than 20 Hindu houses were burned down by a mob in Rangpur. [4] Therefore, similar to POLITIFACT, FactWatch is an IFCN (International Fact-Checking Network) [5], a verified rumor checker website based in Bangladesh. It is a news & media-based website where the fact-checking team rates their facts in seven ways: False, Satire, Missing Context, Partly False, Altered, True, and Unverified [6].

A source shows that normally individuals learn to stop distributing a rumor if it is proved to be incorrect. Post acceptance in social media platforms has a short-term lifeline that typically stays for a few days or a few weeks, called the explosive increase phase. So, on social media, the primary concern is to detect a rumor before circulating the cycle so that everyone gets aware of that before time. It is troublesome to integrate early-stage rumor detection, and the most significant issue is the missing datasets. Searching and looking over comments of a post can solve rumor detection problems where comments can be used to find the correct information

through users' judgment, theory, and evidence to stop publicizing. In addition, Rumors are taking part in the span of computer science and artificial intelligence.

Nevertheless, reactions, comments, and shares are sometimes excessively limited in early-stage rumor detection on social media posts. Contradictory, few intellectual people proposed that they can use the circumstances by the primary processing unit for the prediction, such as the hierarchical structure model. The accuracy and scope of a single processing layer to completely take out the text information cannot be high. Still, it can be beneficial for detection with higher-level structural models. Furthermore, a multi-modal post embedding layer to practice instruction improves engagement to work on textual and visual contents together. On that account, the route of information diffusion can be used to detect rumors.

1.2 Problem Statement

As discussed above, rumors can pose a serious threat to our society. Detecting rumors is a challenging task. Because according to a 2021 Statista [7] report, the total number of social media and Twitter combined users is 3250 million or 3.250 billion. Out of the 3250 million social media users in Bangladesh is 48.66 million. That makes our task even more difficult.

Rumors on social media spread in a way that directly torments people's beliefs and social norms. As a result, people often react violently, which results in social unrest. For instance, in 2021, a piece of misleading information through social media triggered violence in the Hindu community of the religious incident in Comilla [8].

Rumor detection in Bangla can be defined as a binary classification problem. Because the social media posts can either be labelled as rumor and non-rumor or 0 or 1. Currently, there is a few research on the detection of misinformation in the Bangla language. But most of this research worked on fake news detection. On the other hand, our work completely focuses on rumor detection on different social media platforms. So, our approach to this problem is completely unique from theirs. For our research to succeed, we require a human-verified textbase rumor and non-rumor Bangla social media post dataset. However, there is no social media textbase rumor dataset currently available in the Bangla language. So, we took the initiative to create an entire new dataset containing a sound ratio of both Bangla rumor and non-rumor social media posts and comments. Our entire dataset is manually taken and human-verified. As a result, our dataset distinguishes us from others. To solve the rumor detection problem in Bangla, our own benchmarked dataset was used to conduct the analysis.

1.3 Research Objectives

This research aims to implement different types of state-of-the-art machine learning classifiers and deep learning models on a special Bangla textbase dataset and analyze the results to propose better models for detecting rumors. So, the objectives of this research are

1. To deeply understand different Machine learning classifiers and deep learning models.
2. To train different deep learning models and Machine learning classifiers with a textbase large-scale social media post and comment dataset.
3. To test and compare the algorithms and models, develop the best-performing ones at detecting rumors on Bangla on social media.
4. To evaluate the proposed models.
5. To offer recommendations for improving the proposed models.

In the future, this model can be extended to detect videos and images spreading rumors in Bangla.

Chapter 2

Literature Review

Researchers tested and compared different machine learning classifiers such as Naive Bayes, Decision Tree, Logistic Regression, KNN, SVM, etc. Including CNN, LSTM, and RNN, different deep learning models are used to detect rumors. Some of the works are:

2.1 Related works

In the paper, Vohra et al. [10] designed an automated system for rumor detection on social media sites based on the Latent Dirichlet Allocation (LDA) algorithm and web scraping of news. At first, the authors collected data by implementing the Twitter API for a particular hashtag. Then they cleaned the data by dumping the URL, username, and punctuation marks and encoded the data to ASCII for removing emojis and other symbols. Secondly, the authors performed topic modeling on the cleaned data by applying LDA. They also generated a topic containing three keywords. These keywords were represented as their dataset and served as queries for their News website scraping. Lastly, they performed web scraping of four major new websites. After the web scraping process, if any of the news sites returned a single article correlated to that particular topic, that topic will be considered as a non-rumor. Otherwise, the topic will be classified as a possible rumor. This proposed automated model has an accuracy rate of 92%, with 96% precision and 70% recall.

In recent studies, Bingol et al. [11] modeled the fake news detection task in social networks as a classification complication and applied different state-of-the-art classifiers such as Random Forest, OneR, JRip, Naive Bayes, SMO, and ZeroR for solving the problem. Firstly, They applied Basic text classification procedures to the datasets. For the dataset, publicly obtained Twitter data by Kwon et al. [12] was used. They conducted classification operations on their selected classifiers after computing the number of terms and the formation of the document matrix. Lastly, when comparing the results, they found that Random Forest and SMO classifiers provided the highest accuracy. However, in concentration metrics, Naive Bayes, SMO, and Random Forest claimed higher values. However, this experiment found the ZeroR classification algorithm to have the lowest accuracy rate.

Wenfeng et al.[13] 2020 proposed a rumor detection model based on Natural Lan-

guage Processing and Naive Bayes to detect micro-blog hoaxes in their paper. The researcher proposed to process the text with the help of NLP methods and input the processed texts in the Naive Bayes classifier to identify whether the input micro-blog is true or false. As a result, they showed they got an accuracy rate of 0.932 in Naive Bayes-NLP, and its F1 score shows 0.951, and when they compared with DTC, it showed only 0.674 in the F1 score. The proposed model showed it decreased the processing time and showed more accuracy compared to other algorithms.

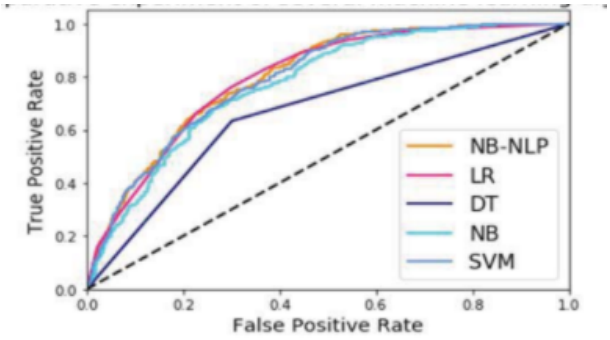


Figure 2.1: ROC curves of several machine learning classifiers [13]

In their research, Kotteti et al. [14] combined an ensemble model with a data analysis method to detect rumors quickly on social networks. Firstly, their data analysis method transformed Twitter data into time-series vectors. On the other hand, their ensemble model used the majority-voting scheme on collective neural network models and took advantage of their individual strengths. Their ensemble model consists of 6 individual neural networks. These neural networks are trained using time-series data. Lastly, for determining the final result as rumor or non-rumor, the predictions of the ensemble model are grouped, and a majority voting scheme was performed on the group with the PHEME [15] dataset.

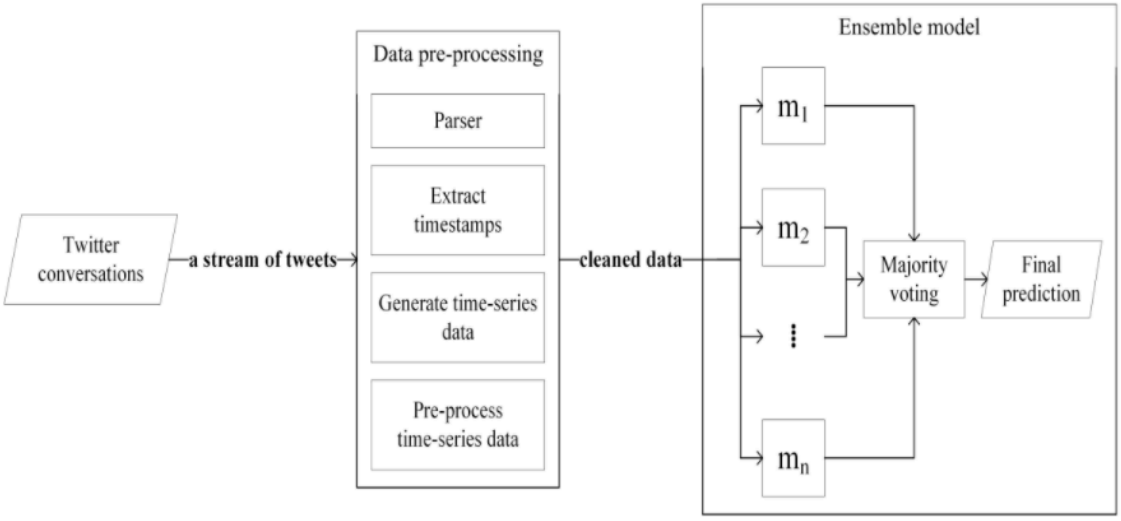


Figure 2.2: Proposed model of Kotteti, et al. [14]

In their study, Yong et al. [16] proposed an emotional classification model rooted

in text and users to identify the rumors on Chinese social platforms (Sina Weibo). Firstly, the researchers constructed an entirely unique emotional classification method that analyzed the emotional adaptation of Weibo replies and also calculated the portion of pessimistic sentimental replies. Secondly, they analyzed the user information, retweet list, and comment list. After that, their model determined if there were authentic users who engaged in spreading rumors. Furthermore, the researchers also used the analyzed characteristics of a user to calculate the reputation value of the user. Lastly, an SVM(Support Vector Machine) classifier was used to detect a microblog post whether it is a rumor or not. Two datasets from the Chinese Weibo Texts of NLP&CC 2014 [17] were used to train the model. For emotional classification and rumor detection, NLP&CC 2014 and "Weibo Misinformation-Declaration Hall" both datasets were used.

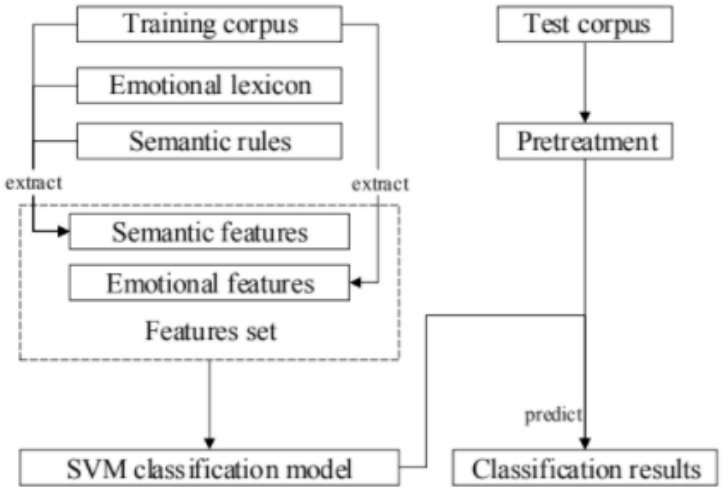


Figure 2.3: Framework of emotional classification method.[16]

MA et al.[18] proposed a model using RNN where they used 3 basic models of RNN, which are TANH, LSTM, and GRU. These models were deeply compared with a few models: SVM-TS, SVM-RBF, DTC, SVM-TS, and RFC. On the other hand, they collected a set from the Sina community management center of common known rumors for Weibo datasets. The final dataset contains 2,313 rumors and 2,351 non-rumors in the paper. The model they used in their test got a higher accuracy rate on the basic RNN model, TANH-RNN, than other models.

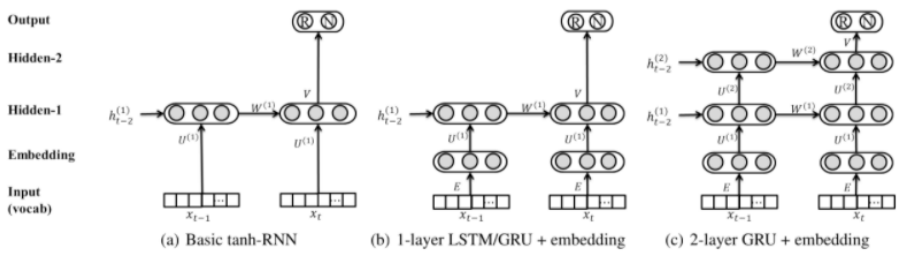


Figure 2.4: Proposed RNN-based rumor detection models of MA et al.[18]

In their work, Guo et al. [19] introduced a rumor detection method based on deep

hierarchical attention-based RNN(LSTM). To make their model more focused on the words with principle understanding, they applied an attention mechanism for building a sentence vector. Then they used the bidirectional LSTM [20] to uproot the background and foreground data individually to attain the features of each word. So, the feature vector of every word also consists of the context data. Their model principally adapted tweet comments as elements to learn and classify. They used two public Twitter datasets known as [21] Twitter15 and Twitterl6 to test the accuracy of their model. Softmax function is used in the output layer for predicting in their model. Moreover, their method paid more attention to both tweets and comments to detect rumors.

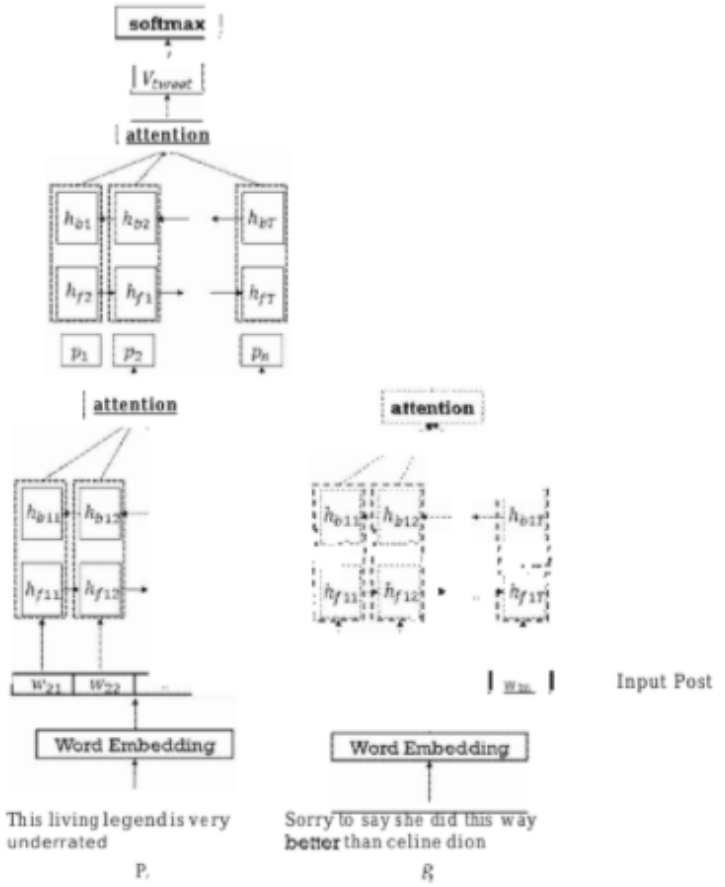


Figure 2.5: Hierarchical Attention Network for Rumor Detection

In this study, Sujana et al. [22] proposed a multi-loss hierarchical BiLSTM model with an attenuation factor inspired by the Hierarchical Model. This method is split into 2 BiLSTM modules: one for post-event and the other for current event-level. Then they proposed a multi-loss BiLSTM, hierarchical structure model, to reduce the training time and increase the training efficiency. Lastly, they added an attenuation factor to the after level, which resulted in reducing the training time and increasing the overall precision concurrently. In this research, they used two PHEME dataset to demonstrate the accuracy rate of their model.

Figure 2.6: The comparison of models proposed by Sujana et al.[22]

In their study, Li et al. [23] proposed a rumor events detection method based on the Deep Bidirectional Gated Recurrent Unit (D-Bi-GRU) which showed that RNN structures containing gate cells like LSTM [19] and GRU are more vigorous than any other structure of RNN which contains only TANH cells. To apprehend the formation of group response information of microblog events, they considered the backward and forward sequence of the microblog flow of group response information with a continuous timeline event. Their model showed in the research that it can provide higher performance and that in microblogs with stacked multi-layers, Bi-GRUs can identify rumors with more efficiency.

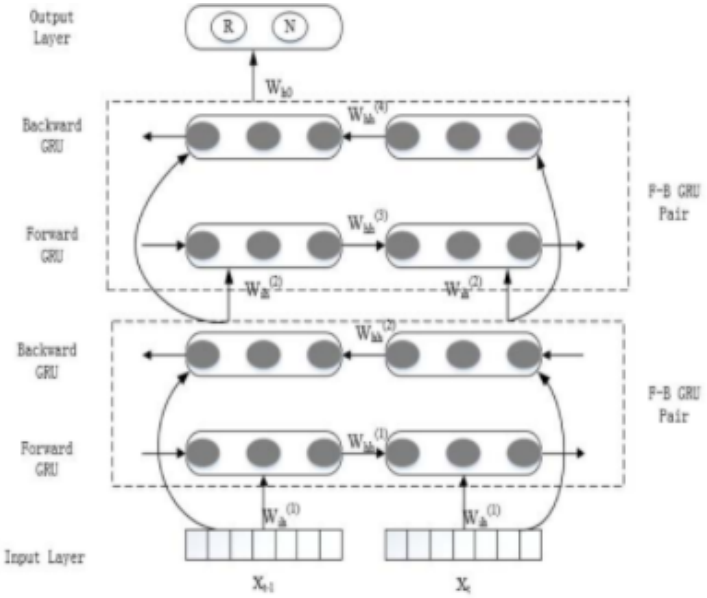


Figure 2.7: Li et al's Rumor detection model based on Bi-GRU

Zhou et al. [24] proposed a rumor detection using a combination of convolutional neural networks and gated recurrent unit networks, C-GRU. Social networks contain tons of information, and feature classification is not good enough.CNN is used to pull out high-level features of the rumors and GRU to extract the sequence features, and they combined these two models with increasing the performance of rumors detection. They used the dataset which was constructed by Ma [9] et al. Compared

to other models, C-GRU gives a better accuracy rate because it can take advantage of the CNN and RNN with the help of hidden layers.

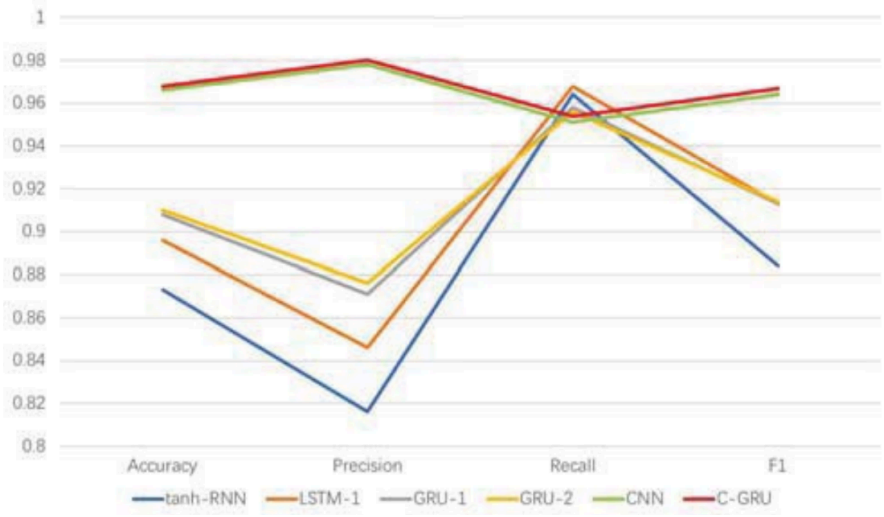


Figure 2.8: Precision Chart for rumors detection models proposed by Zhou et al.[24]

Zhang et al. [25] designed an MM-MTL model for detecting misinformation. They created a post-embedding multi-modal layer that takes both text and image. Basically, BERT [26] is used for text embedding, while VGG19 [27] is used to create the image embedding. In their work, MM-MTL captures the meta-data from rumor detection and stance tasks. The task-specific model may get a multi-modal representation with high accuracy of each post with the help of meta-knowledge. Their work applied a multi-modal meta multi-task learning attention mechanism to fully use the stance information of user responses. The proposed MM-MTL framework's performance can be improved even more with weighted user responses. They used two public datasets, RumourEval [28] and PHEME [29], for rumor identification tasks on Twitter to show that their approach is more efficient and effective.

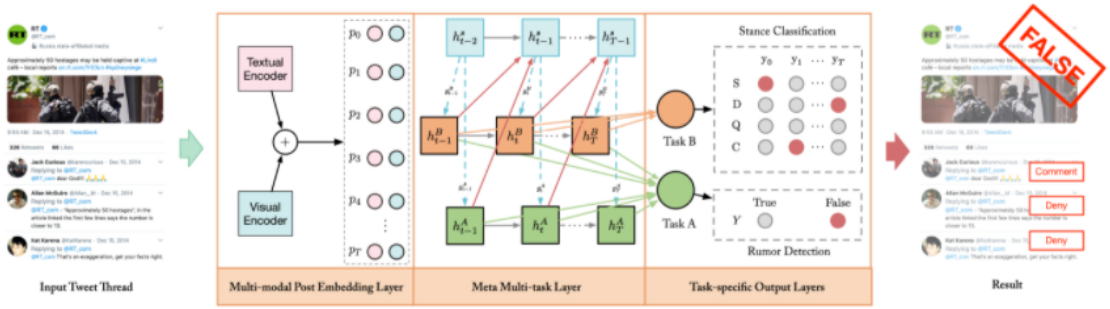


Figure 2.9: Visual representation of MM-MTL framework proposed by Zhang et al. [25]

In their research, Hossain et al.[30] have worked on various models like SVM, Random Forest and Logistic Regression with linguistic features. To focus on language and linguistic features, they have built a dataset of 242 different categories of content, including entertainment and lifestyle. Moreover, this dataset also includes

some clickbait ads-based news. Based on their dataset BANFAKENEWS, the Random Forest model shows higher accuracy than other models like SVM and LR in the case of news embedding. In some other cases, SVM shows more accuracy with some linguistic features. The models have also been compared with the 8.5k human-verified dataset, showing almost the same accuracy rate.

Hussain et al.[31] 2020 proposed a model with supervised vector machines and Multinomial Naive Bayes with around 2500 articles to identify the fake news. In the experiment, they knew these two models almost showed more accuracy than other modalities when they saw the results. As a dataset, they have collected a few articles from online newspapers, and they were limited as it was to the native language, Bangla. After the experiment with two models, Hussain et al. claim that SVM shows more accuracy than MNB in identifying the false news, which was actually false, as well as in detecting the real news.

George et al.[32] 2021 proposed an idea to create a model, where CNN is utilized for deep feature extraction, and LSTM is used for detection with the extracted feature. They have used 8.5k text data and trained the model using Tensorflow 2.0.1. The researchers have collected various data from online news portals and detected the original news and verified the data on many news authentic websites. After training 80% of the data in the model, it showed nearly 76% of accuracy in detecting the original and fake news.

Chapter 3

Dataset Description

3.1 Dataset Analysis

From our research, we found no particular dataset containing only Bangla rumors. There is one dataset, BanFake. However, BanFake contains Bangla fake news, not rumors, and fake news does not go with our methodology. So, we have taken the initiative to create an entirely new dataset for our research.

3.2 Data Collection

Data collection was one of the toughest tasks for us till now. Because most of the rumors of Bangladesh are spread through various social media networks and forums such as social media, Twitter, Reddit, and Quora. Furthermore, the authorities take these rumors down mostly within 10-12 hours.

So, we have taken the help of all Bangla fact-checker websites such as jachai.org, bdfactcheck.com, and rumorscanner.com. From fact-checker websites, we have collected the social media links of the rumors from fact-checker websites and put them in our selenium automation algorithm to scrape data from the links.

But errors in data scraped from the selenium automation algorithm were so much. So, we decided to manually collect Bangla post data and relevant comments from the rumor posts.

However, different social media platforms have different domains like Facebook, Twitter has like, comment, share and quora, and Reddit has upvote, comment. So, we decided to collect the common parameters: the rumor post, date, and comments.

Post Text	থানকুনি পাতা বিসমিল্লাহ বলে তিনবার চিবিয়ে খেলে করোনা ভাইরাস সেরে যাবে।
Post Time	18 March, 2020

Post comments	{
	আলহামদুলিল্লাহ,
	এসব হুজুরাই দেশটা ধংশ করবে,
	এখন থানকুনি পাতা খুঁজতে গিয়ে মানুষ করোনা আরো বাড়াবে,
	অশিক্ষিত বর্বর আপনাকে গুজব ছড়ানোর জন্য পুলিশে দেয়া উচিৎ
	}

Label	0(Rumor)

Table 3.1: Example of a raw data

Until now, we managed to collect 2237 raw data containing the post text along with their public comments. For every post, we have collected a minimum of 3 relevant comments and a maximum of 10 relevant comments.

Label	Post Count	Comment Count
Non- Rumor	1637	8581
Rumor	600	2696
Total	2237	11277

Table 3.2: Total number of rumor and non rumor data

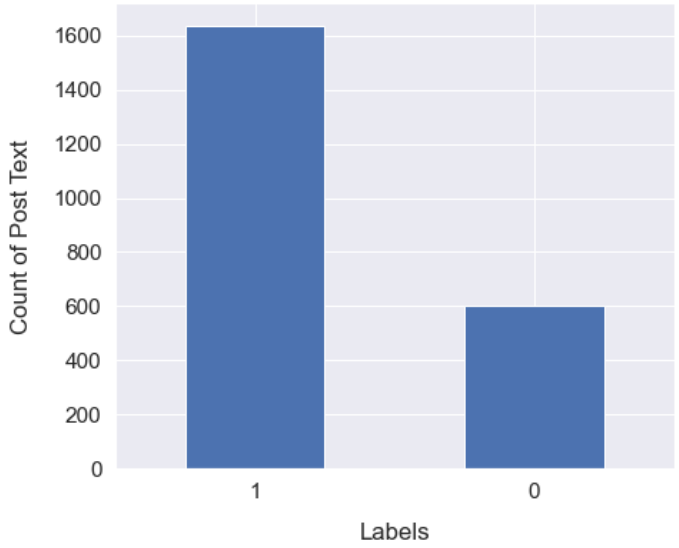


Figure 3.1: Total number of rumor and non rumor data

We had faced many obstacles while collecting the data and creating our dataset. One of the obstacles was according to the new social media policies if a rumor post is reported it gets removed within a few hours. Moreover, if the post gets deleted there is no way we can retrieve the comments of the posts and our models can not work without comments. As a result, the ratio of rumor and non-rumor data is imbalanced.

After collecting the raw data, we have checked all our data manually. Because there is a chance of getting the wrong raw data and sometimes the selenium automation tool fetched data with errors. By the below method we have determined the data labels manually.

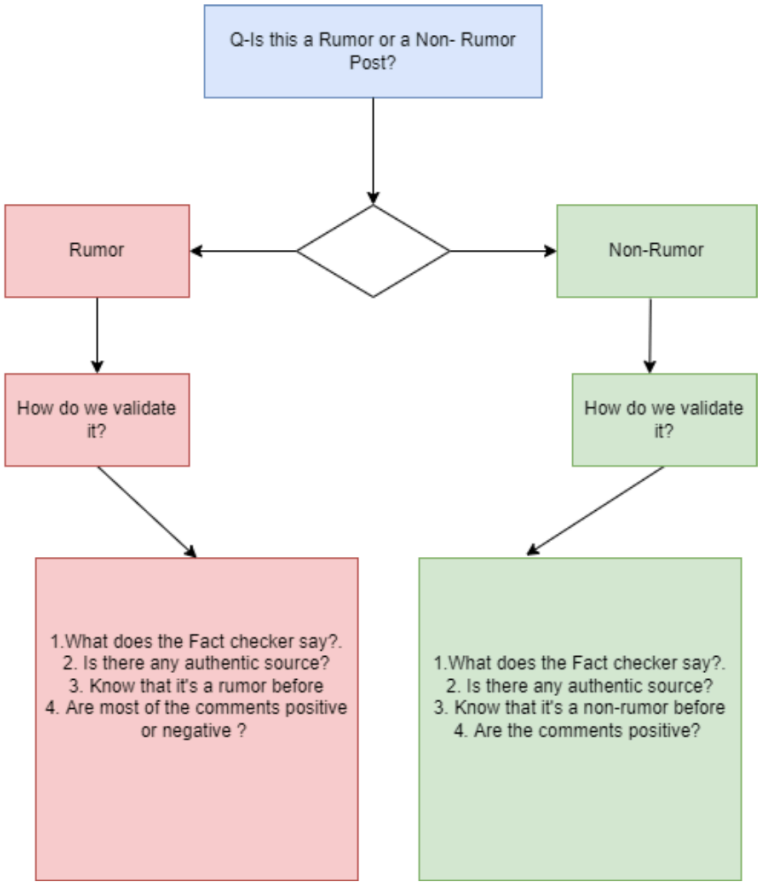


Figure 3.2: Human Validation process of the Dataset

In the future, we plan to make a multimodal dataset containing images, videos, demographics, and audios.

3.2.1 Some Examples of Rumor Post

Figure 3.3: A sample of Facebook rumor post

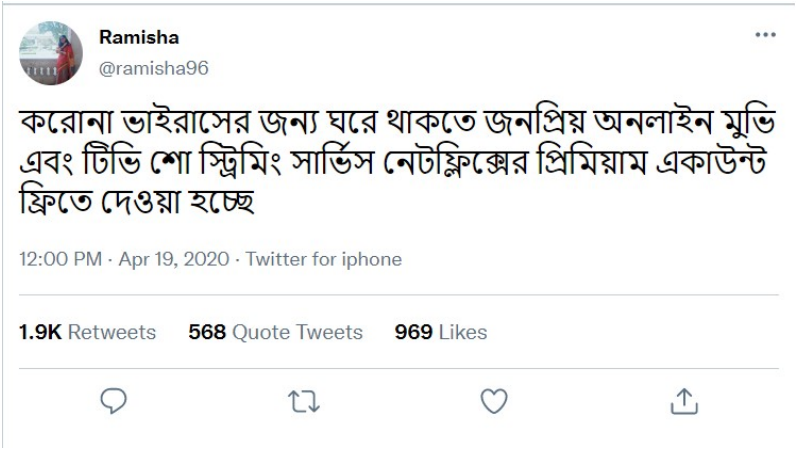


Figure 3.4: A sample of twitter rumor post

3.3 Data Preprocessing

The data preprocessing approach is a data mining procedure that converts unstructured data into a structured shape that is more consistent, usable, and more organized in the machine learning models. There are several advantages like combining all the data from multiple sources, filling in missing data, and pointing out to solve any mismatch to produce better results.
Our data preprocessing method contained 4 steps

1. Remove duplication text
2. Removal of unintelligible words, characters, emojis and punctuation
3. Removal of stop words
4. Stemming

Before the preprocessing task, our textual data looked like the below figure.

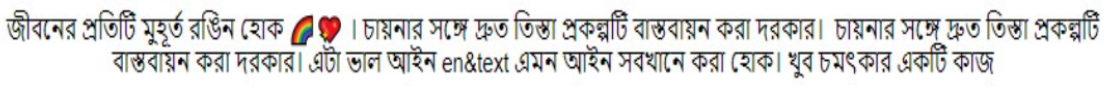


Figure 3.5: Before Preprocessing Textual Data

Firstly, we removed all the duplicate data from our dataset.

Secondly, we extracted all the bad characters, incomplete words, emojis, tags, HTML tags and punctuations from our raw data.

Then, we removed all the stop words from our data. Stop words [34] are common phrases in a language used as prepositions, linkers, quantifiers, and so on. In Bengali, there are stop words such are'আর', 'তাই', 'অথবা', 'অথচ' etc. We gathered a list of Bengali stopwords [35] consisting of 398 stopwords. Then we removed these words from our raw data. This helped our model to improve its accuracy and efficiency. Because by removing them, we reduce the number of unnecessary words, and vocabulary size gets reduced.

Lastly, we used stemming to cut off prefixes and/or endings of words based on common ones. Stemming is used in NLP to reduce inflection from texts. As a result, we have found the base of every word. We have used a stemmer function from BNLTK(Bangla Natural Language Processing Toolkit). [36]

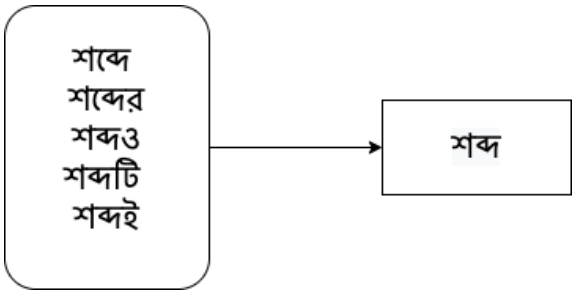


Figure 3.6: Stemming using BNLTK

After preprocessing our textual data looked like below,

জীবন মুহূর্ত রঙিন চায়ন দ্রুত তিস্তা প্রকল্প বাস্তবায়ন দরক চায়ন দ্রুত তিস্তা প্রকল্প বাস্তবায়ন দরকার ভাল আইন সবখানে চমত্

Figure 3.7: After Preprocessing textual data

Chapter 4

Research Methodology

4.1 Work Plan

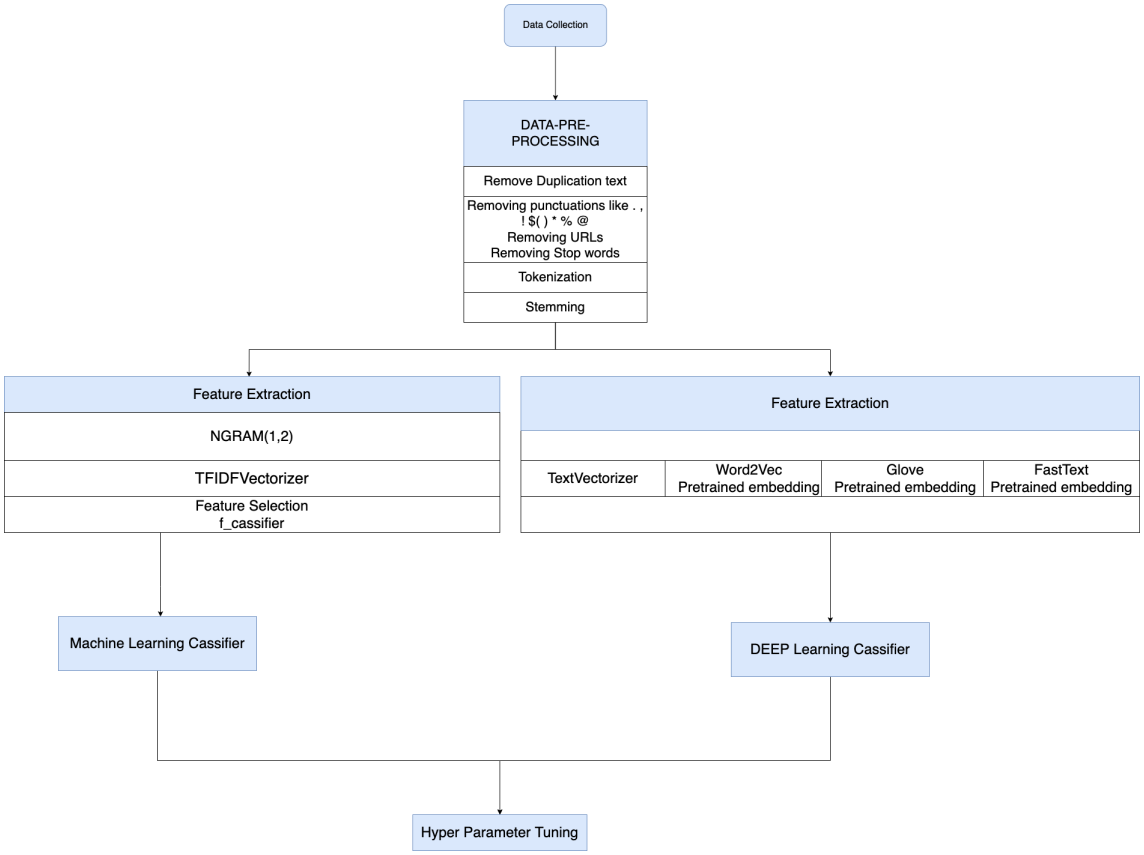


Figure 4.1: Work Plan

4.2 Feature Extraction

Machine Learning algorithms and deep learning models cannot work on the raw text directly. As a result, in order to make our models read the data, we have to transform our data into numerical form. So, Feature Extraction is one of the essential parts for better impact on applying machine learning and deep learning models. We have used several vectors for Feature Extraction.

4.2.1 TF-IDF

TF-IDF calculates the frequency of a word in a document. TF-IDF Vectorizer generates the sparse matrix of numerical features so that the classifiers can understand numerical values. It is a product of TF and IDF. [37]

$$TF(w,d) = \frac{\text{occurences of w in document d}}{\text{total number of words in document d}}$$

Here, TF makes a vocabulary of unique words and calculates the ratio of the word in a text. However, TF fails to find out the importance of the words. That is why the IDF is used. IDF provides the weightage to each word based on its frequency.

$$IDF(w,d) = \ln(\frac{\text{Total number of documents (N) in corpus D}}{\text{number of documents containing w}})$$

4.2.2 Word2Vec

Word2Vec is the procedure of transforming texts into vectors while maintaining semantic correlations and some of their syntactic structure. As frequency-based word embeddings can not represent the semantic value of a text document. [38]

By looking at the context of a word, Word2Vec tries to figure out what it means and how it relates to other words.

For example, "বাংলাদেশ পুলিশ উন্নতি করছে " and "বাংলাদেশ পুলিশ অনেক ভাল", a Word2vec should be able to detect resemblance among them. Here, Word2Vec measures the similarity or distance between two words using cosine distance rather than Euclidean distance.

4.2.3 FastText

FastText is a lightweight and open-source framework that allows users to learn text representations and classifiers. The FastText package uses the skip-gram method to extract features. For example, the word 'পড়ছিলাম' would be treated as পড়, পড়ছি, পড়ছিল, পড়ছিলা, aand so on, here the value of n might range from 1 to the length of the word. This ensures that the dataset's more uncommon or unknown terms do not receive a zero vector representation. [39]

As a result, it makes the model stronger for less common words that contribute more to topic classification.

4.3 Feature selection

The feature selection is a process in which the most relevant, consistent and non-redundant features are chosen using various feature selection algorithms. Therefore, the primary size of the feature vectors is decreased. Then the retrieved and chosen features are delivered to the chosen machine learning models.

In classification applications, the classification algorithms are given qualities from the primary dataset as input. Here, the number of functions available in various

applications is still limited. On the other hand, the number of qualities in some cases may be excessive. Each signal's extracted attributes are kept in a matrix. The accuracy with derived features representing signals and classes associated with the signal is critical. High-level qualities can better discriminate across classes and are therefore more essential in terms of performance than others [40]. Furthermore, the number of transactions is minimized and performance is improved with the help of a few high-level attributes. Because noisy attributes are removed.

When we transformed every texts in our dataset into word both unigram and bigram tokens, we took five thousands of tokens from the unigram and bigram tokenize. Every features/tokens doesn't help in label prediction. Therefore, after terminating specific tokens we can include only the most useful tokens. Various functions which are statistical they accept features and labels that return the feature importance score, such as f_classif [41] and chi2 [42]. Our execution shows the output that these functions are effective in equal.

Chapter 5

Model Specification

Rumor detection in social media posts is a binary classification problem where the post is either Rumor (0) or Non-Rumor (1). We have approached both machine learning methods and Deep Learning models to solve this binary classification problem.

5.1 Machine Learning Methods

As rumor detection is a Binary classification problem and from our initial research we have found the below models to perform best at solving binary classification problems.

5.1.1 Logistic Regression

Logistics regression is widely used for binary classification problems. The logistic regression classifier demonstrates a vector of variables and computes the weights for every input variable, and predicts the post's label. Logistic regression can interpret the relationship between multiple predictor or independent variables and features into a binary explicit target variable marked as "1" and "0". [43]

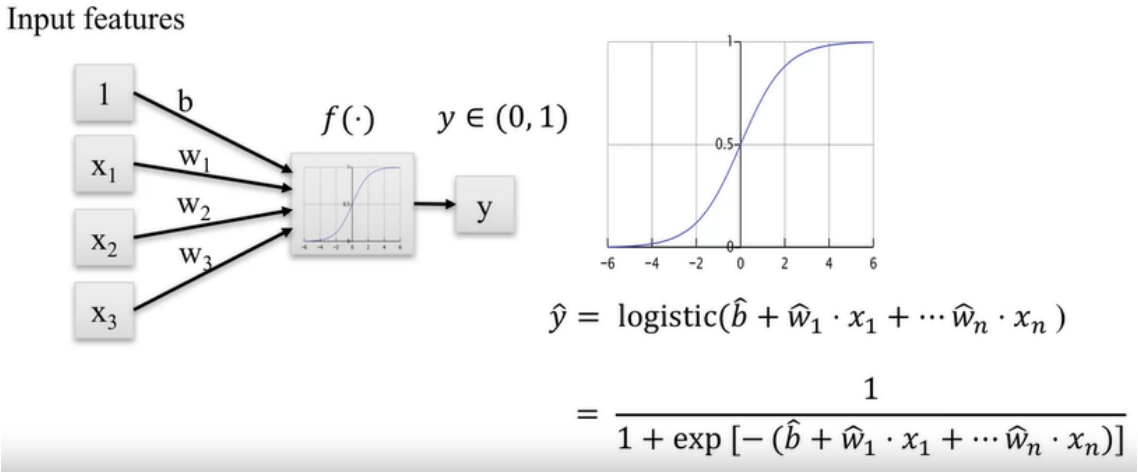


Figure 5.1: Logistic Regression Formula & graph

In Logistics Regression, there is no linear relationship between variables. Because the relient variable has to be a binary variable, and the independent variable must

be linearly connected, not normally distributed, and have the same variance within a group. The groupings must mutually exclude one another.

5.1.2 Support Vector Machines

SVM is a machine learning classifier based on statistical learning theory, which represents the classification and regression approach by using kernel functions to categorize data sets into linear or nonlinear. [44]

This method has gained popularity due to its strong theoretical foundation, ability to operate on enormous data sets, the flexibility provided by kernel functions, and high accuracy. The approach locates the biggest margin among the considerable alternative linear functions to organize separate data linearly. SVM maps to the higher-dimensional space by utilizing kernel functions and discovers multiple planes with the largest margin if data cannot be categorized as linear data.

5.1.3 Naive Bayes

Naive Bayes classifier is a popular technique to decode classification problems. The naive Bayes approach is used to classify the data. Based on equations and calculations, this conditional model was established on the Bayes theorem. Naive Bayes considers independence from the dependent variables. [45]

$$P(c|x) = \frac{P(x|c)P(c)}{P(x)}$$
$$P(c|X) = P(x_1|c)xP(x_2|c)x...xP(x_n|c)xP(x)$$

Here, P= Posterior Probability, $P(x|c)$ = Likelihood, $P(c)$ = Class Prior Probability, $P(x)$ = Predictor Prior Probability.

5.1.4 KNN

In the Vector Space Model(VSM), KNN is one of the best classification algorithms. It has the qualities of being both simple and effective.[46] It's a non-parametric, instance-based text classifier that employs document similarity as a criterion for classification. The majority class is determined by computing the similarity between the test document and its k closest neighbors. However, because it must calculate the distances between each test sample and all training samples, KNN has the disadvantage of having a high temporal complexity. Furthermore, when the data distribution is not balanced, its classification stability is low since it only considers the first k nearest neighbors, neglecting the effect of distances.

5.1.5 Random Forest

The American scientist Leo Breiman proposed an integrated learning model, Random Forest [47]. Random forest is a suitable ensemble method in the case of Binary classification problems. Because randomization ensures that different features such as sparse and dense will be used as primary decision nodes in individual trees. Not but the least, RF/decision trees may be a great approach for inspecting features and their structure well.

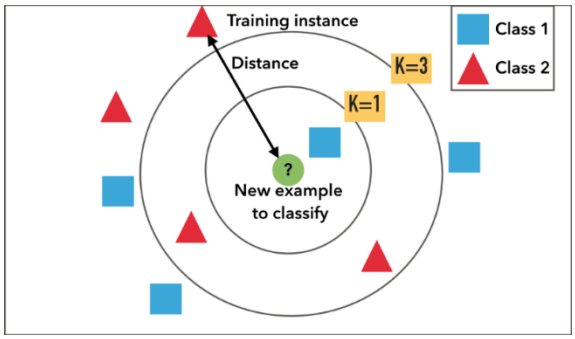


Figure 5.2: KNN classification principle

Furthermore, the random forest creates a multiclass classification model to categorize the dataset which provides a fair accuracy.

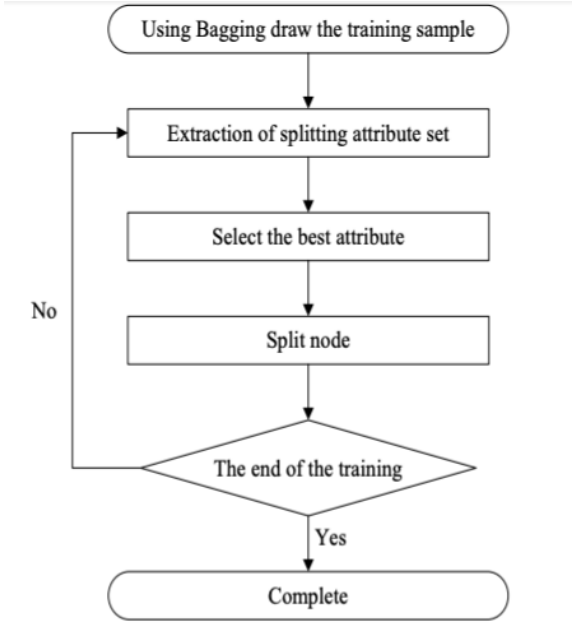


Figure 5.3: Random Forest process

5.2 Deep Learning Models

From our initial research we have found the below deep learning models to perform best at solving binary classification problems.

5.2.1 CNN

The Convolutional Neural Network is a type of neural network in which the output of each layer is used as the input of the following layer of the neuron. [48] The results of each layer are transformed by nonlinear until the output layer using the multilayer convolution process. We used two input layers along with two TextVectorizers and two embedding layers. Then they are put into a concatenated layer and after

that a dropout was performed before putting them into a conv1D layer. After that spatial dropout, global average pooling, and dropout, dense and dropout is performed sequentially. Lastly, the dense layer shows the result.

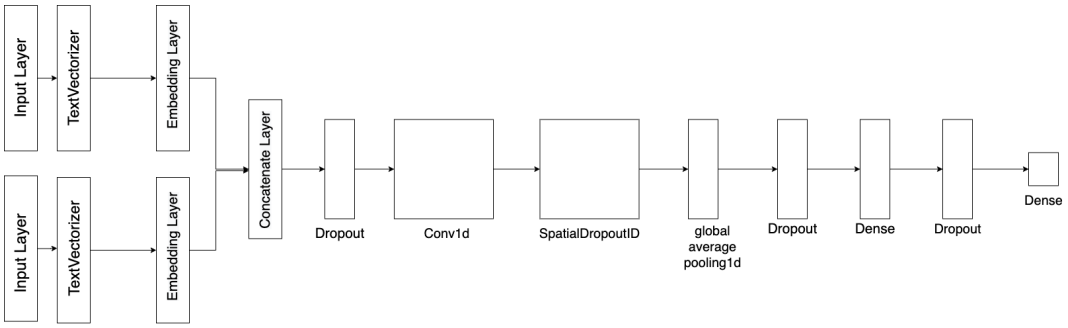


Figure 5.4: CNN diagram

5.2.2 Two Stacked BI-LSTM

A bidirectional LSTM consists of two LSTMs: one that takes the input forward and another that takes it backward. Bidirectional LSTM is a sequential processing model; its units incorporate the past and future context data and learn long-term dependencies without redundant context information [49]. We have used a version of bidirectional LSTM named as Two Stacked BI-LSTM.
The Bi-LSTM is commonly used for text categorization and performs well for sequential modeling issues.

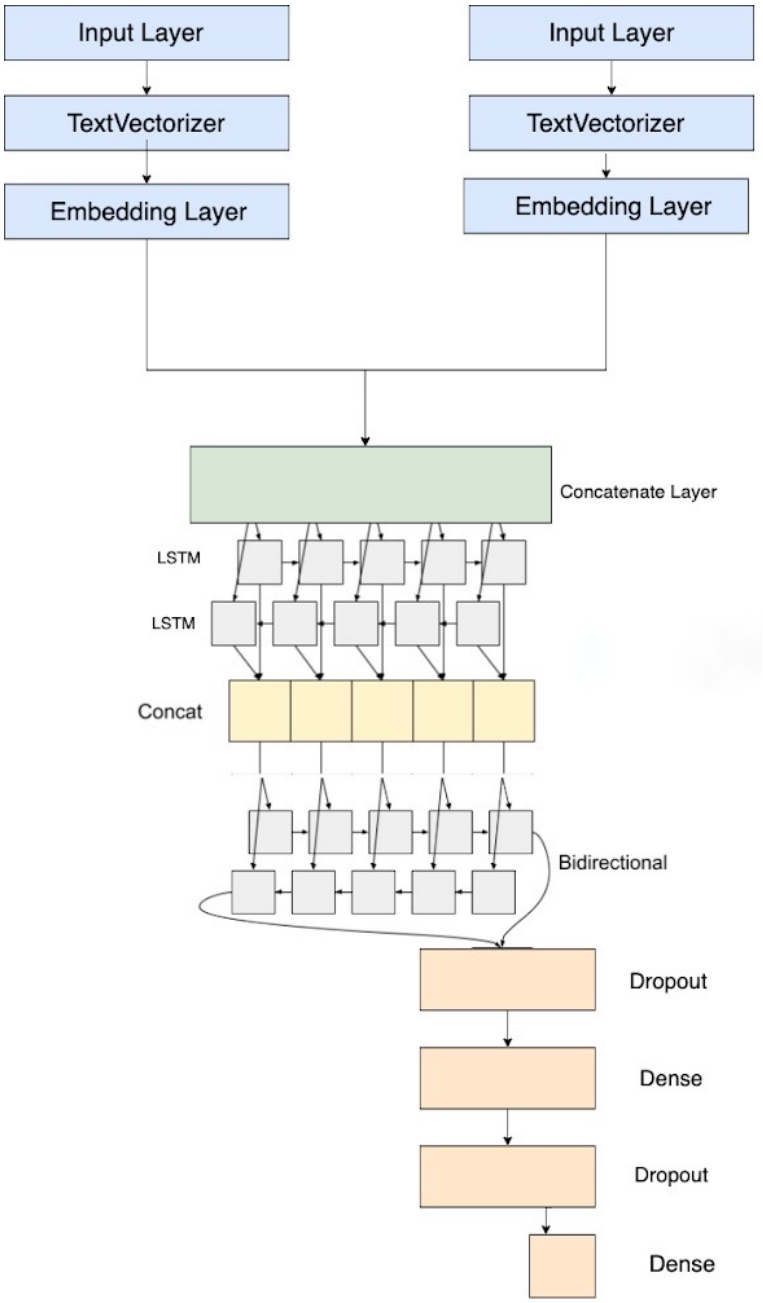


Figure 5.5: Two stacked Bi-LSTM Model diagram

5.2.3 CNN-Bidirectional GRU

We have used our own tuned CNN-Bidirectional GRU. Firstly, We used two input layers along with two TextVectorizers and two embedding layers. Then they are put into a concatenated layer and after that a dropout was performed before putting them into a conv1D layer.After that spatial dropout, max pooling. Then they are put into a Bidirectional GRU layer. Lastly, we have added a GlobalMaxpooling and 2 dense layers.

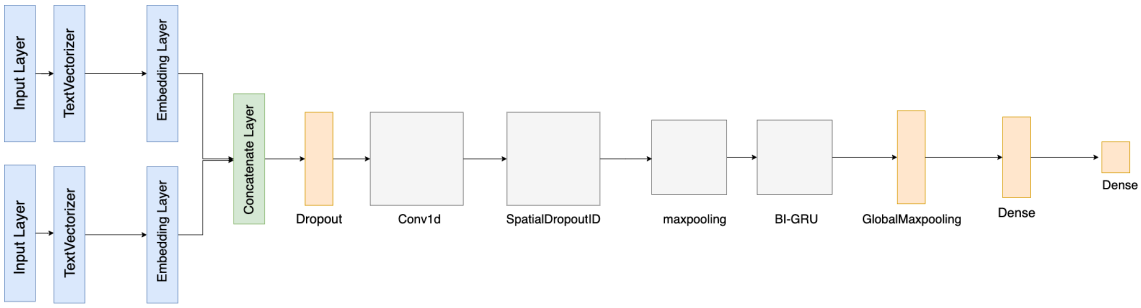


Figure 5.6: CNN-BiGRU Model diagram

5.2.4 Hyper Parameter Tuning

To acquire optimal performance in the models, hyper-parameter learning rate and batch size are needed. Because parameters can significantly impact learning algorithm performance, their values should be carefully tuned manually or automatically.[50]

Moreover, the hyper-parameter can cause the models to not perform at their best capabilities without tuning. The hyper-parameters are performed using a random search to find candidates for learning rate and batch size, along with an experiment on the candidates.

5.2.5 K-Fold

K-Fold is a Cross-validation process in which a resampling technique is used for assessing machine learning models. In K-Fold, the single parameter k indicates the number of divisions into which the given data sample should be divided. [51]

To prevent data leaks, the data is separated into three sets to prevent data leaks: training, validating, and testing. As a result, the testing set should only be converted after the train and validation collections have been utilized to match the model. The prototype is assessed using the test data each time if it fits the train data, and the average of the evaluation scores is used to analyze the overall model.

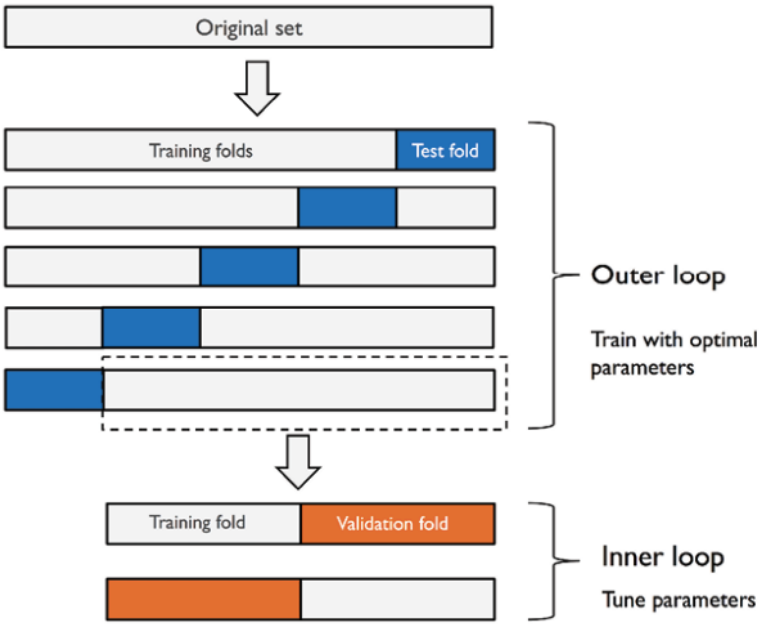


Figure 5.7: K-Fold Cross-validation process

Chapter 6

Experiment Setup

6.1 Creating The Model

We have tried several combinations of word embedding techniques such as TF-IDF Vectorizer, CountVectorizer, Word2Vec, FastText and Glove. Moreover, We trained our train data with a fine tuning hyper parameter.

6.1.1 Machine learning Approach

First of all, we have tokenized post text and comment texts into word unigram and bigrams in machine learning models. After that, we picked the most important 5000 features by vectorizing our data using tfidf vectorizer. Next, we used f_classif to calculate feature importance.

On the other hand, we also used pre-trained Bangla Word2Vec [52], Glove [53], and FastText [54] word embedding and used their mean value as our vector for post and text data. Then, we used gridsearch to run all the different parameters, which fed into the parameter grid to discover tuning hyperparameters. GridSearch has a parameter that performs cross-validation. It splits this train data into train and test to extract the hyper-parameters passed to it. Lastly, it fits the model on the whole train data with the best-found parameters.

6.1.2 Deep learning Approach

We have taken two input layers as we have two features in our data. We have used TextVectorization to vectorize the sequence models. The number of VOCAB sizes is a unique word for each features. An embedding layer is frequently used as the initial layer in sequence models. Throughout the training procedure converting word index sequences into word embedding vectors. We also used pretrained Bangla Word2Vec, FastText and glove word embedding weights in our embedding layer. We have used CNN, two stack Bi-directional LSTM and some hybrid deep learning models. Then we found out whether the models are overfitted or not. If a model is overfitted, we tried to minimize the overfit by reducing the number of hidden layers, which minimizes the complexity of the neuron of the network. Then for the large weights, we involved regularization, which comes down by adding a cost to the loss function.

Lastly, we used dropout layers which removed particular features by setting them to zero.

6.2 Training The Models

For the machine learning approach, we have split our dataset into two parts as we have insufficient data. So we divided our data 70% for training and 30% for testing with a random state of 92. We trained our train data with a fine tuning hyper parameter.

On the other hand, for deep learning, we divided our data 70% for training, 15% for validation and 15% for testing. As it is a binary classification problem, we compiled our models with optimizer adam and loss as binary_crossentropy We set the epoch as 100. Because the initial epoch number produces consistent accuracy. We used the EarlyStopping callback [55] in the fit parameter to stop training when a monitored metric has stopped improving.

6.2.1 Input Layer

In deep learning models, the input layer receives the text data for neurons. Then our preprocessed data is fed to the learning module of the TensorFlow library for generating word vectors. After that, the word vectors of the text are loaded into the network layer as features.

Nevertheless, we have taken both the post text and the comments of the Bangla rumor post in our dataset. So, for our model, we have two input layers. One is for the post text, and another is for comment text.

6.2.2 Embedding Layer

The Embedding Layer is a feasible way to increase model efficiency in neural networks. An embedding layer works as a matrix or lookup table in which the word vectors are organized in order from top to bottom, corresponding to the words in the sentence. For example, if a sentence has n number of words, the dimension of the word vector is the Embedding Layer matrix would be n*m.

6.2.3 Dropout Layer

As we have limited data and multiple nonlinear hidden layers. So the models can get complex. With the rising complexity of the model, it starts to learn the noise present in the training dataset. As a result, there is a chance of model overfitting. [56]

To avoid model overfit we have used the dropout technique. Here, the Dropout layer randomly drops units along with all the incoming and outgoing connections associated with those units in the training time.

6.2.4 SpatialDropout

SpatialDropout is one kind of dropout layer that works the same as the dropout function. But instead of dropping individual elements, it drops the entire 1D feature map. A regular dropout does not perform regularization when the consecutive frames of the feature maps are substantially associated. As a result, the learning rate will drop. In these kinds of circumstances, SpatialDropout performs the best as it endorses feature map independence.

If there is a shape(x), $[p, q, r]$ and $noise_shape = [p, q, r]$ and it will drop the entire $1 - D$ feature map of a matrix.

6.2.5 GlobalAveragePooling

Global Average Pooling is a pooling method specially planned to replace fully linked layers of convolutional neural networks. The goal of Global Average Pooling is to create one feature map for each matching category of the classification task in the last mlpconv layer. Here, the average of each feature map is taken and fed to the resulting vector into the dense layer. As a result, it avoids building completely connected layers on top of the existing feature map.

6.3 CNN

```
Model: "model_141"
_________________________________________________________________________________________________
 Layer (type)                Output Shape              Param #     Connected to
=================================================================================================
 input_283 (InputLayer)      [(None, None)]            0           []

 input_284 (InputLayer)      [(None, None)]            0           []

 embedding_24 (Embedding)    (None, None, 100)         1000000     ['input_283[0][0]']

 embedding_25 (Embedding)    (None, None, 100)         884800      ['input_284[0][0]']

 concatenate_29 (Concatenate)  (None, None, 100)       0           ['embedding_24[3][0]',
                                                                    'embedding_25[3][0]']

 dropout_63 (Dropout)        (None, None, 100)         0           ['concatenate_29[0][0]']

 conv1d_29 (Conv1D)          (None, None, 128)         64128       ['dropout_63[0][0]']

 spatial_dropout1d_29 (SpatialD  (None, None, 128)     0           ['conv1d_29[0][0]']
 ropout1D)

 global_average_pooling1d_29 (G  (None, 128)           0           ['spatial_dropout1d_29[0][0]']
 lobalAveragePooling1D)

 dropout_64 (Dropout)        (None, 128)               0           ['global_average_pooling1d_29[0][
                                                                    0]']

 dense_58 (Dense)            (None, 128)               16512       ['dropout_64[0][0]']

 dropout_65 (Dropout)        (None, 128)               0           ['dense_58[0][0]']

 dense_59 (Dense)            (None, 1)                 129         ['dropout_65[0][0]']

=================================================================================================
Total params: 1,965,569
Trainable params: 1,965,569
Non-trainable params: 0
```

Figure 6.1: CNN Model Summary

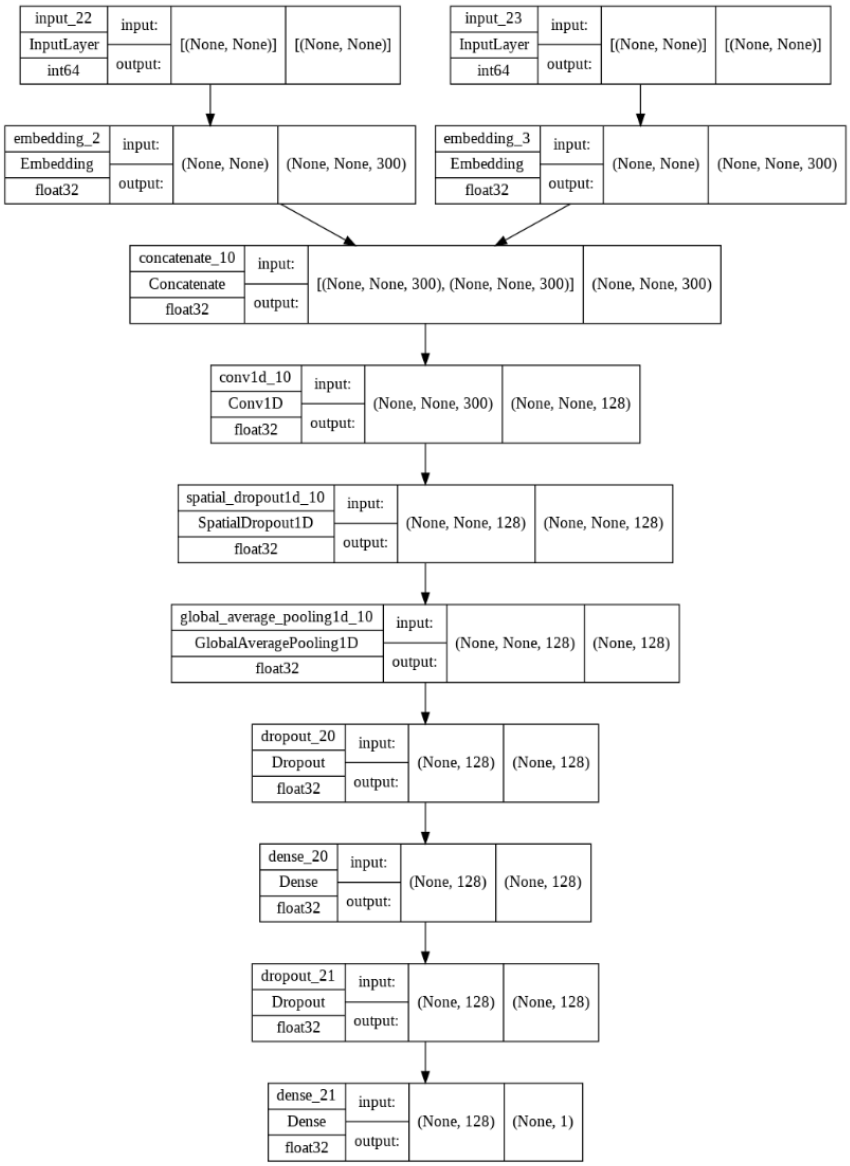
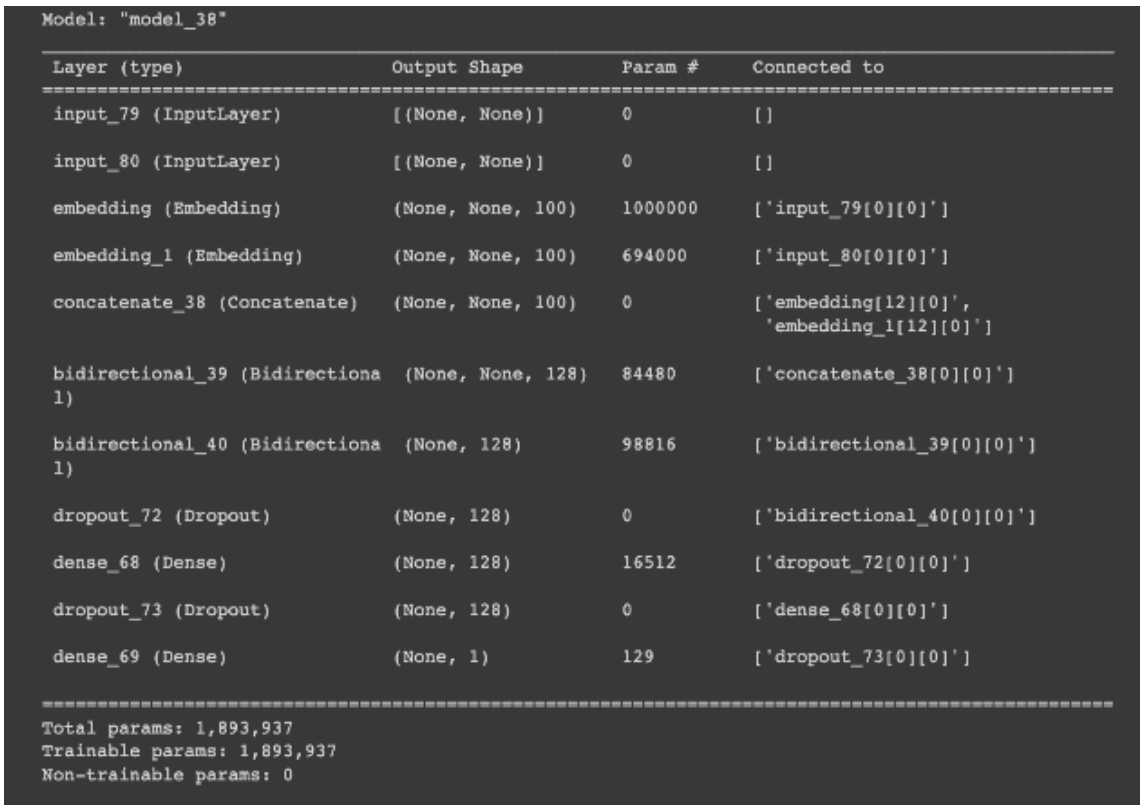


Figure 6.2: CNN Model Architecture

6.4 Two Stacked Bi-LSTM



```
Model: "model_38"
_________________________________________________________________________________________________
 Layer (type)                   Output Shape         Param #     Connected to
=================================================================================================
 input_79 (InputLayer)          [(None, None)]       0           []

 input_80 (InputLayer)          [(None, None)]       0           []

 embedding (Embedding)          (None, None, 100)    1000000     ['input_79[0][0]']

 embedding_1 (Embedding)        (None, None, 100)    694000      ['input_80[0][0]']

 concatenate_38 (Concatenate)   (None, None, 100)    0           ['embedding[12][0]',
                                                                  'embedding_1[12][0]']

 bidirectional_39 (Bidirectiona  (None, None, 128)   84480       ['concatenate_38[0][0]']
 l)

 bidirectional_40 (Bidirectiona  (None, 128)         98816       ['bidirectional_39[0][0]']
 l)

 dropout_72 (Dropout)           (None, 128)          0           ['bidirectional_40[0][0]']

 dense_68 (Dense)               (None, 128)          16512       ['dropout_72[0][0]']

 dropout_73 (Dropout)           (None, 128)          0           ['dense_68[0][0]']

 dense_69 (Dense)               (None, 1)            129         ['dropout_73[0][0]']

=================================================================================================
Total params: 1,893,937
Trainable params: 1,893,937
Non-trainable params: 0
```

Figure 6.3: Two stacked Bi-LSTM Model SUMMARY

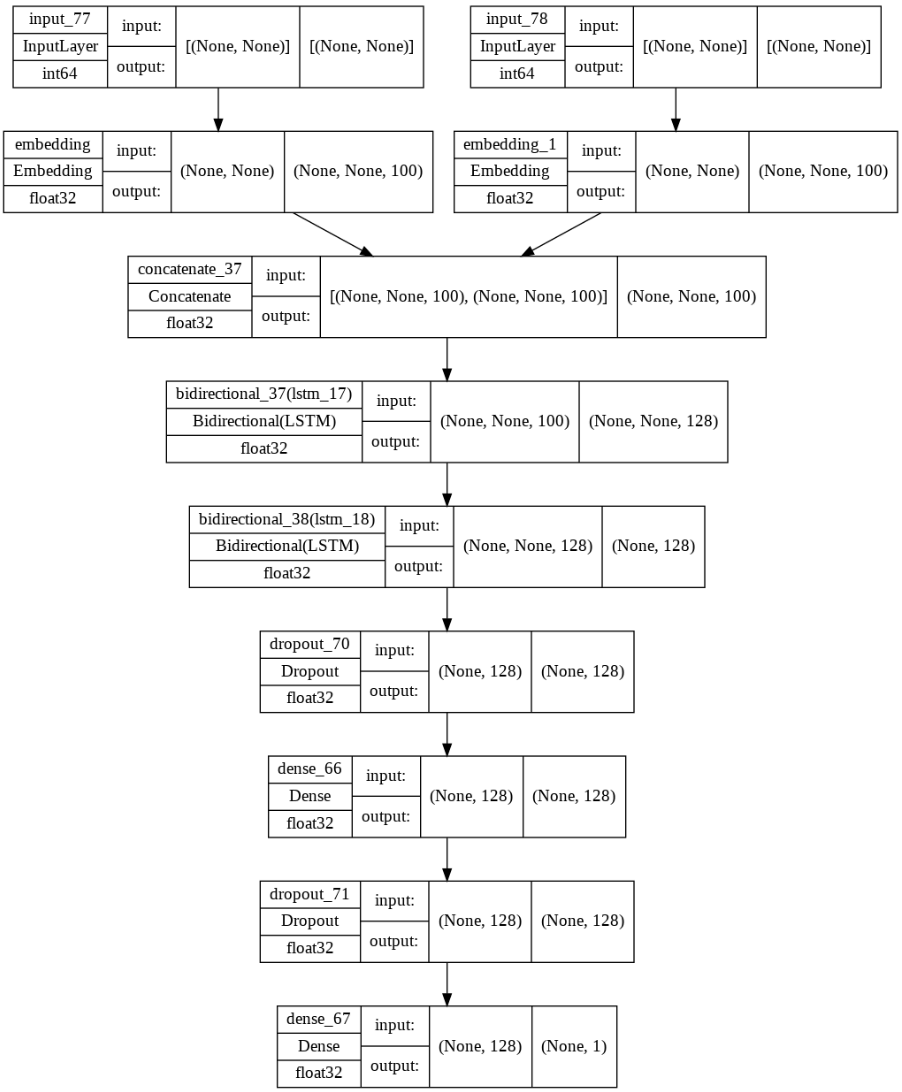
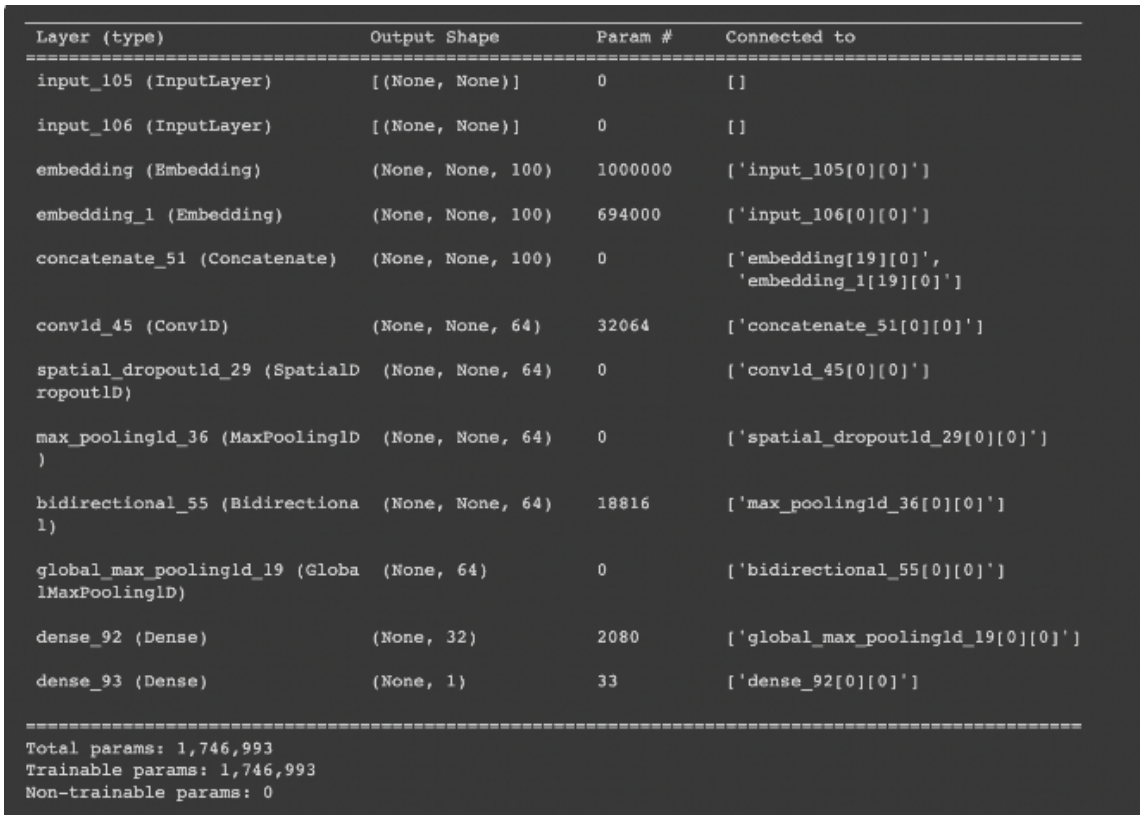


Figure 6.4: Two stacked Bi-LSTM Model Architecture

6.4.1 CNN-BiGRU



Layer (type)	Output Shape	Param #	Connected to
input_105 (InputLayer)	[(None, None)]	0	[]
input_106 (InputLayer)	[(None, None)]	0	[]
embedding (Embedding)	(None, None, 100)	1000000	['input_105[0][0]']
embedding_1 (Embedding)	(None, None, 100)	694000	['input_106[0][0]']
concatenate_51 (Concatenate)	(None, None, 100)	0	['embedding[19][0]', 'embedding_1[19][0]']
conv1d_45 (Conv1D)	(None, None, 64)	32064	['concatenate_51[0][0]']
spatial_dropout1d_29 (SpatialDropout1D)	(None, None, 64)	0	['conv1d_45[0][0]']
max_pooling1d_36 (MaxPooling1D)	(None, None, 64)	0	['spatial_dropout1d_29[0][0]']
bidirectional_55 (Bidirectional)	(None, None, 64)	18816	['max_pooling1d_36[0][0]']
global_max_pooling1d_19 (GlobalMaxPooling1D)	(None, 64)	0	['bidirectional_55[0][0]']
dense_92 (Dense)	(None, 32)	2080	['global_max_pooling1d_19[0][0]']
dense_93 (Dense)	(None, 1)	33	['dense_92[0][0]']

Total params: 1,746,993
Trainable params: 1,746,993
Non-trainable params: 0

Figure 6.5: CNN-BiGRU Model Summary

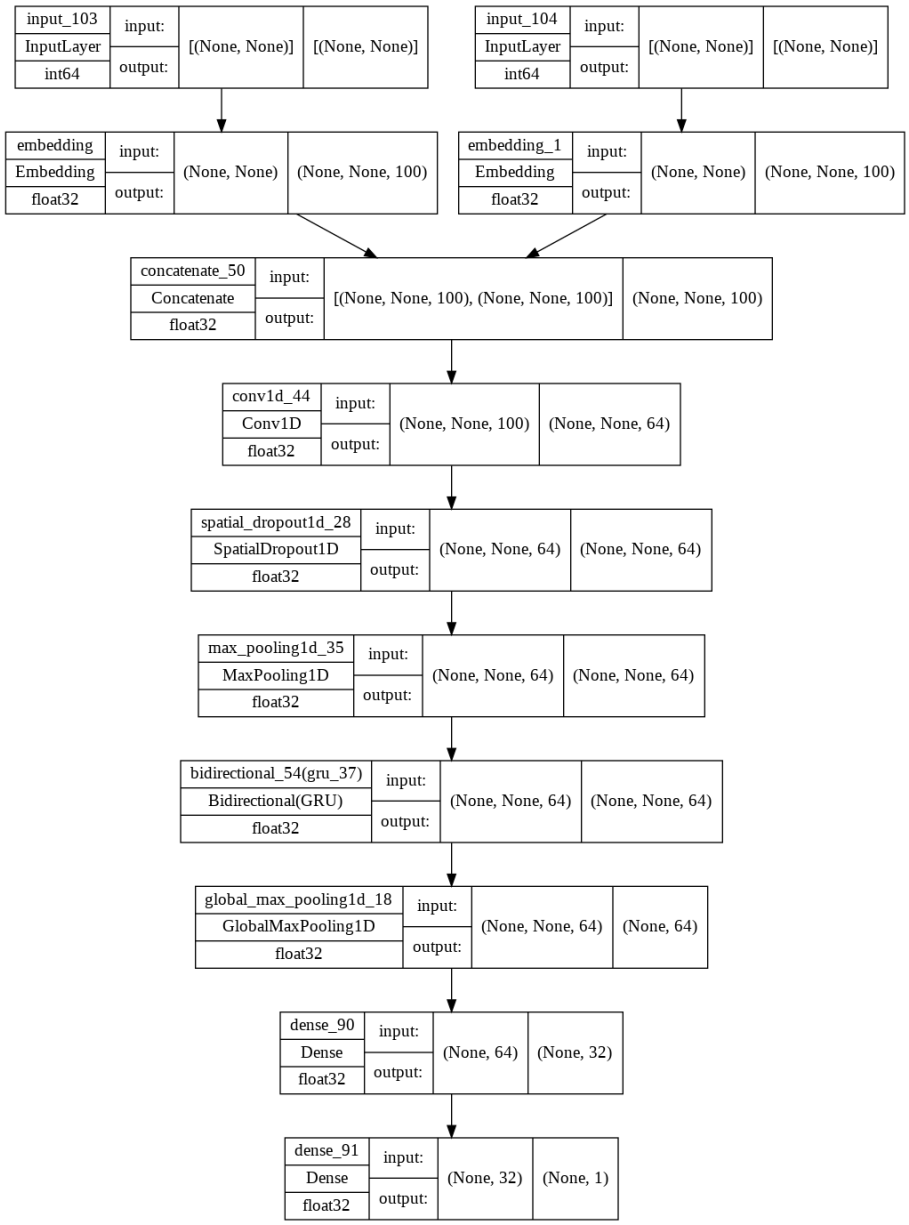


Figure 6.6: CNN-BiGRU Model Architecture

Chapter 7

Result and Discussion

We have tried both machine learning and deep learning approaches. The test results are then evaluated in four assessment criteria. They are
Accuracy: It is the number of accurate predictions out of total predictions.

$$\frac{\text{Number of Correct Prediction}}{\text{Total number of predictions made}}$$

F1 Score: The F1 Score summarizes a model's predictive effectiveness by merging two previously opposing variables, accuracy and recall.

$$2 * \frac{\text{precision * recall}}{\text{precision + recall}}$$

Precision: Precision determines the standard of positive predictions determined by the models. It's derived by dividing the total number of positive forecasts by the number of genuine positives.

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

Recall: It estimates a model's ability to determine positive samples. It is estimated within the range of [0,1].

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

7.1 Machine Learning Models

After training all the Machine Learning classifiers, we have observed the Precision, Recall, F1 score and Test accuracy for each model. We found that Random Forest has a satisfactory test accuracy of 91.66% and F1 score of 88.74%. Because it uses multiple decision trees for multiple features of the dataset, and the majority of prediction decisions are taken for the best result. Then Logistic Regression, MultinomialNB, KNN has gained 86.07%, 78.92% and 76.53% F1-Score sequentially.

7.1.1 Test Result

Models	Train/Test	Accuracy	F1-Score	Precision	Recall
SVC-Linear	Train	0.912595	0.873745	0.926531	0.842044
SVC-Linear	Test	0.833333	0.828978	0.904599	0.792457
MultinomialNB	Train	0.901583	0.851467	0.936126	0.748815
MultinomialNB	Test	0.866987	0.789236	0.903187	0.748815
Logistic Regression	Train	0.935306	0.911894	0.930422	0.89681
Logistic Regression	Test	0.900641	0.860744	0.891903	0.839009
KNN	Train	0.923606	0.898559	0.904194	0.893327
KNN	Test	0.833333	0.765278	0.790827	0.748707
Random Forest	Train	0.998624	0.9982	0.997333	0.999074
Random Forest	Test	0.916667	0.887432	0.900413	0.876401

Table 7.1: Test Result

7.1.2 SVM Linear

Hyper Parameter

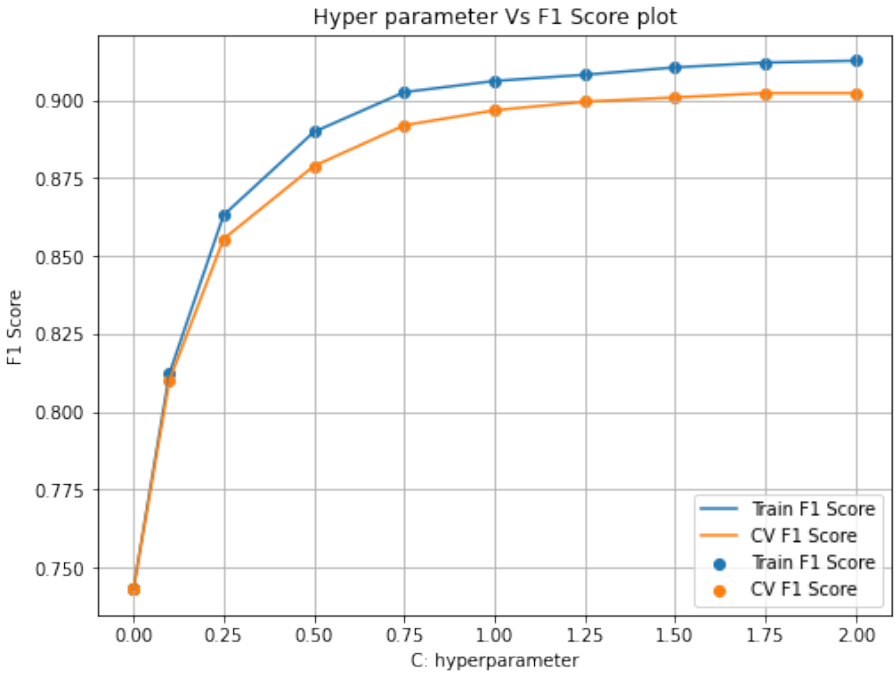


Figure 7.1: Hyper Parameter vs F1 Score for SVM Linear

We have visualized the hyper parameter vs F1 score so here we can see which best parameter model gives a good f1-score. In here the best parameter for SVM Linear is {'C': 1.75, 'kernel': 'linear'}

Confusion Matrix

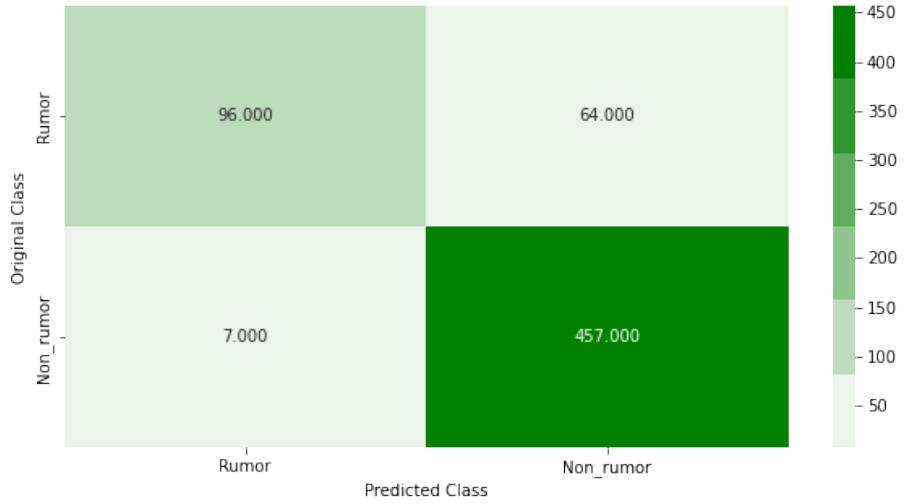


Figure 7.2: Confusion Matrix of SVM Linear

ROC Curve

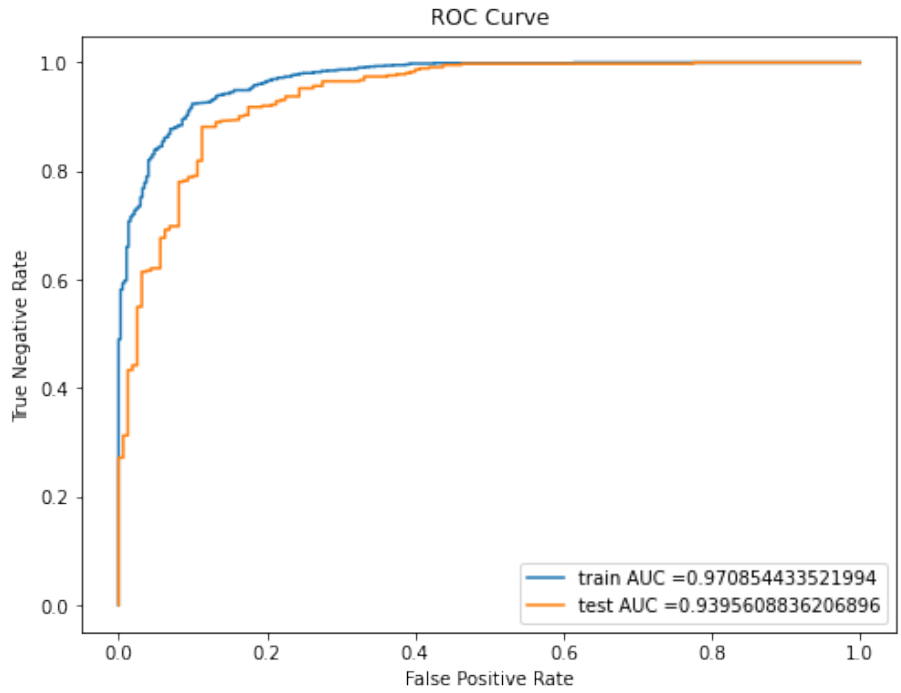


Figure 7.3: ROC curve for SVM Linear

This graph shows how a SVM Linear's diagnostic ability changes as the discrimination threshold is changed. As it is a binary classification, we visualize the roc to show how accurate the predictive power of this SVM Linear is.

7.1.3 Multinomial Naive bayes

Hyper Parameter

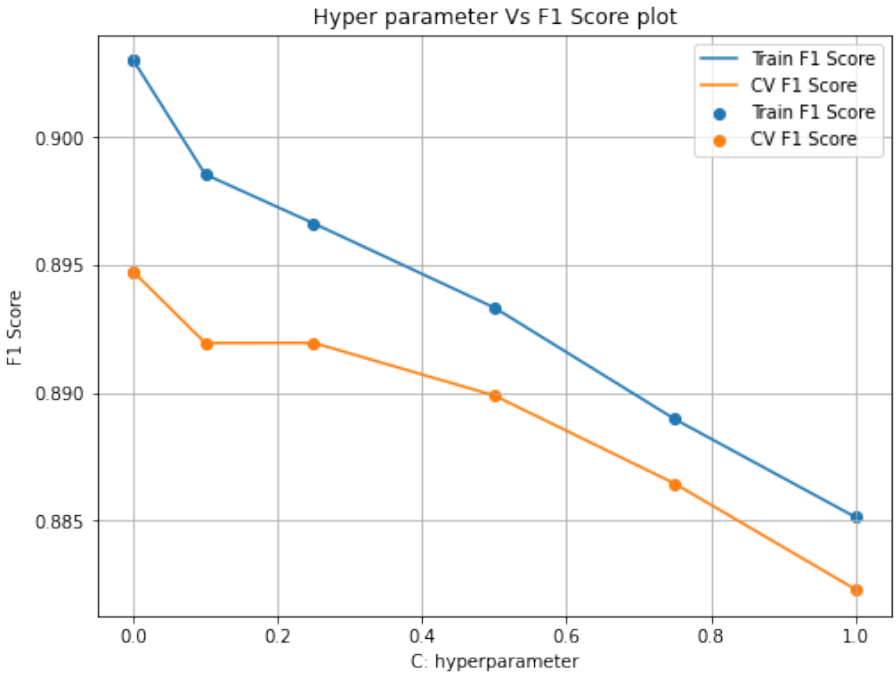


Figure 7.4: Hyper Parameter vs F1 Score for Multinomial Naive Bayes

We have visualized the hyper parameter vs F1 score so here we can see which best parameter model gives a good f1-score. In here, the Best hyperparameter for this Multinomial Naive bayes is {'alpha': 0.0001}

Confusion Matrix

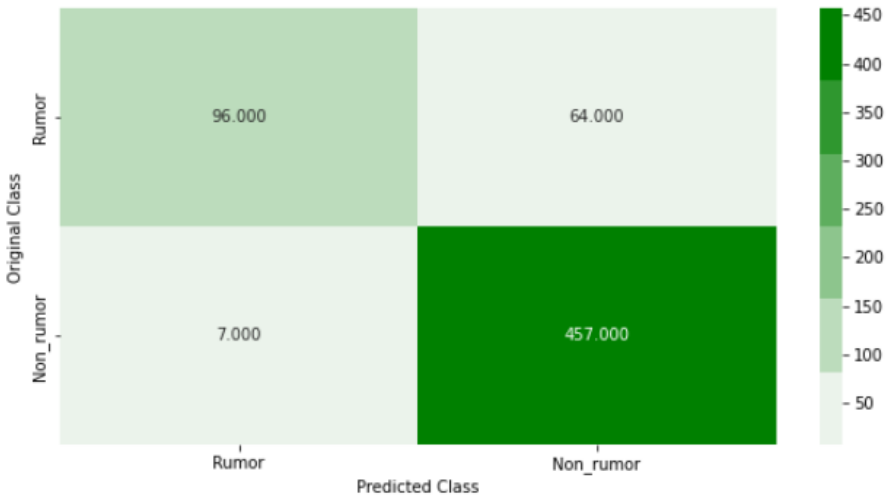


Figure 7.5: Confusion Matrix for Multinomial Naive Bayes

ROC Curve

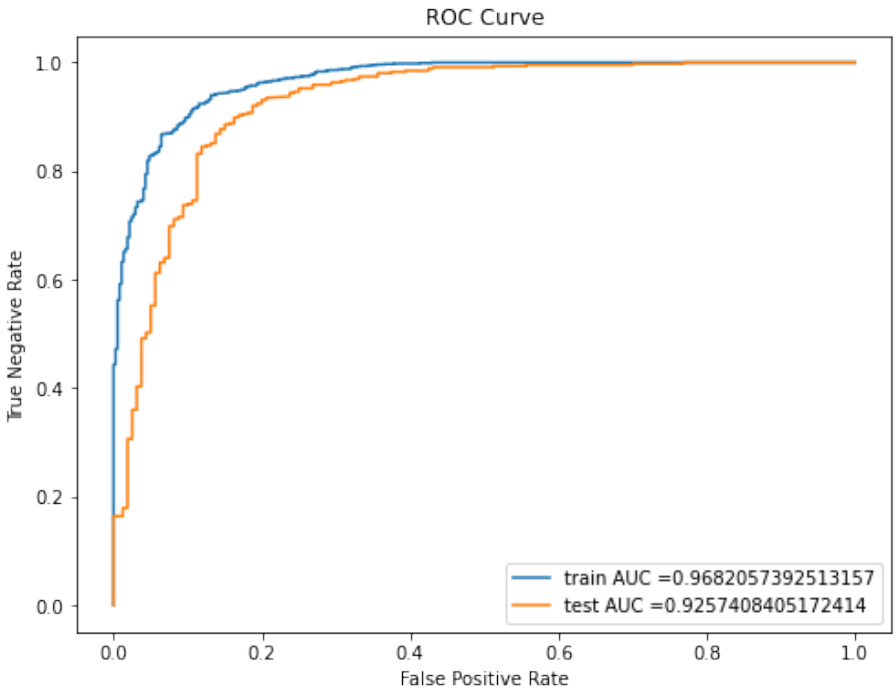


Figure 7.6: ROC curve Multinomial Naive bayes

This graph shows how a Multinomial Naive Bayes's diagnostic ability changes as the discrimination threshold are changed. As it is a binary classification, we visualize the roc to show how accurate the predictive power of this Multinomial Naive Bayes is.

7.1.4 Logistic Regression

Hyper Parameter

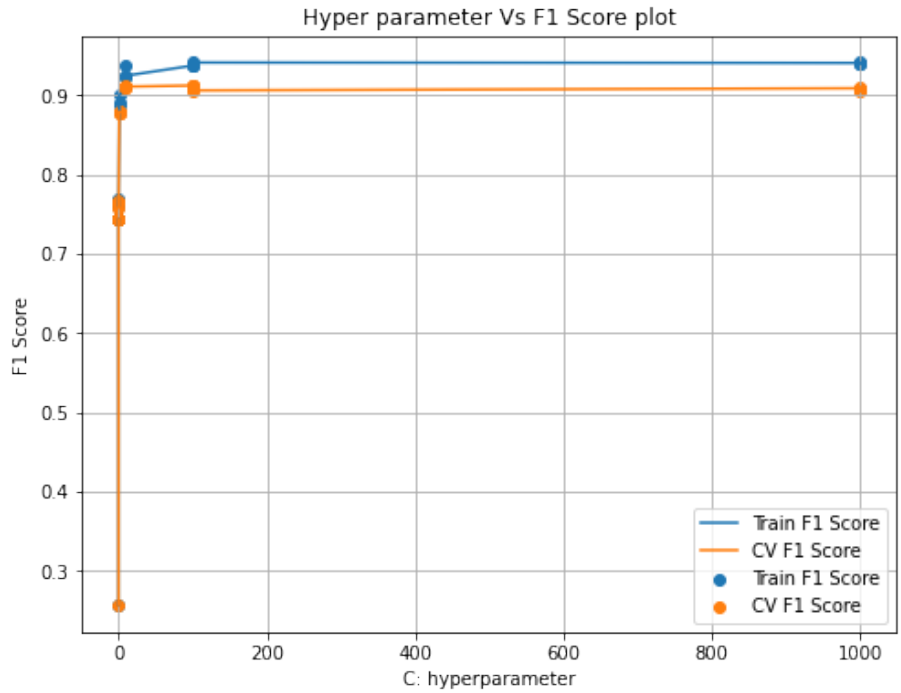


Figure 7.7: Hyper Parameter vs F1 Score for Logistic Regression

We have visualized the hyper parameter vs F1 score so here we can see which best parameter model gives a good f1-score. In here the Best hyperparameter for Logistic Regression is {'C': 100.0, 'penalty': 'l2', 'solver': 'newton-cg'}

Confusion Matrix

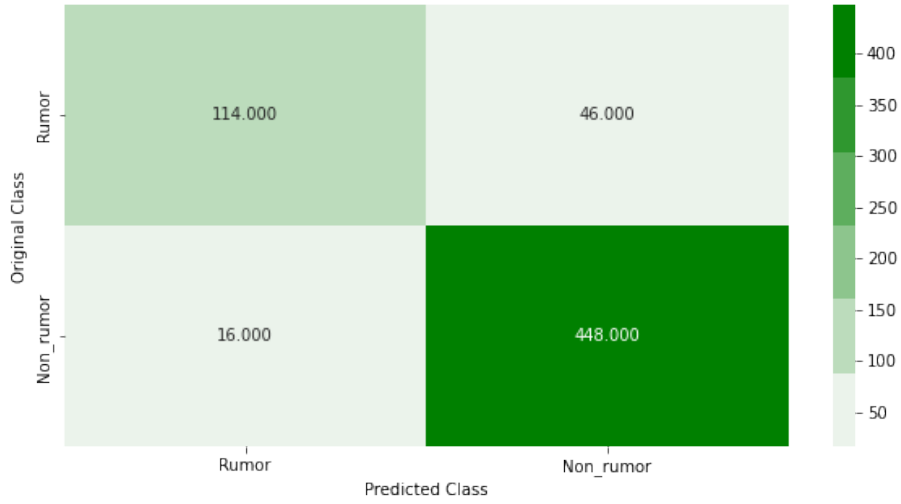


Figure 7.8: Confusion Matrix for Logistic Regression

ROC Curve

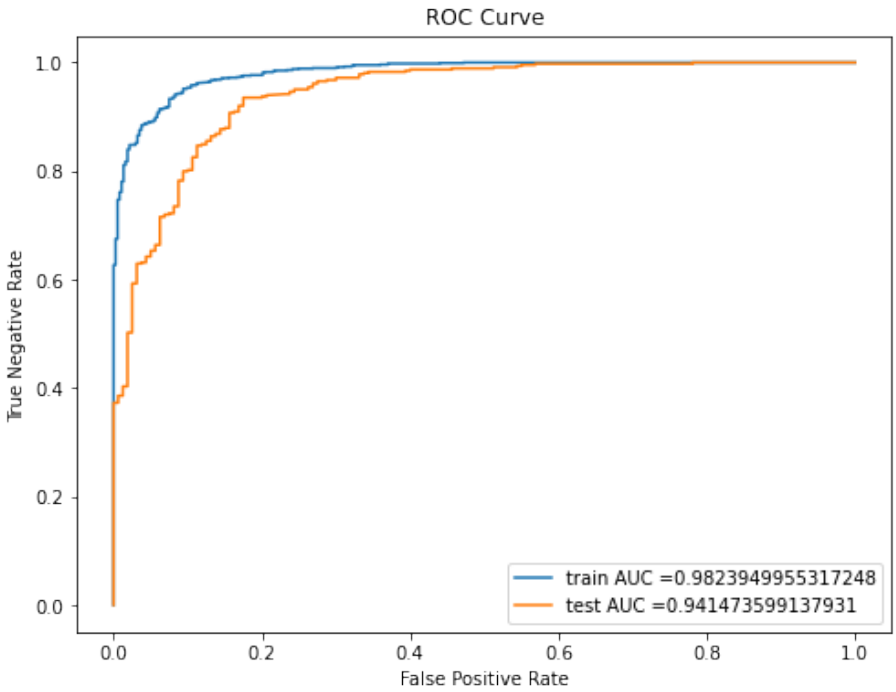


Figure 7.9: ROC curve for Logistic Regression

This graph shows how a Logistic Regression's diagnostic ability changes as the discrimination threshold are changed. As it is binary classification, we visualize the roc to show how accurate the predictive power of this Logistic Regression model is.

7.1.5 KNN

Hyper Parameter

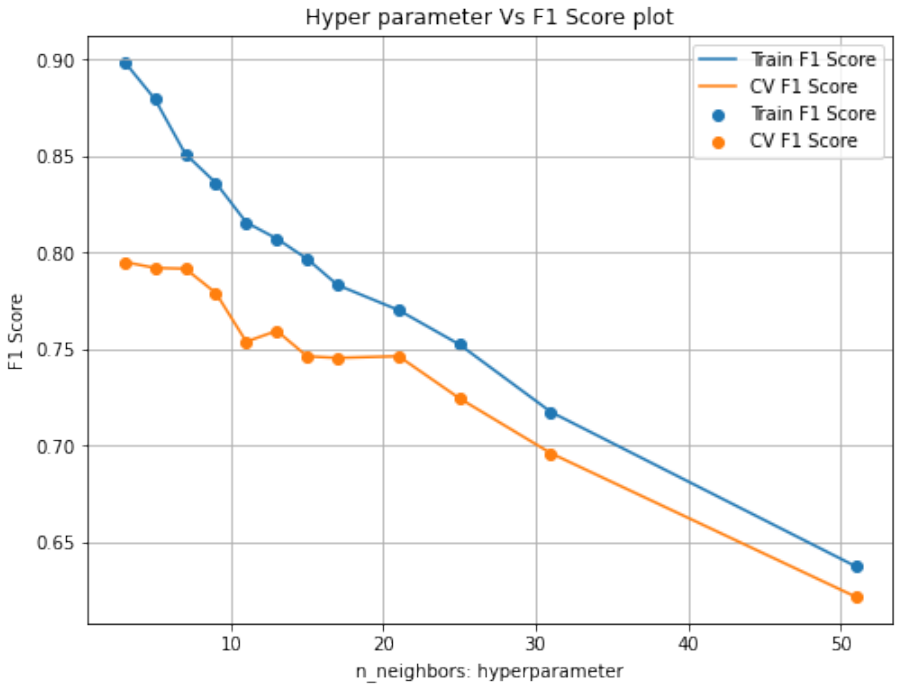


Figure 7.10: Hyper Parameter vs F1 Score for KNN

We have visualized the hyper parameter vs F1 score so here we can see which best parameter model gives a good f1-score. In here the Best hyperparameter for KNN $'n_neighbors' : 3$

Confusion Matrix

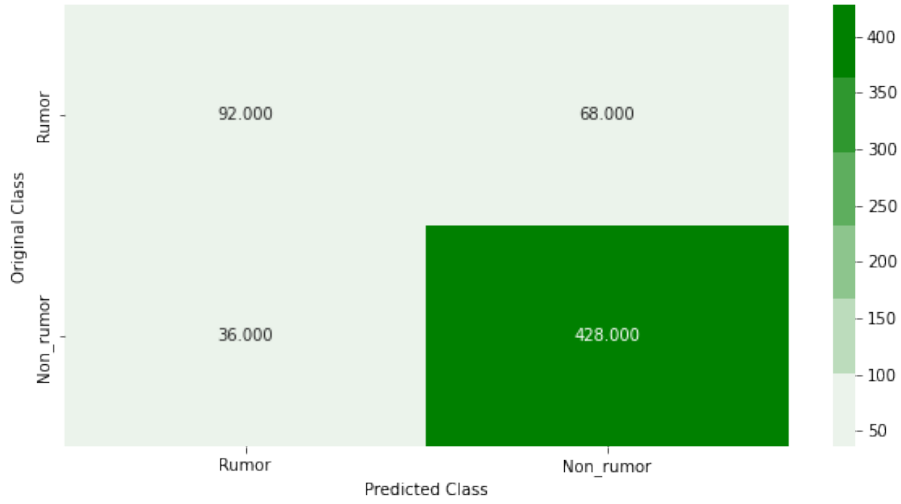


Figure 7.11: Confusion Matrix for KNN

ROC Curve

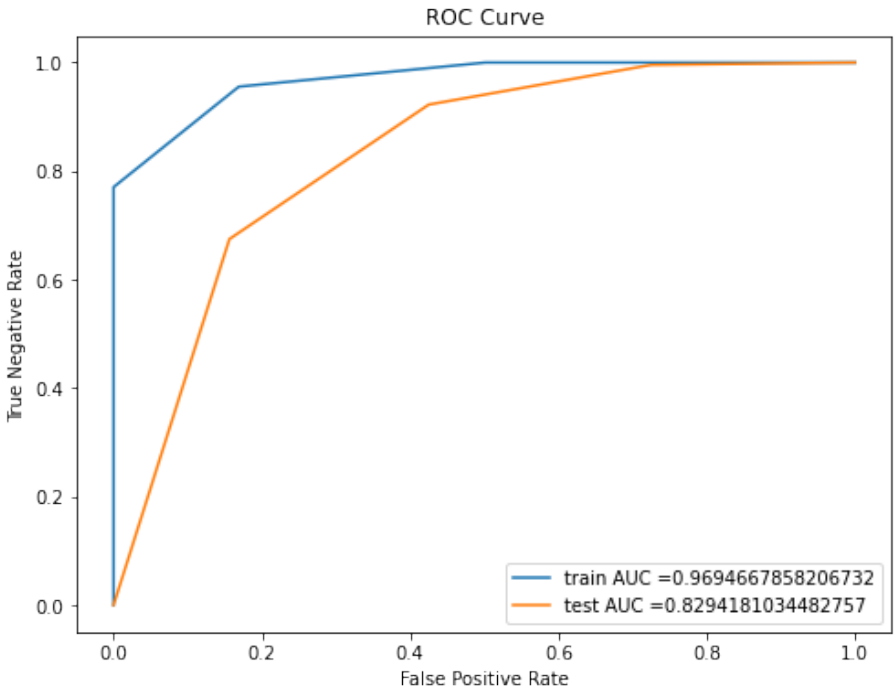


Figure 7.12: ROC curve Matrix for KNN

This graph shows how a KNN's diagnostic ability changes as the discrimination threshold is changed. As it is binary classification, we visualize the roc to show how accurate the predictive power of this KNN model is.

7.1.6 Random Forest

Hyper Parameter

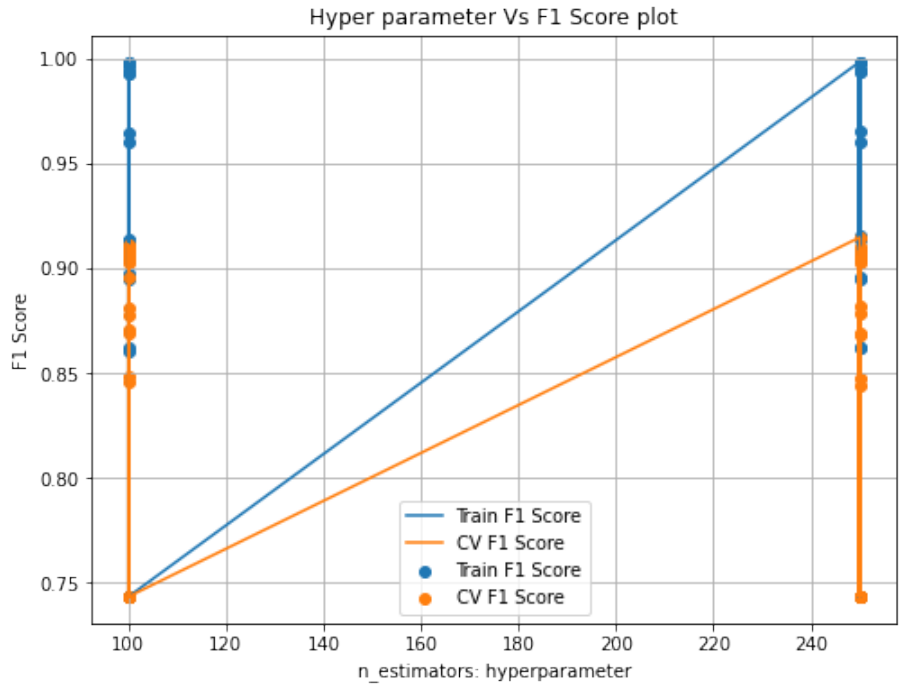


Figure 7.13: Hyper Parameter vs F1 Score for Random forest

We have visualized the hyper parameter vs F1 score so here we can see which best parameter model gives a good f1-score. In here the Best hyperparameter for Random forest $\{'\max_depth' : 250, 'max_samples' : 0.75, 'n_estimators' : 250\}$

Confusion Matrix

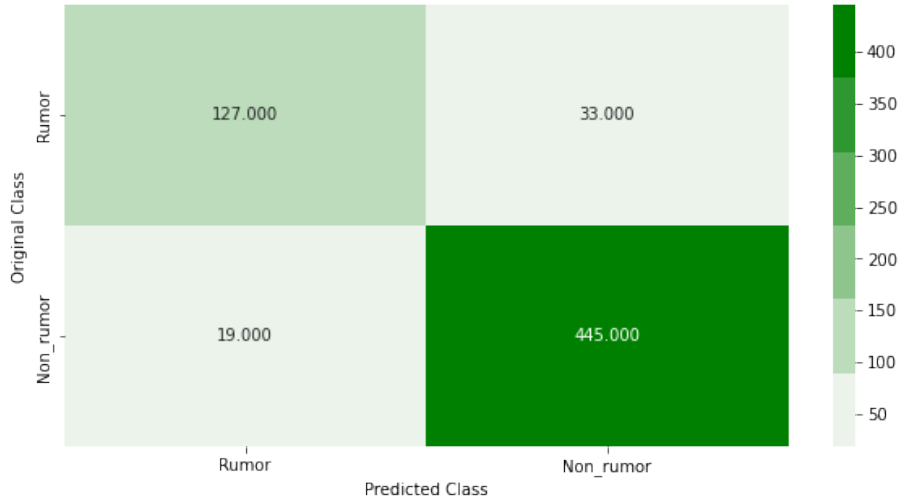


Figure 7.14: Confusion Matrix for Random forest

ROC Curve

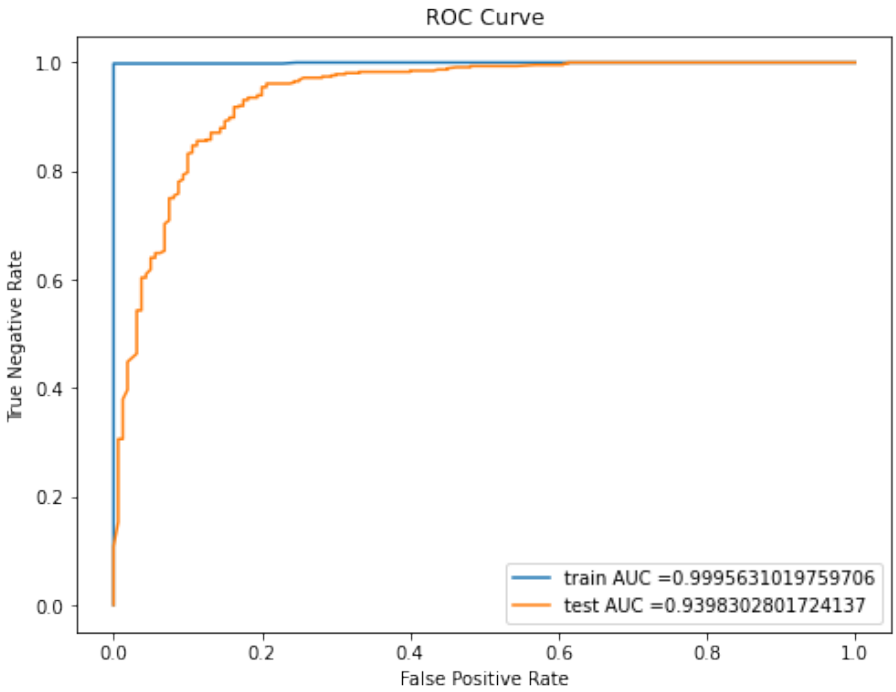


Figure 7.15: ROC curve for Random forest

This graph shows how a Random forest's diagnostic ability changes as the discrimination threshold is changed.As it is a binary classification, we visualize the roc to show how accurate the predictive power of this Random forest bayes is.

7.2 Deep learning Models

After training, we have observed the Precision, Recall, F1 score and Test accuracy for each combination. It is observed that all the deep learning models have gained a satisfactory test accuracy with Baseline and Glove as word embedding. Firstly, CNN baseline has gained F1 score for CNN baseline is 92.5% and CNN Glove is 90.5%. Secondly, our hybrid CNN-Bidirectional GRU was 94.1% with baseline and 91.3% with Glove. The F1 score for CNN-Bidirectional GRU baseline is 92.0% and CNN-Bidirectional GRU Glove is 87.2%. In all of the deep learning models we tested, we tried 3 different word embedding techniques in each along with a baseline. Here, the baseline would have the most words since the embedding layer is trained using the words in our dataset. So, it performs better. Moreover, unlike word2vec and fasttext, Glove does not only rely on local statistics but also global statistics. As a result, Glove also performs better in our models
On the other hand, other pre-trained embeddings like word2vec, and fasttext are not trained against text in the same context, so the number of common words between our text and these pre-trained embeddings is not the same. As a result, we got the best result in Baseline embedding.

Models	Word Embedding	Accuracy	F1-Score	Precision	Recall
CNN	Baseline	0.942308	0.925557	0.919173	0.952586
CNN	glove	0.929487	0.905167	0.916534	0.965517
CNN	word2vec	0.907051	0.880531	0.873568	0.926724
CNN	fasttext	0.878205	0.85447	0.834493	0.862069
Bi-LSTM	Baseline	0.865385	0.81399	0.834378	0.935345
Bi-LSTM	glove	0.88141	0.838371	0.853691	0.939655
Bi-LSTM	word2vec	0.798077	0.76092	0.747119	0.801724
Bi-LSTM	fasttext	0.782051	0.758195	0.752725	0.737069
CNN-Bi GRU	Baseline	0.939103	0.919148	0.924648	0.965517
CNN-Bi GRU	glove	0.913462	0.872393	0.940325	0.99569
CNN-Bi GRU	word2vec	0.900641	0.85653	0.906244	0.978448
CNN-Bi GRU	fasttext	0.891026	0.863552	0.85	0.900862

Table 7.2: Test Result of Deep Learning Models in different embedding

7.2.1 CNN

Word Embedding	Train accuracy	Val accuracy	Test accuracy	Train Loss	Val Loss	Test Loss
Baseline	1.0	0.946	0.942	0.019	0.173	0.168
glove	0.973	0.929	0.929	0.094	0.198	0.206
fasttext	0.920	0.840	0.878	0.215	0.350	0.301
word2vec	0.955	0.891	0.907	0.140	0.279	0.260

Table 7.3: CNN Test Result

This table shows that without pertained word embedding (baseline) and with pertained word embedding weights train,test,validation accuracy and loss.

Graphs

Baseline

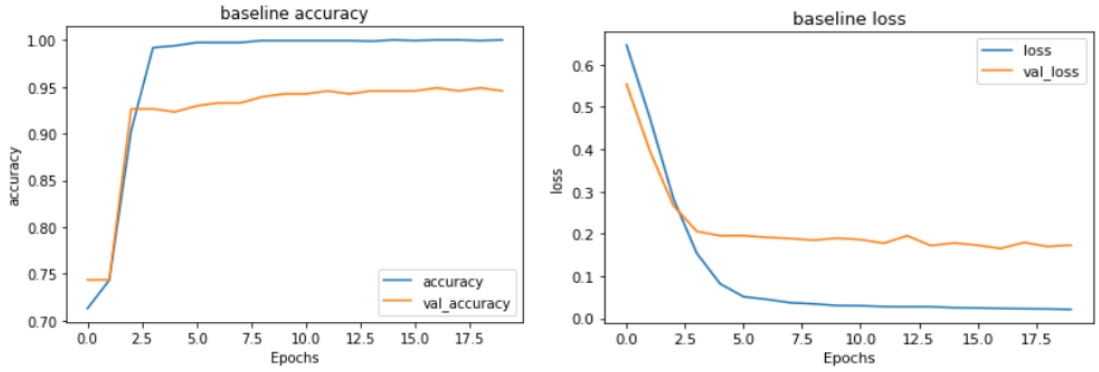


Figure 7.16: Baseline Accuracy and Baseline Loss graph of CNN

In this figure we have visualize the accuracy and loss of train data and validation data for each epoch to understand the performance of a model

Glove

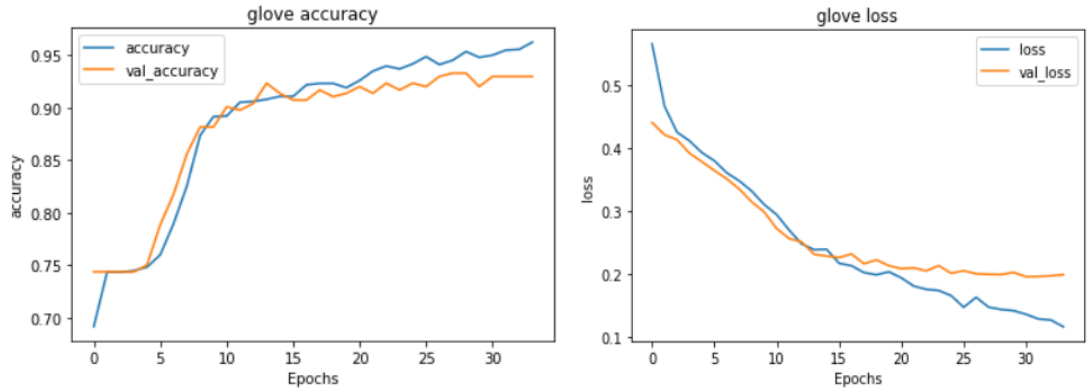


Figure 7.17: Globe Accuracy and Glove Loss graph of CNN

In this figure we have visualized the accuracy and loss of train data and validation data for each epoch to understand the performance of a model.

Word2vec

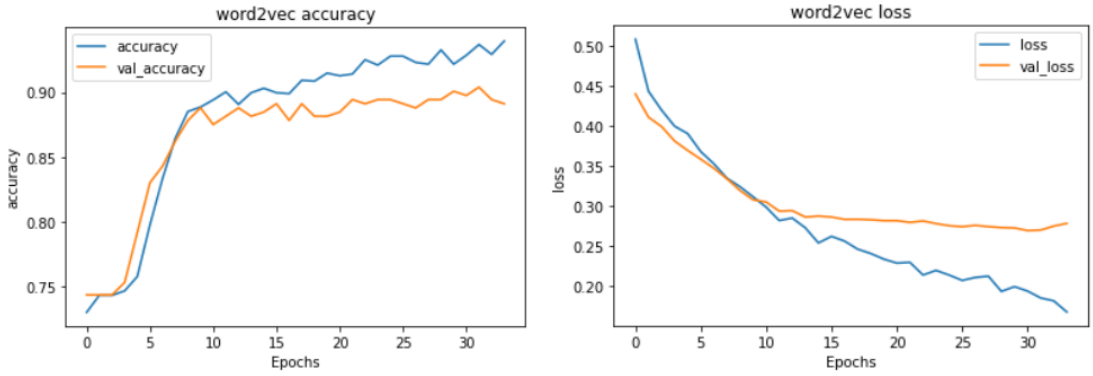


Figure 7.18: word2vec Accuracy and word2vec Loss graph of CNN

In this figure we have visualized the accuracy and loss of train data and validation data for each epoch to understand the performance of a model.

Fasttext

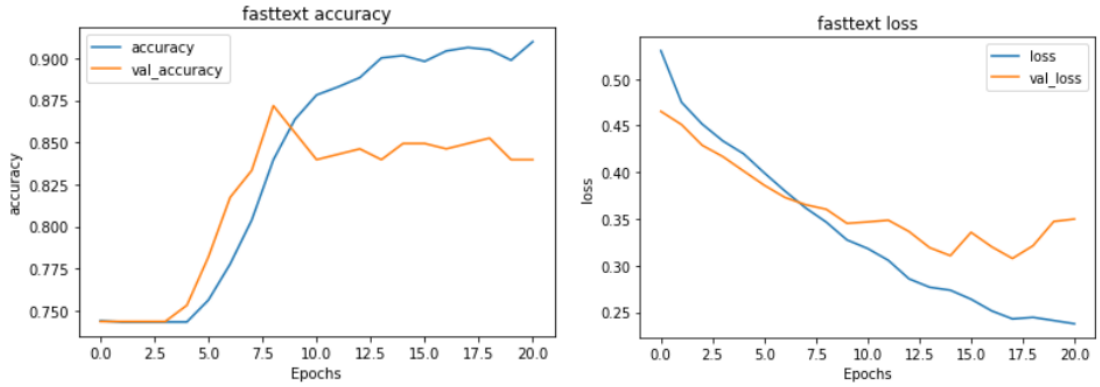


Figure 7.19: Fasttext Accuracy and Fasttext Loss graph of CNN

In this figure we have visualize the accuracy and loss of train data and validation data for each epoch to understand the performance of a model

Training Accuracy

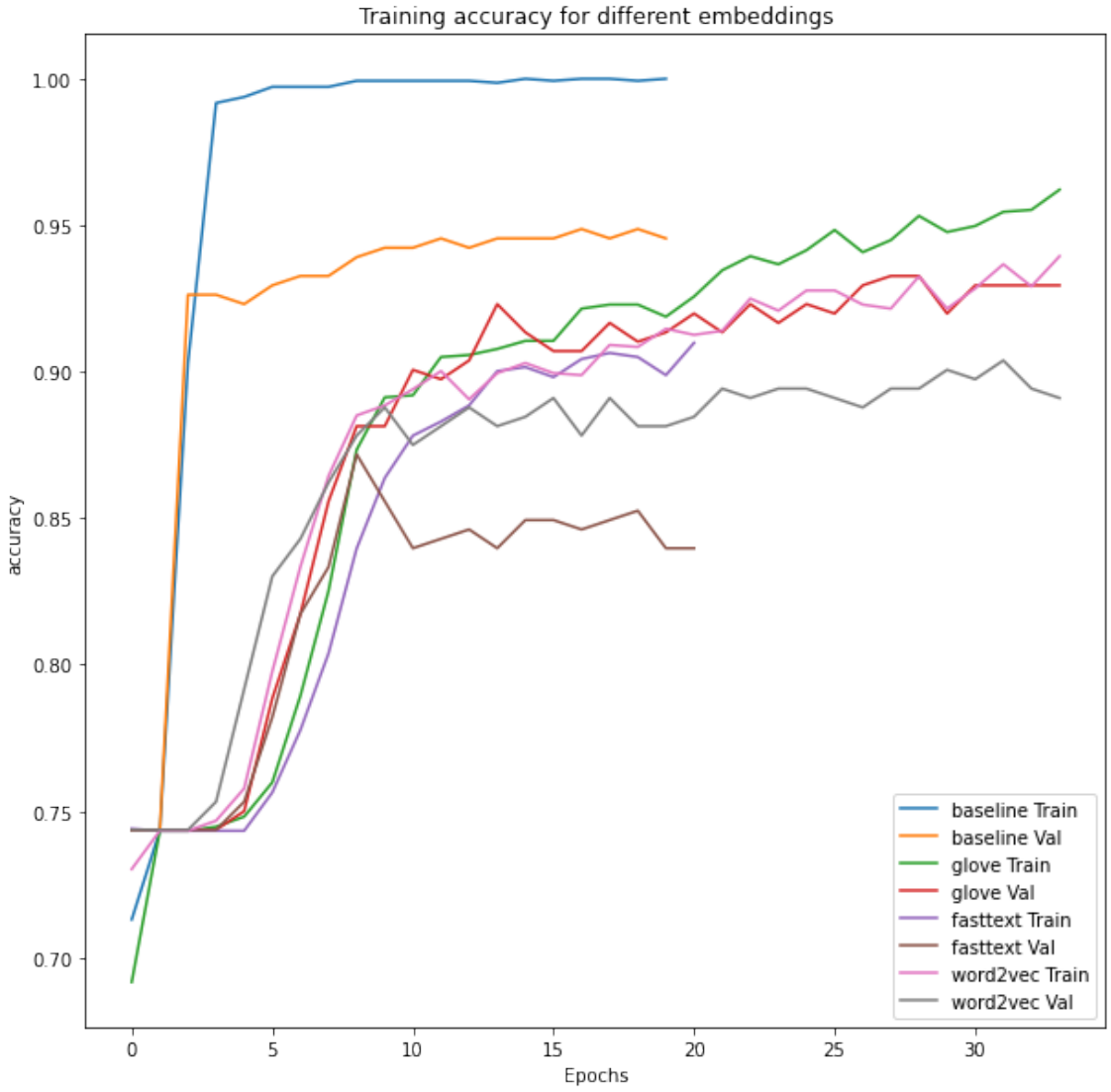


Figure 7.20: Training accuracy for CNN

In this figure we have visualized the loss of train data and validation data with every word embedding for each epoch.

Training Loss

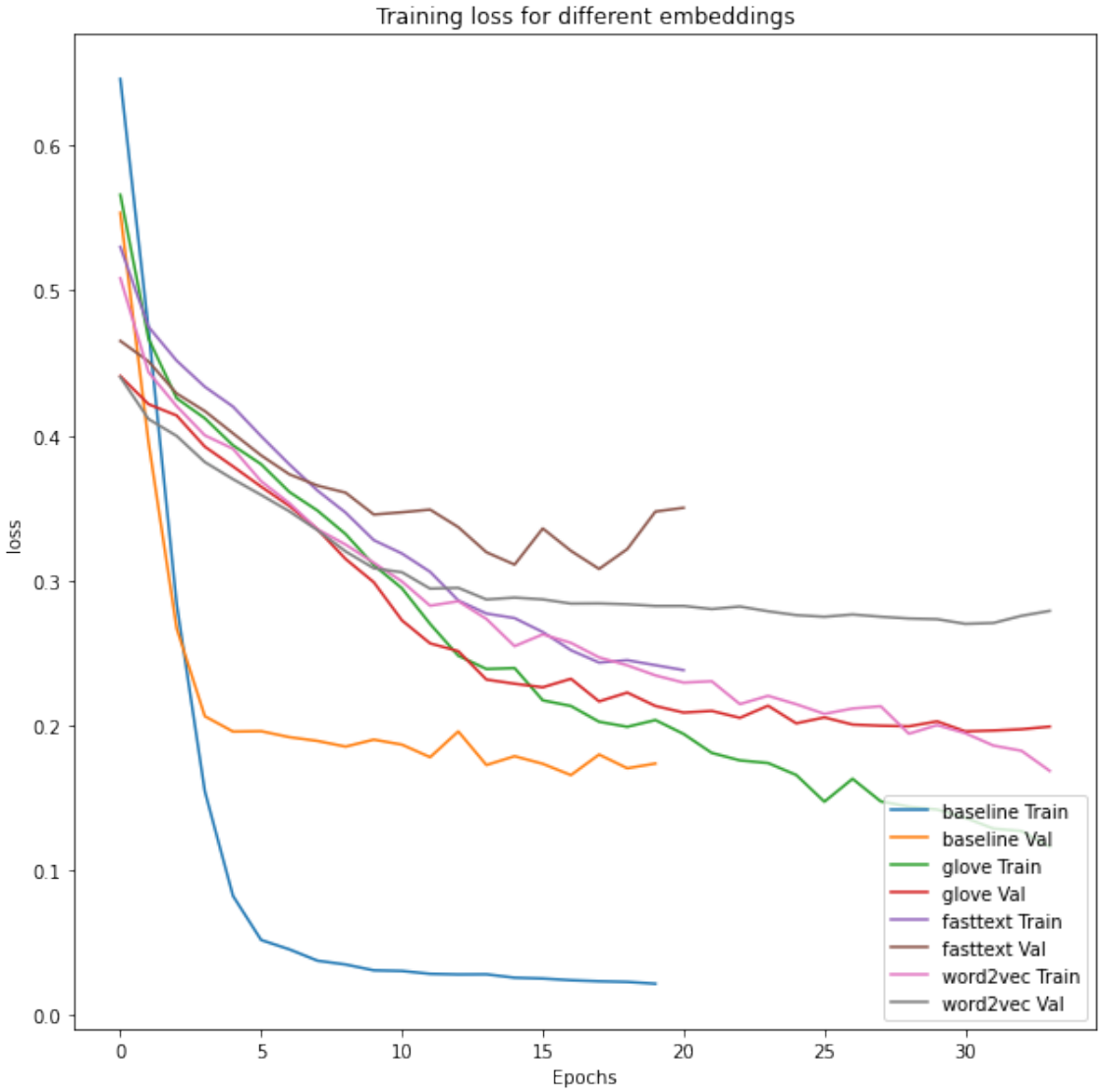


Figure 7.21: Training loss for CNN

Confusion Matrix

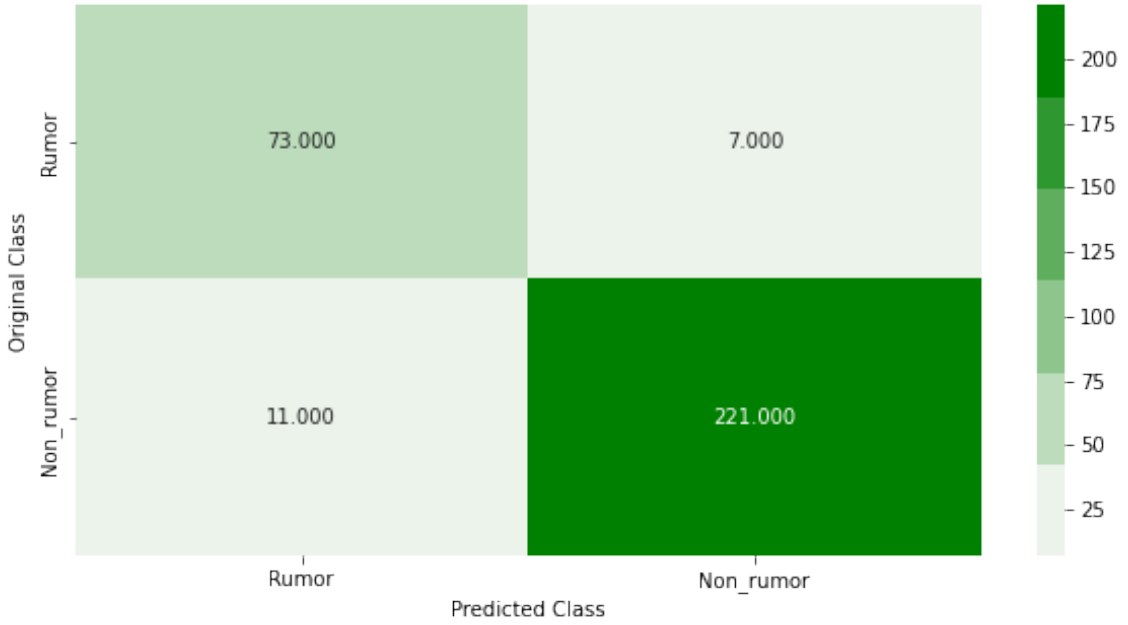


Figure 7.22: Confusion Matrix of CNN

7.2.2 Two Stacked Bi-Directional LSTM

Word Embedding	Train accuracy	Val accuracy	Test accuracy	Train Loss	Val Loss	Test Loss
Baseline	0.9993	0.8558	0.86538	0.36125	1.0439	1.0190
glove	0.98004	0.8782	0.8814	0.4496	0.7777	0.8877
fasttext	0.9036	0.8237	0.7981	0.69277	0.9426	1.0033
word2vec	0.9015	0.8173	0.7820	0.62125	0.80924	0.8545

Table 7.4: Two Stacked Bi-Directional LSTM Test Result

This table shows that without pertained word embedding (baseline) and with pertained word embedding weights train,test,validation accuracy and loss.

Graphs

Baseline

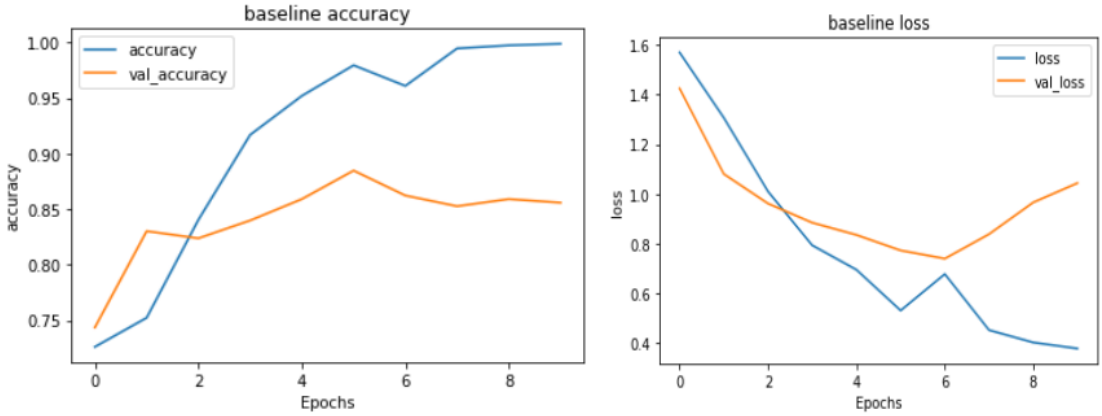


Figure 7.23: Baseline Accuracy and Baseline Loss graph of Two Stacked Bi-Directional LSTM

In this figure, we have visualize the accuracy and loss of train data and validation data for each epoch to understand the performance of a model.

Glove

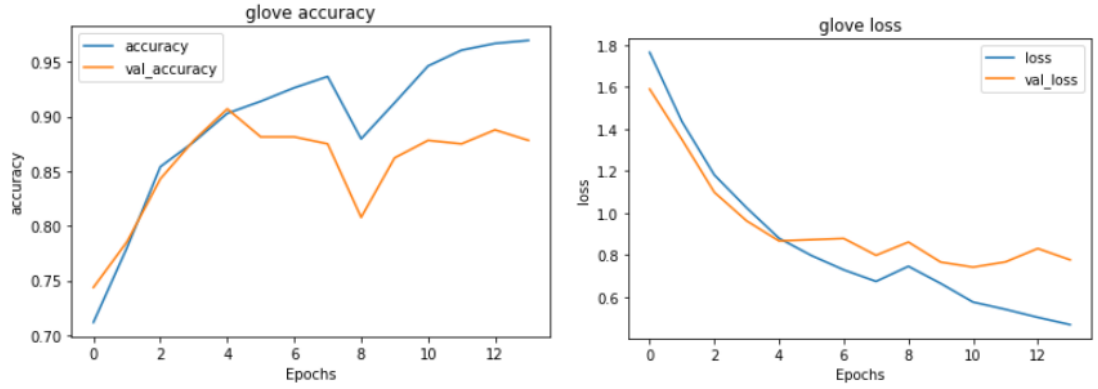


Figure 7.24: Globe Accuracy and Glove Loss graph of Two Stacked Bi-Directional LSTM

In this figure, we have visualized the accuracy and loss of train data and validation data for each epoch to understand the performance of a model.

Word2vec

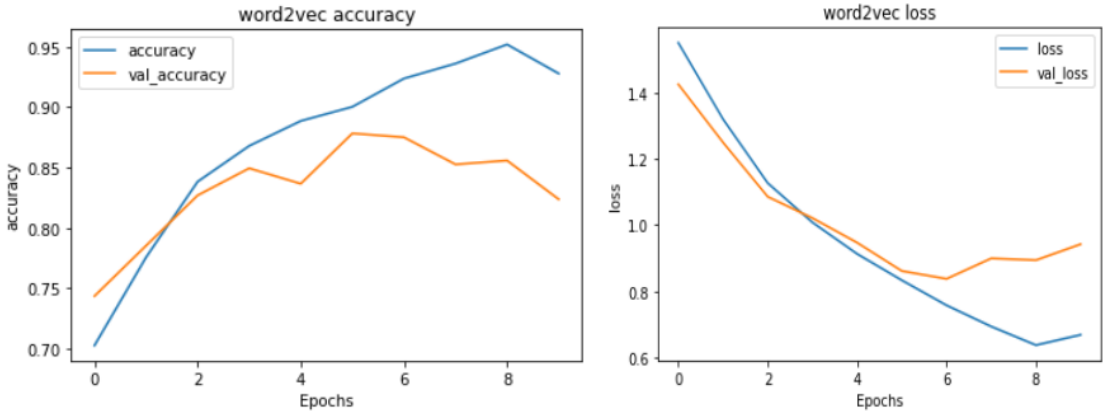


Figure 7.25: word2vec Accuracy and word2vec Loss graph of Two Stacked Bi-Directional LSTM

In this figure, we have visualized the accuracy and loss of train data and validation data for each epoch to understand the performance of a model.

Fasttext

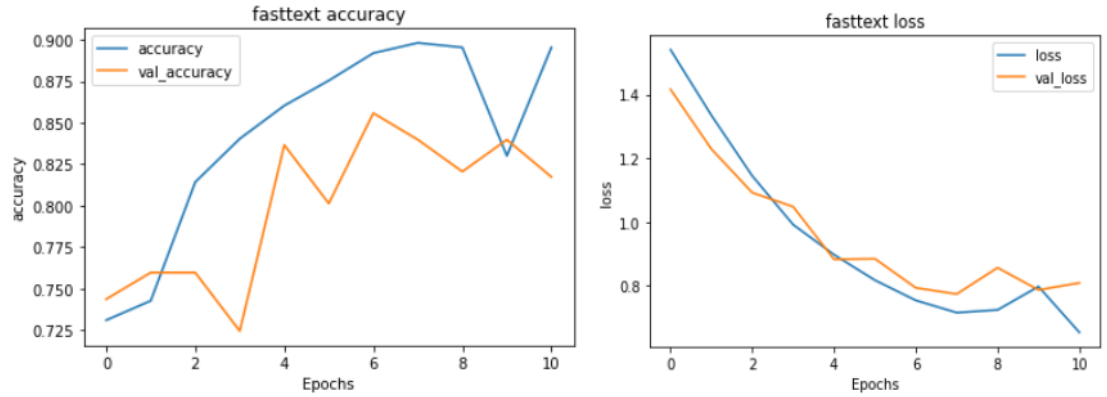


Figure 7.26: Fasttext Accuracy and Fasttext Loss graph of Two Stacked Bi-Directional LSTM

In this figure, we have visualized the accuracy and loss of train data and validation data for each epoch to understand the performance of a model

Training Accuracy

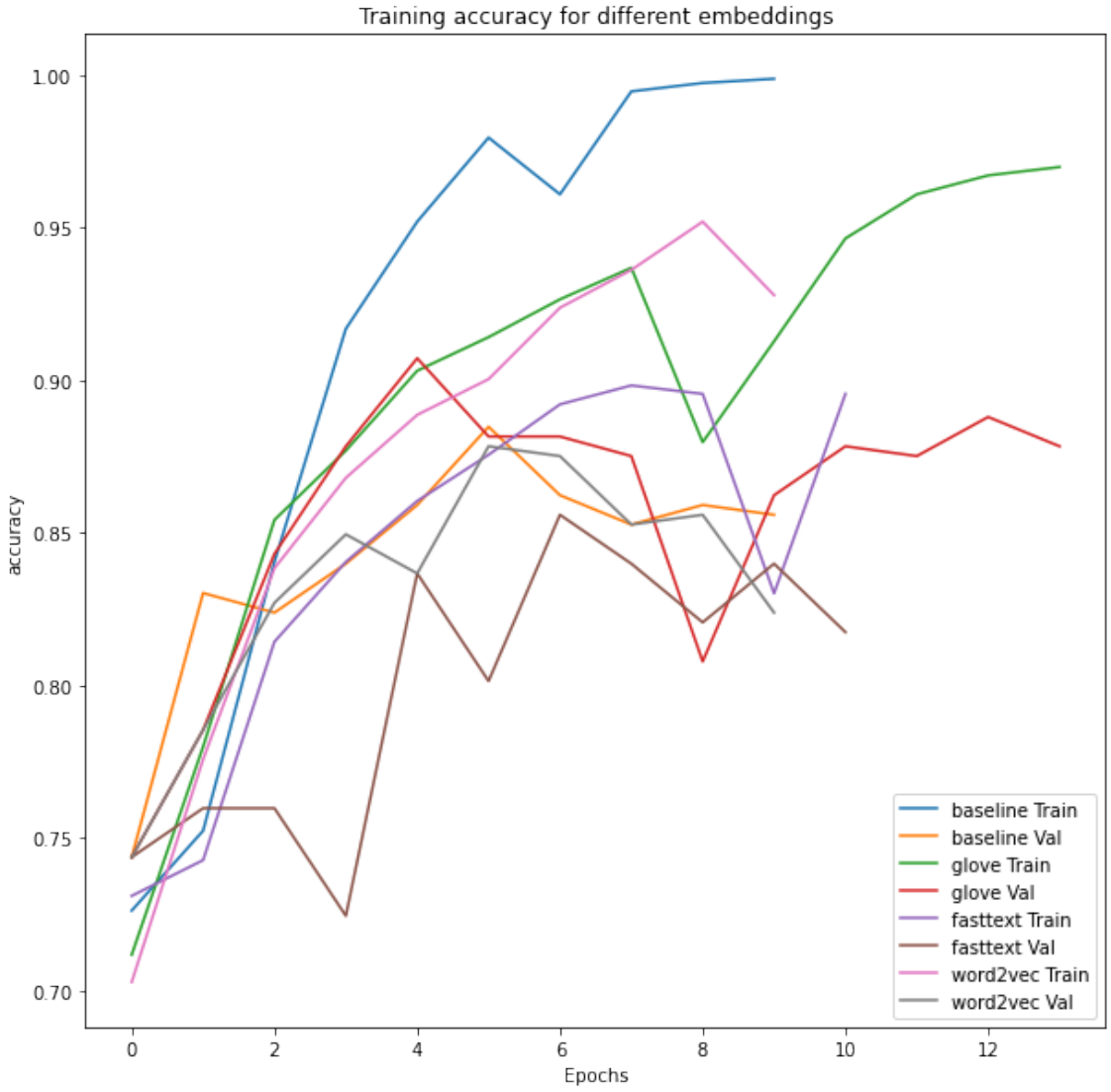


Figure 7.27: Training accuracy for Two Stacked Bi-Directional LSTM

Training Loss

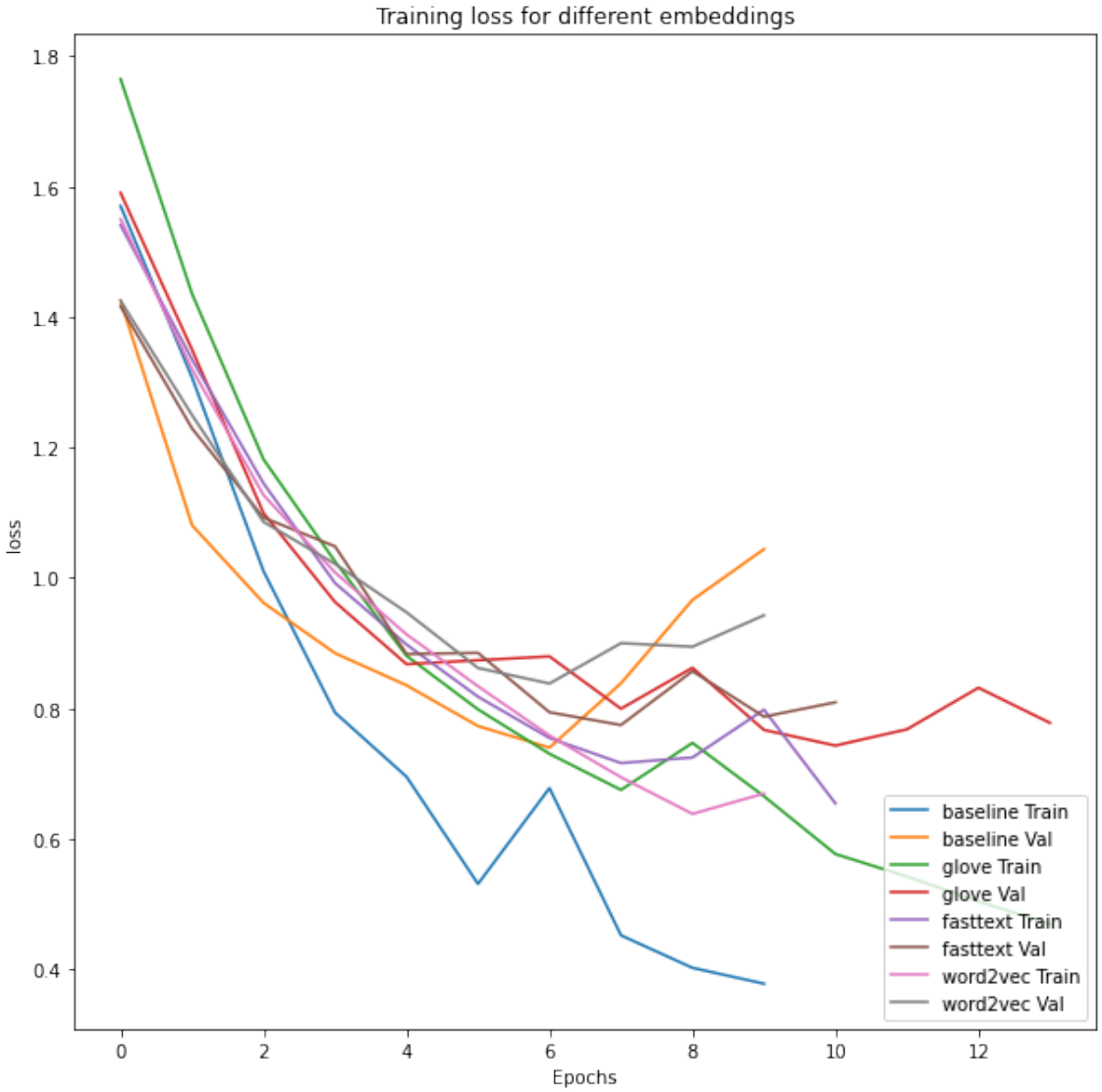


Figure 7.28: Training loss for Two Stacked Bi-Directional LSTM

In this figure, we have visualized the loss of train data and validation data with every word embedding for each epoch.

Confusion Matrix

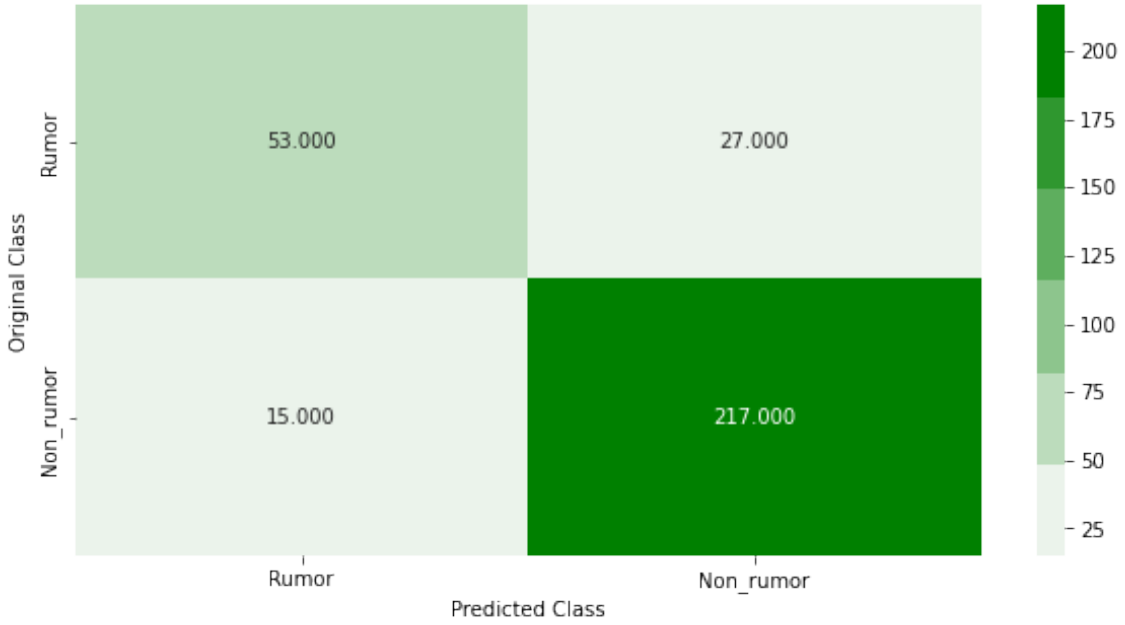


Figure 7.29: Confusion Matrix of Two Stacked Bi-Directional LSTM

ROC Curve

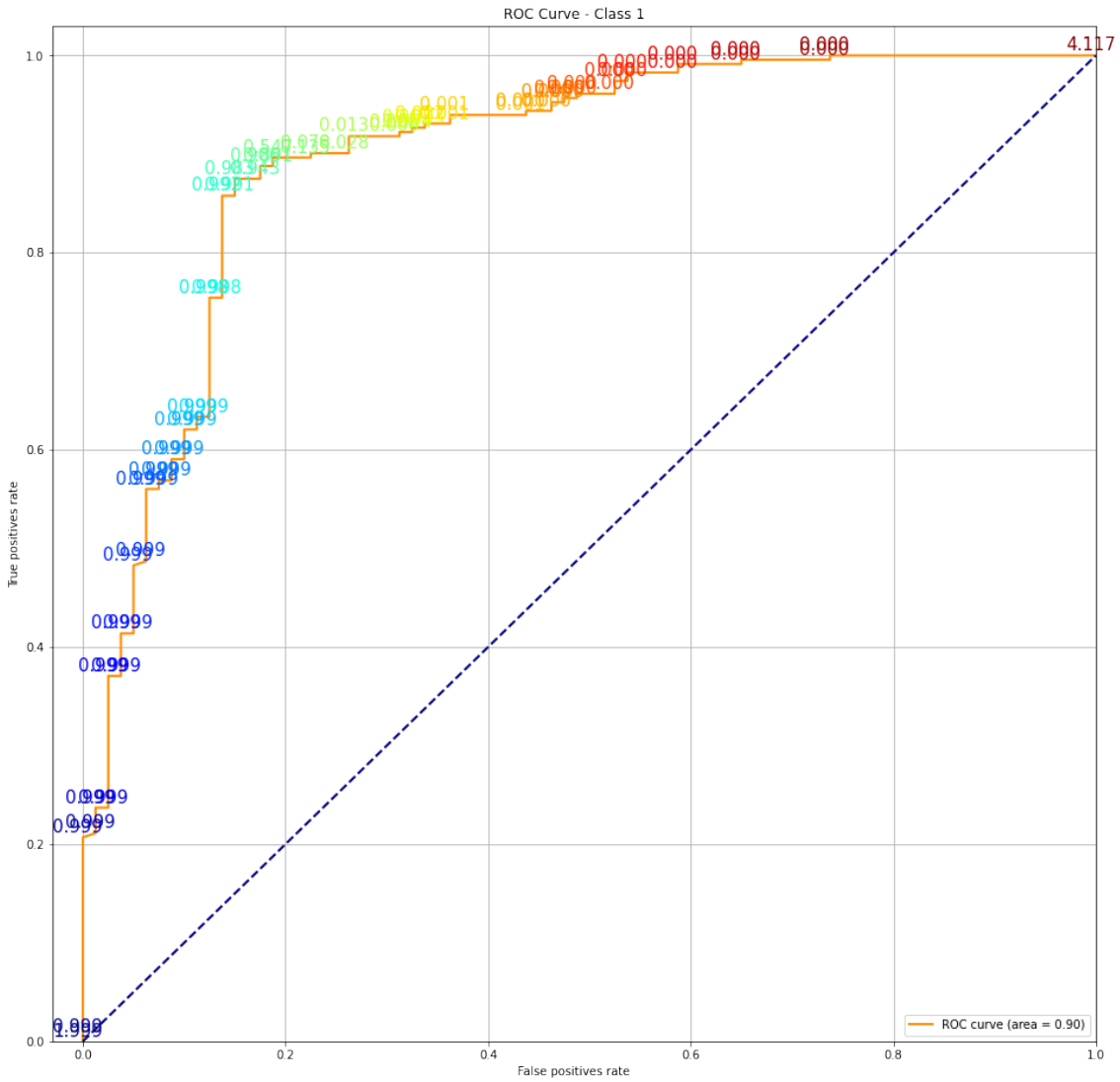


Figure 7.30: ROC curve for Baseline Two Stacked Bi-Directional LSTM

This graph shows how a Two Stacked Bi-Directional LSTM's diagnostic ability changes as the discrimination threshold is changed.

7.2.3 CNN-Bidirectional GRU

Word Embedding	Train accuracy	Val accuracy	Test accuracy	Train Loss	Val Loss	Test Loss
Baseline	0.9989	0.9134	0.93910	0.27454	0.3611	0.3464
glove	0.99862	0.92948	0.91346	0.3083	0.370	0.37823
fasttext	0.9305	0.8718	0.8910	0.435	0.4718	0.46509
word2vec	0.9972	0.9007	0.90064	0.2312	0.34407	0.35427

Table 7.5: CNN-Bidirectional GRU Test Result

This table shows that without pertained word embedding (baseline) and with pertained word embedding weights train,test,validation accuracy and loss.

Graphs

Baseline

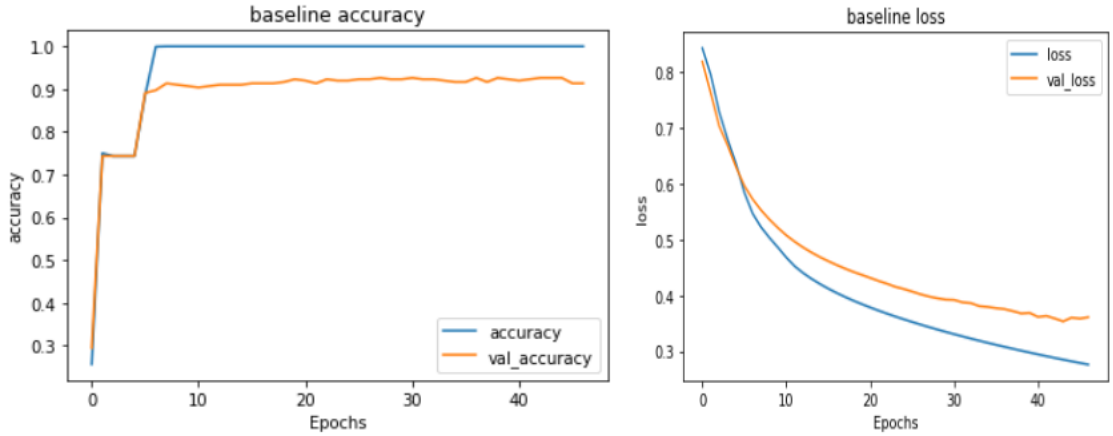


Figure 7.31: Baseline Accuracy and Baseline Loss graph of CNN-Bidirectional GRU

In this figure, we have visualized the accuracy and loss of train data and validation data for each epoch to understand the performance of a model.

Glove

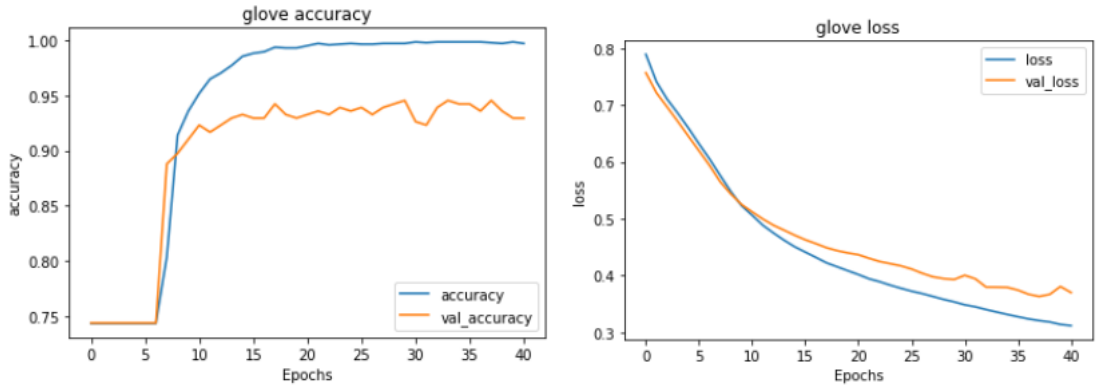


Figure 7.32: Globe Accuracy and Glove Loss graph of CNN-Bidirectional GRU

In this figure, we have visualized the accuracy and loss of train data and validation data for each epoch to understand the performance of a model.

Word2vec

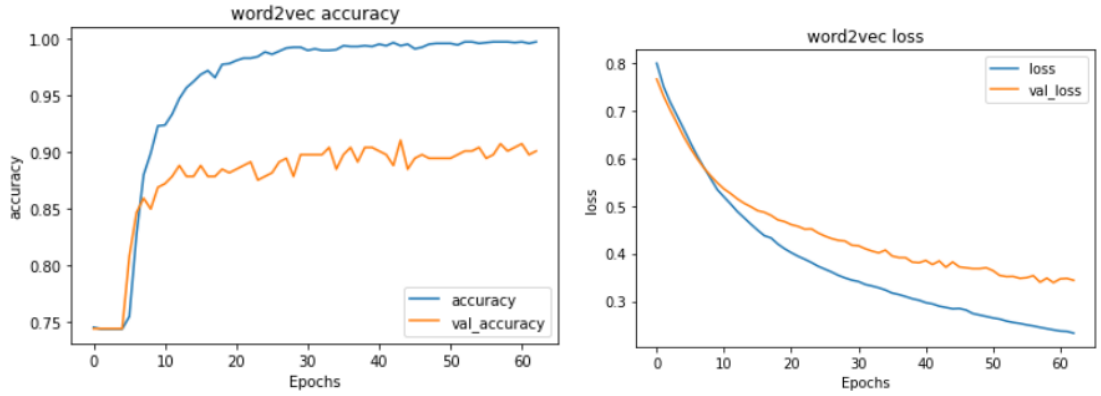


Figure 7.33: word2vec Accuracy and word2vec Loss graph of CNN-Bidirectional GRU

In this figure, we have visualized the accuracy and loss of train data and validation data for each epoch to understand the performance of a model.

Fasttext

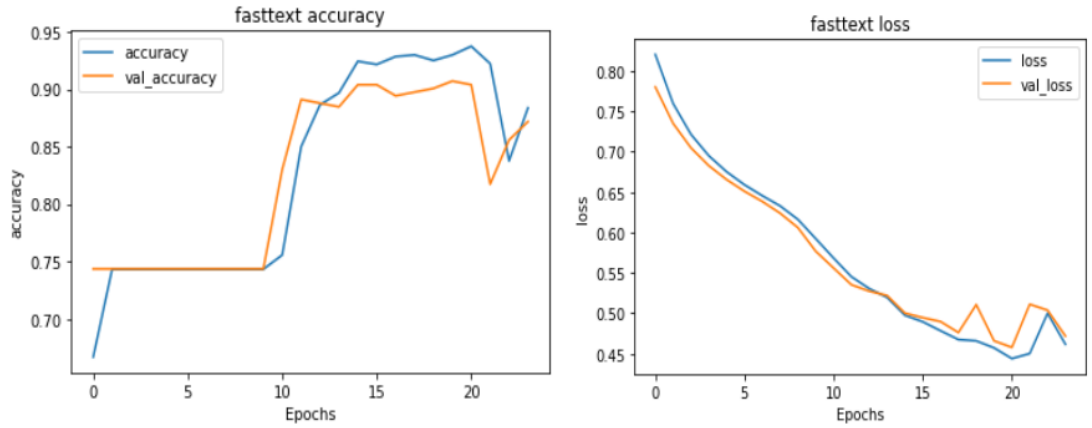


Figure 7.34: Fasttext Accuracy and Fasttext Loss graph of CNN-Bidirectional GRU

In this figure, we have visualized the accuracy and loss of train data and validation data for each epoch to understand the performance of a model.

Training Accuracy

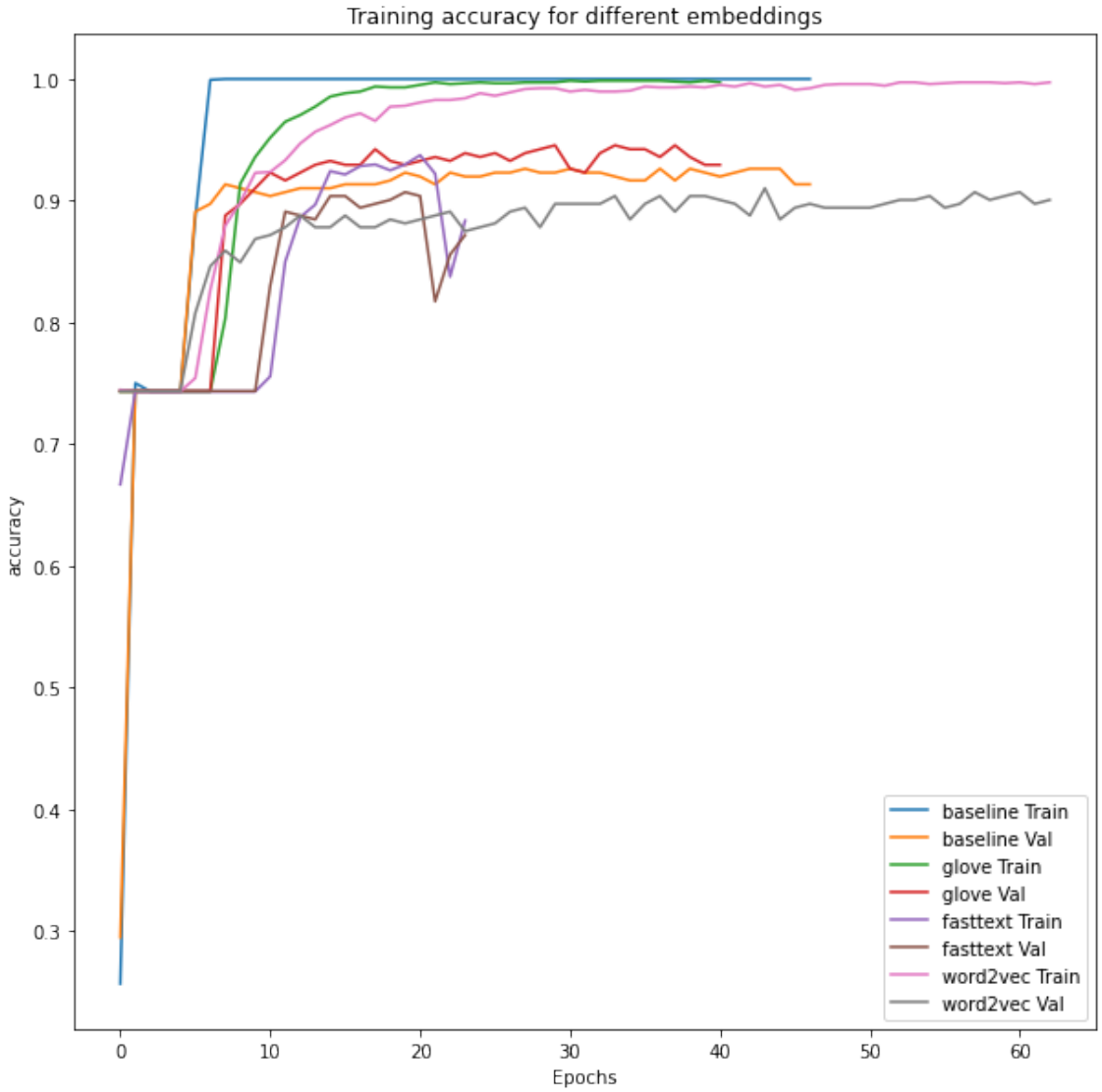


Figure 7.35: Training accuracy for CNN-Bidirectional GRU

In this figure, we have visualized the accuracy of train data and validation data with every word embedding for each epoch.

Training Loss

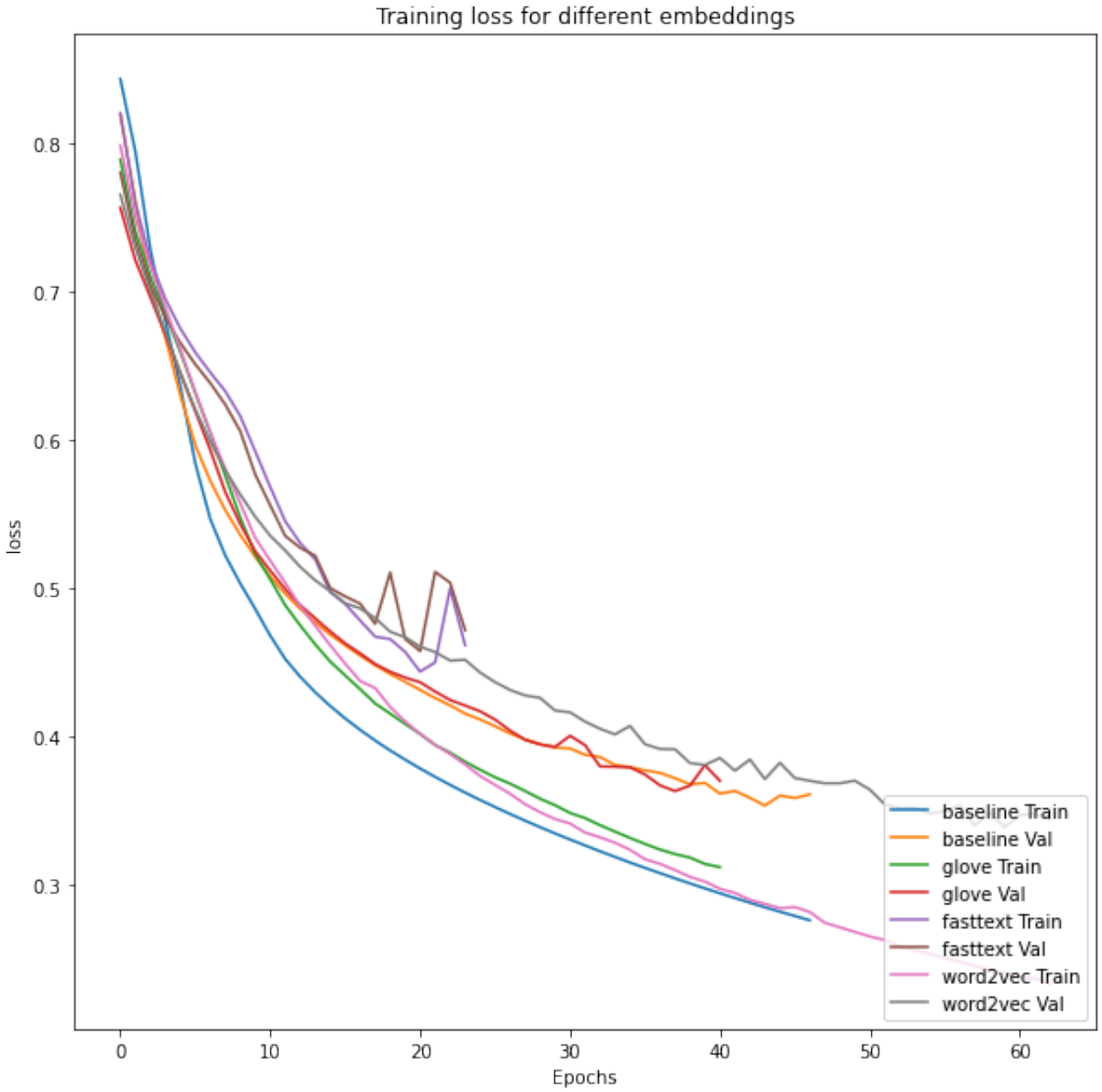


Figure 7.36: Training loss for CNN-Bidirectional GRU

In this figure we have visualized the loss of train data and validation data with every word embedding for each epoch.

Confusion Matrix

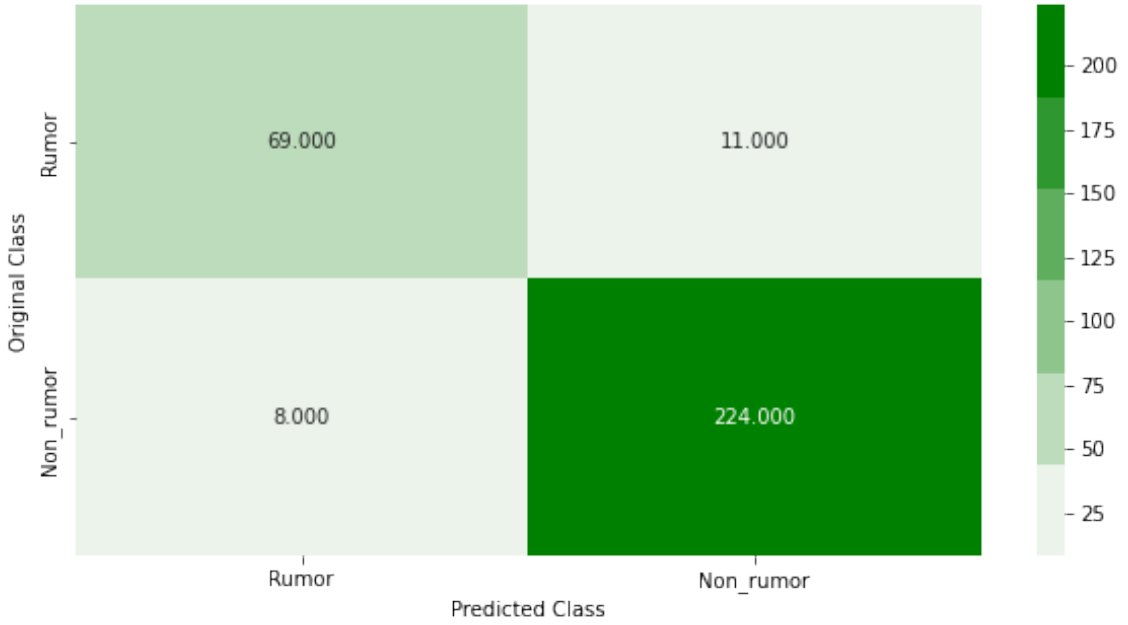


Figure 7.37: Confusion Matrix of CNN-Bidirectional GRU

ROC Curve

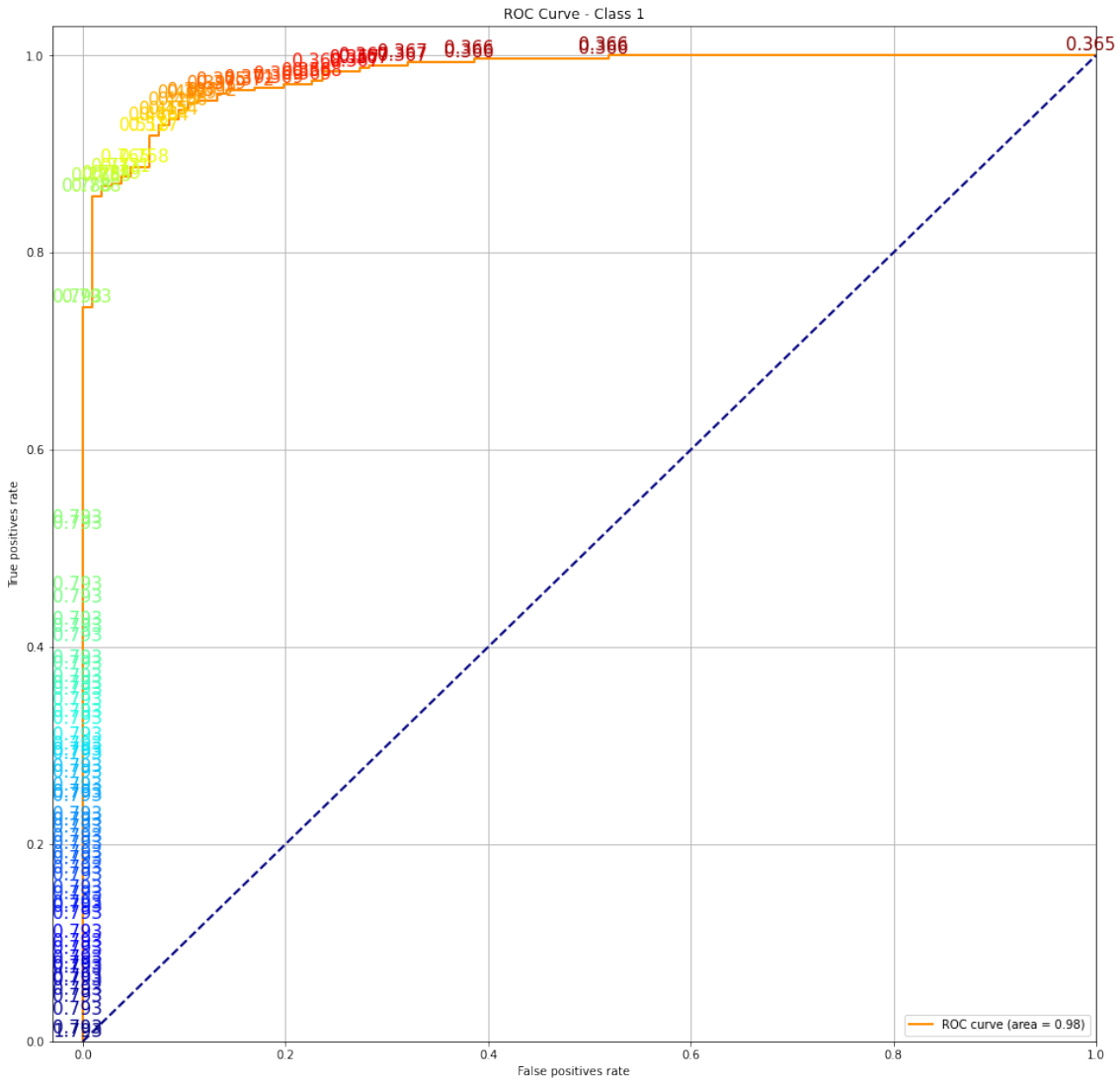


Figure 7.38: ROC curve for Baseline of CNN-Bidirectional GRU

This graph shows how a CNN-Bidirectional GRU's diagnostic ability changes as the discrimination threshold is changed.

Chapter 8

Conclusion and Future Work

8.1 Conclusion

In conclusion, Finding the solution to the text classification problem of detecting rumors in Bangla using Machine Learning algorithms and deep learning models for implementation is one of the most difficult tasks of Machine learning. Detecting rumors in the early stage, preventing them, and creating an automated system for rumor detection are the big challenges. Yet we take these challenges as our inspiration to develop the most efficient solution for rumor detection using different Machine Learning and deep learning approaches. This research is still at its primordial level. However, we will look into it by combining different cutting-edge machine learning algorithms and deep learning models, and we can fabricate satisfactory results.

8.2 Future Work

For finding the best performing techniques for rumor detection in Bangla we have already tested 10 machine learning classifiers and 6 deep learning models. However, due to obstacles related to data collection and limited resources and libraries in the Bangla language, we could only bring satisfactory results from 5 machine learning models and 3 deep learning models. So, our first future concern is enriching our dataset so that the usability and reliability of our models can increase. Moreover, we plan to release our dataset publicly in future and We will test more state of the art models with our enriched dataset. Secondly, we will develop a multi-modal Bangla rumor detection system. Lastly, we want to develop an early stage rumor detection system.

Bibliography

[1] Cooper, P. 2019. 28 Twitter Statistics All Marketers Need to Know in 2019. Retrieved February 18, 2019, from https://blog.hootsuite.com/twitter-statistics/

[2] L. A Badal, "Mob sets upon Hindu village in Rangpur over rumoured Facebook post" Dhaka Tribune November 11, 2017. [Online], Available: https://archive.dhakatribune.com/bangladesh/nation/2017/11/10/one-killed-angry-go-berserk-rangpur-hindu-village-facebook-status.[Accessed: May 22, 2022].

[3] L. A Badal, "53 arrested over attack on Rangpur Hindu houses" Dhaka Tribune November 11, 2017. [Online], Available: https://archive.dhakatribune.com/bangladesh/nation/2017/11/11/53-arrested-attack-hindu-houses. [Accessed: May 22, 2022].

[4] TBS Report 18 October and T. B. S. Report, "Mob burns down 20 Hindu homes in Rangpur," The Business Standard, 19-Oct-2021. [Online]. Available: https://www.tbsnews.net/bangladesh/crime/mob-burns-down-20-hindu-homes-rangpur-317398. [Accessed: May 22, 2022].

[5] "Factwatch of Ulab verified by IFCN," New Age | The Most Popular Outspoken English Daily in Bangladesh. [Online]. Available: https://www.newagebd.net/article/132364/index.php.[Accessed: May 22, 2022].

[6] "Factwatch of Ulab verified by IFCN," New Age | The Most Popular Outspoken English Daily in Bangladesh. [Online]. Available: https://www.newagebd.net/article/132364/index.php. [Accessed: May 22, 2022].

[7] Most popular social networks worldwide as of July 2021, ranked by number of active users https://www.statista.com/statistics/272014/global-social-networks-ranked-by-number-of-users/

[8] J. Hassan, "Quran burnt during the recent Tripura communal violence? no, viral image is old," The Logical Indian, 30-Oct-2021. [Online]. Available: https://thelogicalindian.com/fact-check/burnt-quran-31593. [Accessed: 22-May-2022].

[9] Ma, J., Gao, W., & Wong, K.-F. 2018b. Rumor Detection on Twitter with Tree-structured Recursive Neural Networks. Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), 1980–1989. https://aclanthology.org/P18-1184/

[10] M. Vohra and M. Kakkar, "Detection of Rumor in Social Media," 2018 8th International Conference on Cloud Computing, Data Science & Engineering (Confluence), 2018, pp. 485-490, doi: 10.1109/CONFLUENCE.2018.8442442.

[11] H. Bingol and B. Alatas, "Rumor Detection in Social Media Using Machine Learning Methods," 2019 1st International Informatics and Software Engineering Conference (UBMYK), 2019, pp. 1-4, doi: 10.1109/UBMYK48245.2019.8965480.

[12] S. Kwon, M. Cha, K. Jung, "Rumor detection over varying time windows", PloS one, 12(1), e0168344, 2017.

[13] G. Wenfeng, Z. Hong and C. Ruoyi, "A Study on Online Detection of microblog Rumors Based on Naive Bayes Algorithm," 2020 Asia-Pacific Conference on Image Processing, Electronics and Computers (IPEC), 2020, pp. 22-25, doi: 10.1109/IPEC49694.2020.9115171.

[14] C. M. Madhav Kotteti, X. Dong and L. Qian, "Ensemble Deep Learning on Time-Series Representation of Tweets for Rumor Detection in Social Media" 2020 arXiv:2004.12500.

[15] Kochkina, E.; Liakata, M.; Zubiaga, A. All-in-one: Multi-task learning for rumor verification. arXiv 2018, arXiv:1806.03713.

[16] Z. Yong, H. Yao and Y. Wu, "Rumors Detection in Sina Weibo Based on Text and User Characteristics," 2018 2nd IEEE Advanced Information Management,Communicates,Electronic and Automation Control Conference (IMCEC), 2018, pp. 1380-1386, doi: 10.1109/IMCEC.2018.8469468.

[17] http://tcci.ccf.org.cn/conference/2014/pages/page04$_t data.html$

[18] MA, Jing; GAO, Wei; MITRA, Prasenjit; KWON, Sejeong; JANSEN, Bernard J.; WONG, Kam-Fai; and CHA, Meeyoung. Detecting rumors from microblogs with recurrent neural networks. (2016). Proceedings of the 25th International Joint Conference on Artificial Intelligence (IJCAI 2016). 3818-3824. Research Collection School Of Computing and Information Systems. Available at: https://ink.library.smu.edu.sg/sis$_r esearch$/4630

[19] Z. Guo and J. Yang, "Rumor Detection on Twitter with Hierarchical Attention Neural Networks," 2018 IEEE 9th International Conference on Software Engineering and Service Science (ICSESS), 2018, pp. 783-787, doi: 10.1109/ICSESS.2018.8663917.

[20] Sepp Hochreiter and Jürgen Sch midhuber. Long short-term memory. Neural computation, 9(8):1735- 1780, 1997.

[21] Jing Ma , Wei Gao, and Kam-Fai Wong. Detect rumors in microblog posts using propagation structure via kernel learning. In Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), volume 1, pages 708-717, 2017.

[22] Sujana, Yudianto Jiawen, Li Kao, Hung-Yu. (2020).Rumor Detection on Twitter Using Multiloss Hierarchical BiLSTM with an Attenuation Factor.

[23] L. Li, G. Cai and N. Chen, "A Rumor Events Detection Method Based on Deep Bidirectional GRU Neural Network," 2018 IEEE 3rd International Conference on Image, Vision and Computing (ICIVC), 2018, pp. 755-759, doi: 10.1109/ICIVC.2018.8492819.

[24] Z. Zhou, Y. Qi, Z. Liu, C. Yu and Z. Wei, "A C-GRU Neural Network for Rumors Detection," 2018 5th IEEE International Conference on Cloud Computing and Intelligence Systems (CCIS), 2018, pp. 704-708, doi: 10.1109/CCIS.2018.8691263.

[25] H. Zhang, S. Qian, Q. Fang and C. Xu, "Multi-modal Meta Multi-Task Learning for Social Media Rumor Detection," in IEEE Transactions on Multimedia, doi: 10.1109/TMM.2021.3065498.

[26] J. Devlin, M. Chang, K. Lee, and K. Toutanova, "BERT: pre-training of deep bidirectional transformers for language understanding," in Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers). Association for Computational Linguistics, 2019, pp. 4171–4186.

[27] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings, 2015.

[28] L. Derczynski, K. Bontcheva, M. Liakata, R. Procter, G. W. S. Hoi, and A. Zubiaga, "Semeval-2017 task 8: Rumoureval: Determining rumour veracity and support for rumours," in Proceedings of the 11th International Workshop on Semantic Evaluation, SemEval@ACL 2017, Vancouver, Canada, August 3-4, 2017. Association for Computational Linguistics, 2017, pp. 69–76.

[29] A. Zubiaga, G. W. S. Hoi, M. Liakata, R. Procter, and P. Tolmie, "Analysing how people orient to and spread rumours in social media by looking at conversational threads," in PloS one, 2016.

[30] Hossain, M., Rahman, M., Islam, M. and Kar, S., 2022. BanFakeNews: A Dataset for Detecting Fake News in Bangla. https://doi.org/10.48550/arXiv.2004.08789

[31] M. G. Hussain, M. Rashidul Hasan, M. Rahman, J. Protim and S. Al Hasan, "Detection of Bangla Fake News using MNB and SVM Classifier," 2020 International Conference on Computing, Electronics & Communications Engineering (iCCECE), 2020, pp. 81-85, doi: 10.1109/iCCECE49321.2020.9231167.

[32] M. Z. H. George, N. Hossain, M. R. Bhuiyan, A. K. M. Masum and S. Abujar, "Bangla Fake News Detection Based On Multichannel Combined CNN-LSTM," 2021 12th International Conference on Computing Communication and Networking Technologies (ICCCNT), 2021, pp. 1-5, doi: 10.1109/ICCCNT51525.2021.9580035.

[33] T. Islam, S. Latif and N. Ahmed, "Using Social Networks to Detect Malicious Bangla Text Content," 2019 1st International Conference on Advances in Science, Engineering and Robotics Technology (ICASERT), 2019, pp. 1-4, doi: 10.1109/ICASERT.2019.8934841.

[34] Sepp Hochreiter and Jürgen Sch midhuber. Long short-term memory. Neural computation, 9(8):1735- 1780, 1997.

[35] Chollet, F. Deep Learning with Python, 1st ed.; Manning Publications Co.: Greenwich, CT, USA, 2017.

[36] Cho, K.; Van Merriënboer, B.; Gulcehre, C.; Bahdanau, D.; Bougares, F.; Schwenk, H.; Bengio, Y. Learning phrase representations using RNN encoder-decoder for statistical machine translation. arXiv 2014, arXiv:1406.1078.

[37] A. Guo and T. Yang, "Research and improvement of feature words weight based on TFIDF algorithm," 2016 IEEE Information Technology, Networking, Electronic and Automation Control Conference, 2016, pp. 415-419, doi: 10.1109/ITNEC.2016.7560393.

[38] K. E. N. N. E. T. H. W. A. R. D. CHURCH, "Word2Vec," Natural Language Engineering, vol. 23, no. 1, pp. 155–162, 2017.

[39] Joulin, Armand and Grave, Edouard and Bojanowski, Piotr and Douze, Matthijs and Jégou, Hérve and Mikolov, Tomas FastText.zip: Compressing text classification models https://doi.org/10.48550/arxiv.1612.03651

[40] https://scikit-learn.org/stable/modules/feature$_s election.html$

[41] https://scikit-learn.org/stable/modules/generated/sklearn.feature$_s election.$ $f_c lassif.html$

[42] https://scikit-learn.org/stable/modules/generated/sklearn.feature$_s election.$ $chi2.html$

[43] Wright, R. E. (1995). Logistic regression. In L. G. Grimm P. R. Yarnold (Eds.), Reading and understanding multivariate statistics (pp. 217–244). American Psychological Association.

[44] Cortes, C. and Vapnik, V. "Support vector networks", Machine Learning, 20 (3): 273-297 (1995).

[45] Webb, Geoffrey I., Eamonn Keogh, and Risto Miikkulainen. "Naïve Bayes." Encyclopedia of machine learning 15 (2010): 713-714

[46] Guo, G., Wang, H., Bell, D., Bi, Y. and Greer, K., 2003, November. KNN model-based approach in classification. In OTM Confederated International Conferences" On the Move to Meaningful Internet Systems" (pp. 986-996). Springer, Berlin, Heidelberg.

[47] Breiman, Leo. "Random forests." Machine learning 45, no. 1 (2001): 5-32.

[48] Luan, Y., Lin, S. (2019, March). Research on text classification based on CNN and LSTM. In 2019 IEEE international conference on artificial intelligence and computer applications (ICAICA) (pp. 352-355). IEEE.

[49] Zhou, P., Qi, Z., Zheng, S., Xu, J., Bao, H., Xu, B. (2016). Text classification improved by integrating bidirectional LSTM with two-dimensional max pooling. arXiv preprint arXiv:1611.06639

[50] Feurer, M., Hutter, F. (2019). Hyperparameter optimization. In Automated machine learning (pp. 3-33). Springer, Cham.

[51] Anguita, D., Ghelardoni, L., Ghio, A., Oneto, L., Ridella, S. (2012). The 'K'in K-fold cross validation. In 20th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN) (pp. 441-446). i6doc. com publ.

[52] https://drive.google.com/file/d/1cQ8AoSdiX5ATYOzcTjCqpLCV1efB9QzT/view

[53] https://github.com/sagorbrur/GloVe-Bengali

[54] https://drive.google.com/file/d/1CFA-SluRyz3s5gmGScsFUcs7AjLfscm2/view

[55] https://keras.io/api/callbacks/early$_s topping/$

[56] https://keras.io/api/layers/regularization$_l ayers/dropout$