Heaven's Light Is Our Guide

A Comparative Study Between Binary and Ternary Class Sentiment Analysis of Bangla Text Using Machine Learning, Deep Learning and BERT Based Models

A Project/Thesis Submitted in Partial fulfillment for the requirement of the degree of

Bachelor of Science
in
Electrical & Computer Engineering

by

Tarikul Islam Tamiti
Roll No. 1610001

to the

Department of Electrical & Computer Engineering
Rajshahi University of Engineering & Technology

October, 2022

A Comparative Study Between Binary and Ternary Class Sentiment Analysis of Bangla Text Using Machine Learning, Deep Learning and BERT Based Models

A Project/Thesis Submitted in Partial
fulfillment for the requirement of the
degree of

Bachelor of Science
in
Electrical & Computer Engineering

by

Tarikul Islam Tamiti
Roll No. 1610001

to the

Department of Electrical & Computer Engineering
Rajshahi University of Engineering & Technology

October, 2022

Acknowledgement

This thesis has been submitted to the Department of Electrical and Computer Engineering of Rajshahi University of Engineering & Technology (RUET), Rajshahi-6204, Bangladesh, for the partial fulfillment of the requirements for the degree of B.Sc. in Electrical & Computer Engineering. Thesis title regards to "**A COMPARATIVE STUDY BETWEEN BINARY AND TERNARY CLASS SENTIMENT ANALYSIS OF BANGLA TEXT USING MACHINE LEARNING, DEEP LEARNING AND BERT BASED MODELS** ".

First and foremost, I offer my sincere gratitude and indebted to my thesis supervisor Sagor Chandro Bakchy, Assistant professor, Department of Electrical & Computer Engineering who has supported me throughout my thesis with his patience and knowledge. I shall ever remain grateful to him for his valuable guidance, advice, encouragement, cordial and amiable contribution to my thesis.

I wish to thank once again Dr. Md. Shahidul Islam as the head of the Department of Electrical & Computer Engineering for his support and encouragement and also for providing all kind of laboratory facilities.

I would like to express my deep gratitude to my mother. He has always encouraged me during the research. I am also grateful to the administration of Rajshahi University of Engineering & Technology for providing a self-sufficient Under Graduate (UG) lab.

Finally, I want to thank the most important and the closest persons of my life and my parents for giving a big support to me.

Tarikul Islam Tamiti
Roll No. 1610001

October, 2022
RUET, Rajshahi

CERTIFICATE

This is to certify that the thesis entitled "***A COMPARATIVE STUDY BETWEEN BINARY AND TERNARY CLASS SENTIMENT ANALYSIS OF BANGLA TEXT USING MACHINE LEARNING, DEEP LEARNING AND BERT BASED MODELS* "** by Tarikul Islam Tamiti (Roll No. 1610001), has been carried out under my direct supervision. To the best of my knowledge, this thesis is an original one and has not been submitted anywhere for any degree or diploma.

Thesis Supervisor:

....
Sagor Chandro Bakchy
Assistant professor
Department of Electrical & Computer Engineering
Rajshahi University of Engineering & Technology

CERTIFICATE

This is to certify that the thesis entitled " ***TRANSLITERATION BETWEEN BRAILLE AND BANGLA LANGUAGE USING CONVOLUTIONAL NEURAL NETWORK***" has been corrected according to my suggestion and guidance as an external. The quality of the thesis is satisfactory.

External Member:

....
[**External Member (Bold)**]
[Designation]
Department of Electrical & Computer Engineering
Rajshahi University of Engineering & Technology

Declaration

This is to certify that this thesis work has been done by me and not submitted elsewhere for the award of any degree or diploma.

....

Tarikul Islam Tamiti

B.Sc. Engineering

Roll: 1610001

Department of Electrical & Computer Engineering

Rajshahi University of Engineering & Technology

Abstract

Natural Language Processing (NLP) is the domain of Artificial Intelligent (AI) in which the processing of Human Languages is carried out by computers. Through the process of evolution human brain is evolved in such a way that it can understand languages very easily at an early age. But the task of understanding human language is very difficult for AI. Over the last half century due to advancement of the technologies, AI has achieved a great deal of advancement. In some cases, AI can outperform humans in some tasks.

Unfortunately, the majority of the works of NLP have been so far conducted in a high-resource language like English, Hindi, and Mandarin. Although Bangla is the seventh most spoken language and among almost 300 million people all over the world, very little NLP research work has been carried out so far in Bangla. So, the Bangla language is categorized as low resource language in the NLP domain. Thus, there are many research opportunities in Bangla NLP as it is still in the early infant stages.

Sentiment Analysis (SA) is the domain of NLP in which by analysis a group of words, sentences, paragraph thoughts, expressions, and feelings are extracted. SA is very useful for understanding public views, and sentiments towards a particular topic. Social media is now the most popular media. People use various social media like Facebook, Twitter, and YouTube for expressing their sentiments. Most of them are in the textual format. Thus, analyzing textual data of social media is a very vital task for understanding the current trends and sentiments towards a particular topic.

In my thesis work, I have used the SentNoB dataset containing three classes (Positive, Negative, Neutral) of 16000 data samples. I have used five Machine Learning (ML), two deep learning (DL), and five Bidirectional Encoder Representations from Transformers (BERT) for Bangla SA. I have also shown a relative comparison between the two classes and three classes and found out that for increasing the number of classes the performance metrics accuracy, and F1 score decreases. For three classes I have achieved an F1 score of almost 86% and for two classes almost 69%.

Contents

Table of Content

List of Figures

List of Tables

List of Abbreviations

Short Form	**Abbreviations**
SVM	Support Vector Machine
NLP	Natural Language processing
CNN	Convolutional Neural Network
SA	Sentiment Analysis
DL	Deep Learning
AI	Artificial Intelligent
ML	Machine Learning
BERT	Bidirectional Encoder Representations From Transformers
SG	Skip Gram
KNN	K- nearest neighbor
TD	Temporal Difference
GPU	Graphical Processing Unit
Bi	Bidirectional
LR	Logistic Regression
RF	Random Forest
MNB	Multinomial Naïve Bayes
TF-IDF	Term Frequency-Inverse document frequency
LSTM	Long short-term memory
CBOW	Continuous Bag of Words
MLP	Multilayer Perceptron
GRU	Gated Recurrent Unit

Publications:

1. "Computer Vision Based Smart Computer Volume Control System with Hand Gesture Signatures using Google API" accepted on ICECTE2022.

Chapter 1
Introduction

Basically, this chapter provides the reasons for choosing this particular topic in the motivation subsection. The related works subsection summarizes the research works so far conducted in this Bangla SA arena. In the Thesis Objective subsection, the mission, vision, expected outcome, and applications of Bangla SA have been discussed. In the Dataset Description subsection, a summary of the SentNoB dataset has been discussed. In the Thesis Outline subsection, a brief discussion of the methodology of the ML, DL, and BERT models has been discussed.

1.1 Motivation

SA is one of the major research areas of computational linguistics. By extracting and analyzing a group of words, and sentences the election results can be estimated by polling, customer satisfaction, share market trends, grammatical error detection, word similarity measurement, etc. can be carried out [1]. Bangla is the fifth most spoken language all over the world. Around 230 million people all over the world use Bangla as their mother language. Thus, Bangla is one of the most spoken languages all over the world. Mostly in Bangladesh and some states of India Bangla is widely used [2].

Though Bangla is one of the most popular languages all over the world, unfortunately very little NLP research work has been conducted on Bangla so far due to a lack of resources, motivation, and the complex nature of the Bangla language. Thus, the Bangla language is termed a low-resource language [3]. As the development of the Bangla Language is still in the early stage thus there are opportunities of becoming a harbinger of many Bangla NLP research fields. Many companies and corporations in near future will invest heavily in this area for various purposes. As the Bangla NLP field is growing rapidly, the opportunities and resources will be huge in the future. Hopefully, many barriers of low resource language barriers like lack of labeled dataset are breaking due to the availability of the labeled dataset. Many researchers are now focusing low resource language like Bangla for their research topics enriching its content.

Besides, Bangla is the only language for which people sacrificed their lives for establishing it as the national language. On 21st February 1952 in Bangladesh this incident took place. Bangla is my mother tongue, I was also motivated to do my thesis work on Bangla.

1.2 Literature Review

Proportionately very few research works has been done on low resource language like Bangla due to lesser availability of resources like built in libraries and labelled datasets. Authors [4] developed BEmoC corpus of six emotion classes. Classify the data by training ML models like Logistic Regression (LR), Random Forest (RF), Multinomial Naïve Bayes (MNB), Support Vector Machine (SVM). For ML models features were

extracted by using Term frequency-inverse document frequency (TF-IDF) and LR model provided highest F1-score of 60.75%. Convolutional Neural Network (CNN), Bidirectional Long short term Memory(BiLSTM) DL models were developed by using Word2Vec and FastText for feature extraction. BiLSTM model provided highest accuracy of 56.94% when the feature is extracted using FastText word embedding. Among Transformer based models XLM-R provided highest F1 score of 69.73%.
Authors [5] extract YouTube comments by using YouTube API version 3.0. They showed a relative comparison between 3 and 5 classes labelled as Strongly and / or positive, negative, neutral by using LSTM, CNN, NB, SVM. They also provided result summary of dataset levelled as Emotion classes like Happy, sad etc. They used Continuous Bag of Words(CBOW) and Skip gram (SG) for vectorization i.e. feature extraction. They found out that accuracy of plain Bangla text is higher than Romanized Bangla.

Similarly authors [6] detected the offensive language of three Indian Tamil, Malayalam,Kannada language by using LR, SVM and Ensemble ML models. They have used LSTM and LSTM + Attention DL models. And XLM-R, m-BERT, Indic-BERT transformer-based models. They showed that transformer-based models provided superior performance with compare to ML and DL models.

Authors [7] employed ensemble majority voting for predicting five classes of hate speech. They showed that Ensemble voting outperformed all ML, DL and Transformer based models when these models were used alone to classify the hate speech.

Authors [8] implemented KNN, RF, NB, MLP, SVM, AdaBoost using different feature extraction techniques like unigram, bigram, trigram, char-n-gram on urdu dataset consisting of positive, negative and neutral class dataset. They also implemented variants of CNN, LSTM and GRU using FastText word embedding. They focused their research to construct language independent DL models for low resource languages.

1.3 Thesis Objective

- Choose a Bangla SA corpus with particular goals in mind – SentNoB dataset
- Choose a set of questions that will be answered after the analysis of results
- Preprocess the dataset by removing stop words, punctuations
- Learn specific language -Python
- Learn basics of ML, DL and transformers
- Choose ML, DL and transformer models for analyzing Bangla SA
- Train the models by finetuning the hyper parameters
- Evaluate and analyze the results
- Find out the answer of the questions
- Present the thesis outcome in a well-defined manner

1.4 Dataset description: Dataset used was developed by the authors [1] known as SentNoB dataset. The corpus was developed from extracting the comments of the most popular Bangla Newspaper 'Prothom Alo' and YouTube videos. SentNoB Dataset contains three classes – Positive (1), Negative (2), Neutral (0). The size of each comment is between 3 to 50 words. For Binary classification, positive and negative class and for ternary classification positive, negative and neutral class has been used. In Table 1.1 and 1.2 snapshot and summary of the SentNoB dataset is provided. Eight percent of the data samples were used as training purposes, ten percent for validation and ten percent for testing purposes for DL and BERT based models. Ninety percent data samples was used for training purposes and ten percent was used for testing purposes.

Table 1.1 : Snapshot of the SentNoB dataset

Data	Label
এক মাস সেহেরি খাইয়া রোজা রাহা সোজা	1
ভাই আমাদের মাতৃভুমিটা এত সুনদর	1
জানুয়ারীতে কেন ঐ সময় স্পেশাল কি	1
তোরা কখন কাচা রসুন চাবিয়ে খেয়ে দেখেচিস	2
হ্যাঁ হ্যাঁ হ্যাঁ হ্যাঁ জীবন টা বেদনা	2
বুমবুম মানে সামনে ধাক্কা খাবে এটা বেস্ট আমি হাসতে হাসতে শেষ	1
ভাই আপনার সব গুলো ভিড়িও আমার খুব ভাল লাগে	1
ভাই স্পেসিফিক লোকেশনটি জানতে চাইছি	1
হয়তো তার পদত্যাগ হবে অথবা কিছুই হবে না কিন্তু দেশ এতোটাই রসাতলে চলে গেছে যে এসব পাপ থেকে মুক্তি লাভ করা খুব কঠিন হয়ে পড়েছে	2
আপনার খাই দাই ডট্কমের গ্যাঙ্গ কারা	1
নেড়ে রা হেগে ছোচে না দুই হাতে খায় নাক খুঁটে হাত না ধুয়ে খাবে আবার ভিডিওতে দেখাবে কিন্তূ বলা যাবে না কারন রসুনের কোয়া সব কয়টা এক	2
এত লোভ দেখাচ্ছেন আপনি সে যাই হোক অবাক লাগছে থেকে অ্যাান্ত অ্যাান্ত পদ আপনি সাবার করবেন কেমন করে	0
এরম একটি রুপালী অমৃত যদি কেউ পাঠাইত বাংলার এইপারে আমারে	1

ধন্যবাদ আপনাকে অসাধারণ সুন্দর ভিডিও উপহার দেওয়া জন্য সাথে আপনার সুন্দর উপস্থাপনা আমাদের কে আরও বেশি মুগ্ধ করে	1
রাইস থেকে সাদা ভাত অথবা পোলাউ অথবা ভুনা খিচুড়ি অথবা চিকেন বিরিয়ানি অথবা মাটন বিরিয়ানি অথবা তেহারি	0

Table 1.2: Summary of the SentNoB dataset

Class	**Instances**	**Percentage**	**Label**
Neutral	5709	36.3%	0
Positive	6401	40.8%	1
Negative	3609	22.9%	2

Chapter 2

Methodology

In this chapter Data Preprocessing, Tokenization, Train test validation data split, feature extraction, ML models, DL models, BERT models, and the summary of the proposed methodology discussed. Overall methodology of this thesis work can be summarized in the following figure 2.1.

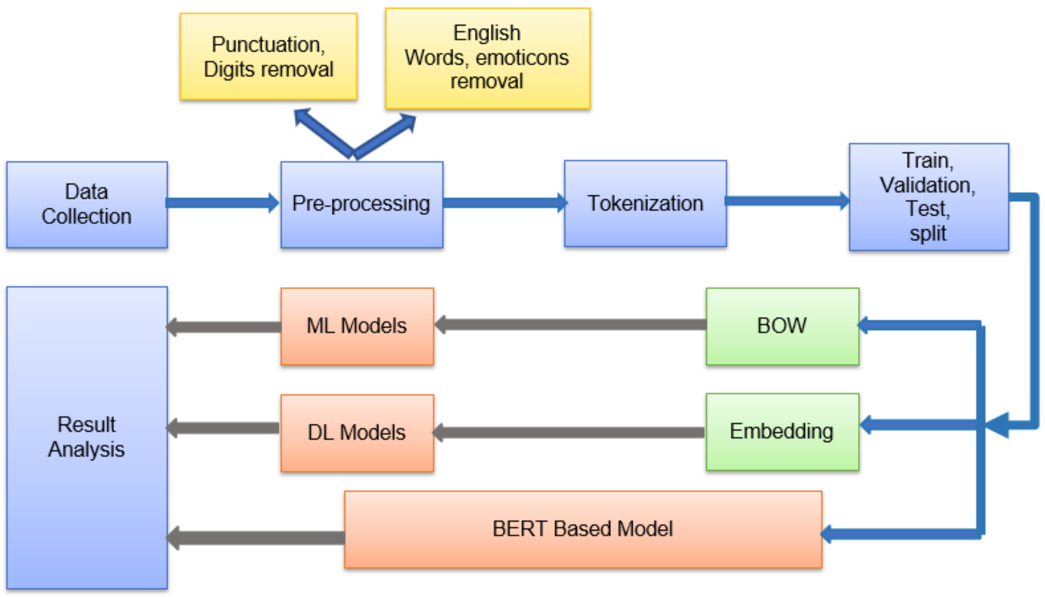


Figure 2.1: Schematic view of the Methodology

2.1 Data Preprocessing: Data contains many words, symbols, numeric values, English words that contain very little or no significant meaning for Bangla SA. Whereas, the adjectives are the main keywords which contributes towards Bangla SA. Thus, removing the unnecessary words and focusing on main keywords improves the overall performance of the models. For these reasons, after data collection, the is cleaned by removing emoticons, symbols and pictographs, transport and map symbols, flags (ios), latin, general punctuations, emoji, English patterns, punctuations by using regular expressions (regex). The overall performance of the models depends on a large extent on the preprocessing.

2.2 Tokenization: Tokenization is the process by which texts are broken down into smaller pieces. These smaller pieces are known as tokens. Tokenization can be done by using python’s split function, regular expression, NLTK library, spaCy library, keras framework, Gensim library. I used BertTokenizer from pretrained function of the transformer for tokenization. The ‘csebuetnlp/banglabert’ pretrained model from transformer is used for tokenization. If the tokens were not present in the vocabulary then ‘[UNK]’ tokens were used. If the words were spitted from the middle then before each sub word ‘##’ token was padded. As I have used BERT tokenizer for tokenization, the words are tokenized as per the pretrained model. Then I have added a new column in the panda’s data frame and store the tokens which are later used for feature extraction by various models. Tokens helps the feature extraction process. Tokens speed up the process of understanding the context of the words thus, classification. One instances of the data sample was taken in the following figure 2.2 and after applying the tokenization the result was also given.

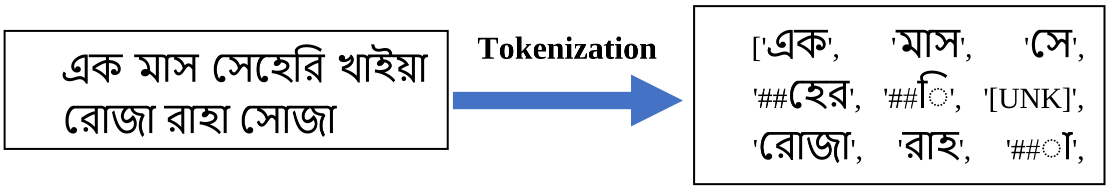

এক মাস সেহেরি খাইয়া রোজা রাহা সোজা
Tokenization
['এক', 'মাস', 'সে', '##হের', '##ি', '[UNK]', 'রোজা', 'রাহ', '##া',

Figure 2.2: Example of tokenization

2.3 Feature Extraction: It has been proven that models perform better when they work with numerical inputs rather than textual inputs [6]. ML and DL models performs well when both the inputs and outputs are fixed lengths. Thus, for getting a better performance the textual data is converted into numerical representation. Various types of feature extraction are available. I used BOW and TF-IDF (unigram) for ML models’ feature extraction. But for DL models I used keras embedding layer. Nowadays Glove, Word2Vec, FastText word embedding are getting very popularity. For choosing the feature extraction it should be kept in mind that the data after feature extraction should not get too sparse. Rather than giving textual data to the input of the models, numerical representation of the data is provided to the models which improves the overall performance of the models. Feature extraction also called as feature encoding or vectorization.

2.3.1 BOW: Bag of word is the numerical representation for the ML models. BOW representation is one of the easiest, simplest and understandable method for numerical representation of the words. BOW counts the frequency of the any word in the document whether it is present in the document or not. But the main disadvantage of BOW is that it didn’t capture the structure, syntax or the context of the words’ surrounding. Besides, BOW results very sparse vectors which contains a large number of zeros. In BOW, the words that present many times starts to dominate the but they may not contain much information that helps in the classification. BOW consists of two main parts. Corpus or vocabulary of the known words and frequency of the word in a sentence. BOW is like histogram analysis of textual data similar to pixel histogram of digital image processing. For BOW, it is assumed that sentences or paragraphs having similar meaning will contain similar words. The complexity of BOW depends on the rules imposed for collecting the vocabulary and how to count the frequency of the number of times a particular word present in a sentence or paragraph. Sometimes a group of words are used known as grams for capturing more information and reducing the size. I have used BOW for feature extraction i.e. vectorization of tokens for feeding them as inputs for ML models.

2.3.2 TF-IDF: Term Frequency-Inverse document frequency is another type of numerical representation of words. So far, the problems arise along with the BOW is partially removed by TF-IDF. In BOW the words which present more than other words known as frequent words, starts to supersedes other words. But the frequent words may not contain necessary information which helps in classification. Therefore, this effect of the frequent words must be removed by using TF-IDF. TF-IDF give emphasis on the rare words. TF-IDF is the weighted multiplication of two terms, Term Frequency and Inverse Document Frequency. TF is the number of times a particular word present in the document. IDF is the natural logarithm of the ratio of Total Number of Documents to the number of documents containing that term. IDF measures how rare a particular word is. Thus, TF-IDF emphasizes the rarity of a word.

$$TF = \frac{Frequency\ of\ a\ term\ in\ a\ particular\ document}{Total\ Number\ of\ terms\ in\ that\ particular\ documents} \ldots\ldots\ldots..(1)$$

$$IDF = \log\left(\frac{Total\ number\ of\ documents}{Number\ of\ document\ term\ containing\ that\ term}\right) \ldots\ldots\ldots..(2)$$

$$TF\ IDF = TF\ \times IDF\ \ldots\ldots\ldots(3)$$

Words can be grouped together, known as grams. Grams are used for capturing more context information and reduce the size. But analyzing higher grams require more computational power. Here, I have used TF-IDF unigram for input to ML models.

2.4 Train, Test, validation split:

Usually models get overfitted during training, provides excellent performance during training and performs badly when evaluated with unseen data. This phenomenon is known as overfitting. The overfitting phenomenon is undesirable and we want our models to be affected as little as possible by this overfitting issue. For this we need a method to evaluate how our model is performing. This is done to measure the generalization of the model. For this reason, the data need to split into train, test and validation data for the purpose of avoiding overfitting and evaluation of the model successfully. 90% of the data is used for training purpose and 10% of data is used for testing purpose for ML models. Whereas for DL and BERT based models, 80% of data is used for training purpose, 10% of data is used for validation purpose, 10% of data is used for testing purpose. Train data is mainly used to fit or train the models. Parameters of the models are adjusted, known as fine tuning for training models as required so that models provide better performance. Validation data is used to measure the measure of the models performance on the training data. Validation data is used for feature and threshold cut-off selection and unbiased evaluation of the model. Test data are separated and there is no way that models will able to see the test data. Thus, test data is used to measure how well a model will perform on unseen data. I have used scikit learn's train_test_split method twice to split the data into train, test and split data. When using train_test_split method first time, we get the validation data and the remaining data was used again on train_test_split for getting train and test data. The split into train, test and validation data will be random. Thus, the performance of the models may vary significantly. For getting fixed split, we use the random_state parameter of the train_test_split method to a fixed value so that each time the program executes, the split occurs same each time.

2.5 Machine Learning models: Prior to Machine learning, all the algorithms, programs, computational tasks require to provide inputs, a set of well-defined rules performed on the inputs. Some operations are performed based on the well-defined rules, the algorithms/ programs provide the output. But the main challenge is to the finding of the well-defined rules. For a very complex system with thousands or millions of parameters involved it is almost impossible for finding out the well-defined rules.
Machine learning revolutionized the domain of computer science by the machines learns it selves from the data provided to it. Here input and output are both given to the model during the training process. The models find out the rules by themselves during this training period. After that, only the inputs are provided, the models provide the output based on the rules it learns during training. By this Machine learning approaches analysis of the very complex systems with large number of parameters involved become very easy. But the main problem of the Machine learning approach is that, the ML models require very large amount of data for providing better performance. Broadly, ML models can be classified into three types. Supervised, Unsupervised and Reinforcement are three main types of ML models. I have used GridSearchCV method to find the best parameters of the models for getting the better performances of the models. Here various values of the parameters are given. GridSearchCV method tries various combination of the parameters automatically and finds out the best combination of the parameters for each models which provides the best classification performance.

2.5.1 Supervised Learning: In supervised learning, the input along with the desired output, also known as target is provided to the model during the training process. After the training, models are able to predict the target value for the given inputs with a desired level of performance like accuracy. Regression, Decision Tree, Random Forest, KNN, Logistic Regression are the examples of supervised learning.

2.5.2 Unsupervised Learning: Unsupervised learning is mainly clustering. Here no output or target value is provided. Algorithms and models only predict the number of clusters or groups represent in the input data by finding out the hidden patterns among the data. Sometimes implementation of clustering i.e. unsupervised learning make sense because always the number of class may not be known. In that case unsupervised learning is applicable. K means clustering, Hierarchical clustering, Apriori algorithm are the examples of unsupervised learning.

2.5.3 Reinforcement learning: Reinforcement type learning is a special type of Machine Learning in which models learn from the environment. Models are introduced in the environment. From observing the environment models learn and predict. Here for each correct prediction the models are rewarded and for making incorrect decision, the models get punished by some penalty score. Thus, the correct decision-making is being reinforced to the model. Temporal Difference (TD) , Markov Decision Process are the examples of reinforcement learning.

Among the different types of ML algorithms, I have used SVM, Decision Tree, Logistic Regression, Multinomial Naïve Bayes, K Nearest Neighbor, Stochastic Gradient Descent algorithm. A brief discussion of the algorithms is provided in the following sections:

2.5.4 Logistic regression: Logistic regression is a type of supervised learning in which inputs are given as independent variables. The output of the logistic regression is a categorial dependent variable. Logistic regression provide output which is probabilistic in nature, i.e. the value of the output lies between 0 to 1. This Logistic regression used in computer science for mainly classification problems. The logistic regression models are trained on input data such that the model is fitted on 'S' shaped curved of the logit function. The logit/sigmoid functions are widely used function in other domain of science. But in Machine learning arena, it is mainly use to map the prediction of the model into probabilities based on the inputs provided to it. No matter what is given as input to the logit/ sigmoid function, the output of the function will always between 0 to 1. The threshold value of the logit function should be selected carefully. Above the threshold value, the output of the logit/sigmoid function should be 1 and below the output is equal to 0. In figure 2.3, output of the logit function is shown.

Threshold value output of the logit/ sigmoid function should be 0. For the successful operation of the Logistic regression model it should be assured that inputs must not have multi-collinearity and the output should categorical. Logistic Regression can be classified into three types. They are discussed below:
Binomial Logistic Regression: The output of the Binomial Logistic Regression can be only two types, 0 and 1.
Multinomial Logistic Regression: The output of the multinomial Logistic Regression can be more than two types i.e. three or more types. The output of this type of Logistic Regression is unordered type.

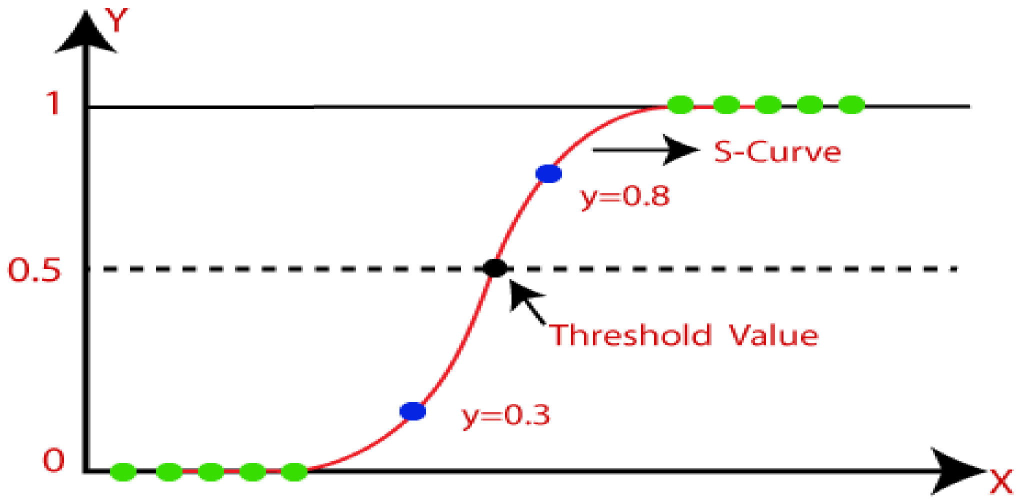


Figure 2.3: Logit/ sigmoid function [9]

Ordinal Logistic Regression: The output of the ordinal Logistic Regression can be more than two types i.e. three or more types. But the main difference between Multinomial and Ordinal Logistic Regression is that in Ordinal Logistic Regression the output is ordered type, but in Multinomial Logistic Regression is unordered type.

I have implemented Logistic Regression by using scikit learns' linear model LogisticRegression. I set the value of C hyperparameter to different values (1,5,10,25). Hyperparameters are actually used to indicate the models how to choose parameters. C hyperparameter is used for the regularization purpose. The value of C must be positive float type value. High value of C indicates to assign high weight to training data, overlook outliers (extreme datapoints which leads to overfitting issue).

2.5.5 Support Vector Machine: Support Vector Machine (SVM) provides very accurate result with less computational power. SVM mainly used for classification tasks. SVM finds out the best hyperplane to accurately classify the data samples. The classes can be classified by many hyperplanes, but SVM finds out the best hyperplane so that the margin between the two extreme datapoint is maximized. The dimension of the hyperplane depends on the number of classes present in the dataset. If the dataset has two classes then the hyperplane will be a straight line. If the dataset has three classes then the hyperplane will be a plane. SVM can be classified into two types. Linear SVM and non-linear SVM. In the linear SVM, the data points can be classified using a straight line. That means the hyperplane for linear SVM is a straight line. In figure 2.4, straight line decision boundary is drawn to separate two classes of data. But for non-linear SVM, the data points can not be classified using a straight line. For this case, the data points are taken into higher dimensions, then the data points become linearly separable. Then by using a straight line the data points can be separated. For getting a optimal hyperplane, the constraint optimization problem must be solved by using Lagrange Multiplier and Karush–Kuhn–Tucker (KKT) conditions. I have implemented the SVM from scikit learn library. The parameters of the model are assigned with values as per the documentation of the scikit learn library. 'C' parameter was assigned with 1,10,20. 'C' parameter is a regularization parameter and the strength are inversely proportional

to the assigned value of 'C'. For each error the penalty is a squared 'l2' penalty. 'kernel' parameter assigned with 'rbf', 'linear' and 'sigmoid'. 'Kernel' parameter specifies the type of function used to convert the data points into higher dimensions from lower dimensions.
'gama' parameter assigned with 'auto' and 'scale' value. 'decision_function_shape' assigned with 'ovo' and 'ovr'. One-vs-rest('ovr') or one-vs-one('ovo') will be the return policy, is defined by the 'decision_function_shape' parameter.

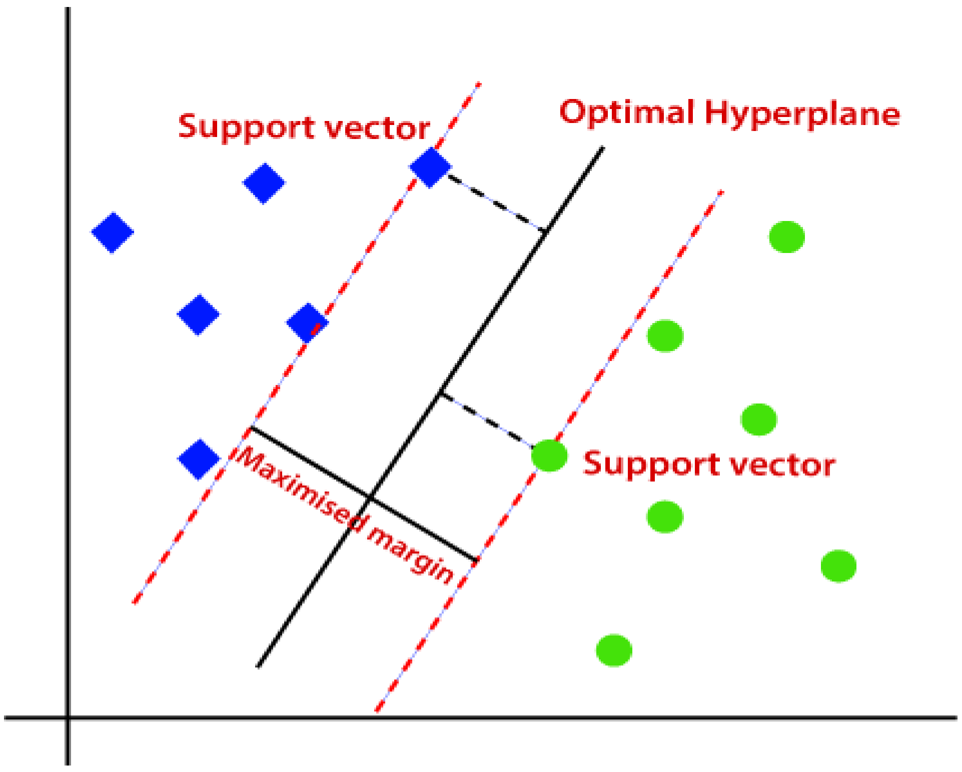


Figure 2.4: Selection of Hyperplane on the basis of support vector [10]

2.5.6 Decision Tree: Decision Tree is the tree-structured, non-parametric classifier used for classification. Decision tree is in the domain of supervised learning. In the tree structured classifiers there are internal nodes, branches and leaf node. Internal nodes are the representation of the features, branches are the representations of decision rules and leaf nodes is the representation of the results. In the decision tree, decision and leaf node remain present. Decision nodes makes the decision based on some pre-defined rules or conditions. Multiple branches may be branched out from the decision nodes. But the leaf nodes are the results of the decision nodes. Leaf nodes don't further branch out. The decision nodes are formulated based on the features. Decision tree is the graphical or pictorial representation of the all possible outcomes of the classification problems. As the name suggests, decision tree is basically a tree structure, beginning with root node. For developing the decision tree, Classification and Regression Tree algorithm (CART algorithm) is deployed. Decision tree raises a question, whose answer can be provided with simple 'Yes' or 'No'. Based on the answer being provided, tree is branched out into sub-tree. Decision tree copies the pattern of thinking of the human brain. Because of its tree-like structure, it is very easy to understand. Decision tree considers all possible scenarios and require less data cleaning. The complexity of Decision tree is logarithmic. Decision tree can be used for both categorical and numeric data. Decision tree can be used for multiclass classification problems. Some of the drawbacks of the decision tree are the formulation of very complex tree i.e. overfitting, stability issue for very small changes the structure of the tree might be changed. In figure 2.5 the process of classification by decision tree is shown.

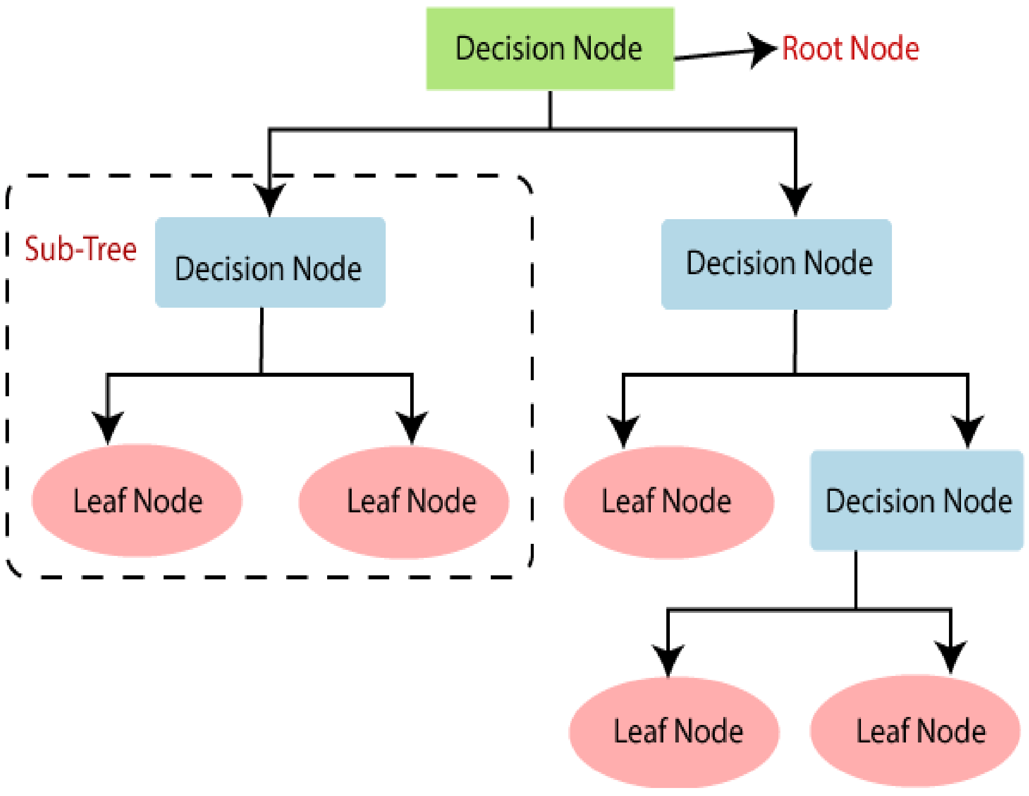


Figure 2.5: Branching out of the Decision Tree [11]

'criterion' parameter is used for the measurement of the splitting done at each node. The value of the 'criterion' parameter is assigned to 'entropy'. 'max_depth' is used for indicating how deep the nodes will be expanded. The value of the 'max_depth' parameters assigned as 125,500,700,900. 'min_samples_split' indicates the lowest number of samples required to split internal node. 'min_samples_split parameter is assigned with 15,55,95 value. 'max_features'
Defines the number of features should be considered before searching for best split. The value assigned to 'max_features' are 'sqrt','log2'.

2.5.7 Multinomial Naïve Bayes: Naïve Bayes is a probabilistic method for classification. That means Naïve Bayes deals with the probability of the data class belonging to a particular class. There are two types of Naïve Bayes. Gaussian and

Multinomial Naïve Bayes are the two types of Naïve Bayes. Multinomial Naïve Bayes can classify non-numeric data. The computational complexity of Multinomial Naïve Bayes is much lower, classification can be done by small datasets. Continuous retraining is not possible. Multinomial Naïve Bayes provides easy implementation, robust, very accurate. However, Multinomial Naïve Bayes did not provide any set of independent predictors resulting a model. Multinomial Naïve Bayes uses statistical approaches for classification. Multinomial Naïve Bayes is the alternative representation of semantic analysis, providing simplified solution of textual data classification. Multinomial Naïve Bayes classify the textual data by decision function. For preventing cold starts problems and earlier convergence semi supervised learning algorithms are used. Multinomial Naïve Bayes also has been implemented using scikit learn library. Alpha parameter is used as controlling parameter for Multinomial Naïve Bayes. Alpha parameter is additive smoothing parameter. There are two types of smoothing. Laplace and Lidstone are two types of smoothing parameter. For zero value of alpha, no smoothing will be conducted. I have assigned .10, .25, .40, .75, .90 values for Alpha parameters.

2.5.8 K Nearest Neighbor: K Nearest Neighbor is a type of supervised learning. K Nearest Neighbor mainly employed for classification problems. K Nearest Neighbor consider K numbers of neighboring data points for classification, thus it is named as such. K Nearest Neighbor is a versatile algorithm and also used for completing missing data by imputing methods and resampling. This algorithm uses instance-based learning where instead of adjusting weights from training data, it uses the whole spectrum of training instances. This technique is also called lazy learning. KNN is also non-parametric, that means no predefined form of mapping function is used.

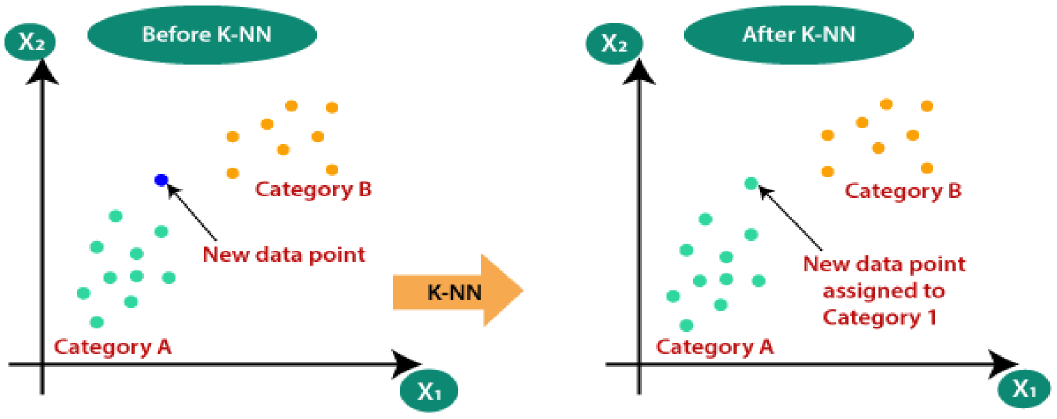


Figure 2.6: Implementation of KNN for new data point [12]

K Nearest Neighbor compares similarity between new data sample with it's K nearest neighbor by some distance matrix like Euclidian distance and classify that new data sample to the most similar class. In figure 2.6 the new data point is similar with Category A. Thus, it is classified as Category A. The main problem of selecting the particular

value of K, where K is the number of neighboring data points to consider, based on which the similarity is measured. If the value of K is low then the effects of the outliers i.e. extreme data points may lead to the misclassification problems. For higher value of K leads to more computational complexity. I have implemented KNN by using scikit learn library. Here, ‘n_neighbors’ parameters was used for the number of neighbors selection. ‘n_neighbors’ parameters was assigned with 3,4,5,6,7 value.
‘weights’ parameters were used for weight function. I have assigned ‘uniform’, ‘distance’ value for ‘weights’ parameters. ‘uniform’ weight means all the neighborhoods are weighted equally. Assigning ‘distance’ value means weight points are inverse of their distance.‘algorithm’ parameter computes the value of nearest neighbors. ‘ball_tree’, ‘kd_tree’ are assigned for generalization of N-point problems. ‘brute’ value uses a brute-force search algorithm.

2.5.9 Stochastic Gradient Descent (SGD): Gradient Descent is the type of algorithm in the domain of generic optimization. Gradient Descent finds optimal solutions of various problems. The parameters are tuned iteratively for minimizing the cost function. Learning rate is a very important hyperparameter. If learning is too low, then convergence will take many iterations i.e. time. Whereas, if the learning rate is too high, then the solution may cross over the optimal value. Batch, Stochastic, Mini-Batch Gradient Descent are three types of Gradient Descent algorithms. I have used Stochastic Gradient Descent from scikit learns library. The term ‘stochastic’ means random probability linked with a process. In Stochastic Gradient Descent the random selection of data samples was done instead of whole data set. The size of data samples to be considered during each iteration for the calculation of gradient is called ‘batch’. If we consider whole dataset for the calculation of the gradient, then the calculation will become very difficult. This problem is solved by Stochastic Gradient Descent, in which a portion of dataset, known as ‘Batch’ is considered for the calculation of the gradient. The path taken by Stochastic Gradient Descent to reach the global minima is noisier than other gradient descent algorithm. But as the time require to reach the global minima is comparatively shorter, the nosier path doesn’t matter. In figure 2.7 the convergence of SGD is shown. From scikit learns library, I have implemented Stochastic Gradient Descent. By tuning hyperparameters, I have trained by model. ‘loss’ parameter was assigned with ‘log’ value. ‘penalty’ parameter was assigned with ‘l2’,’l1’,’elasticent’ value ‘alpha’ parameter was assigned with .0001, .0005, .0009, .0012 value.

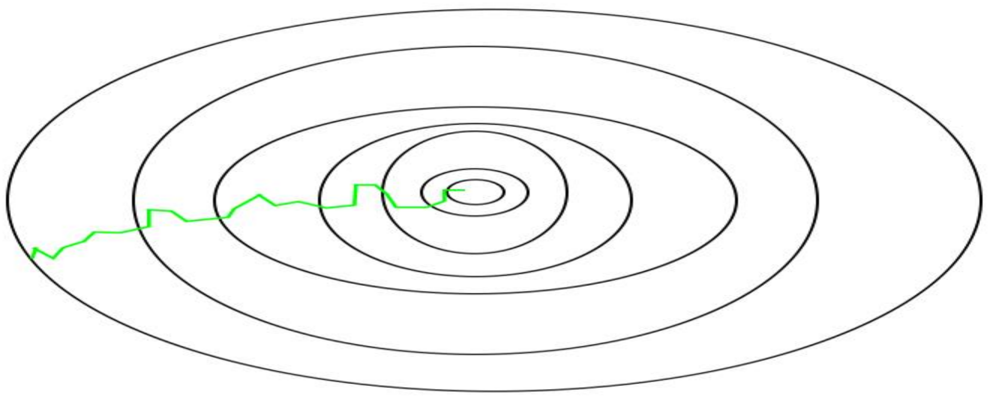


Figure 2.7: Path taken by Stochastic Gradient Descent to reach global minima

2.6 Deep learning models: Deep learning is a subset of Machine learning. In deep learning, the neurons are the basic processing units. The main disadvantages of the machine learning models are greatly reduced by using deep learning models. In machine learning models, if the models make an incorrect decision then there are no ways of correcting the mistake by the model itself. Whereas, in deep learning, by the backward error propagation, the weights are adjusted such that the error will be minimized. Machine learning models learn from the data during training, parse data, make prediction based on its learning from training data. Whereas, deep learning models mimics the human neural networks by creating an 'Artificial Neural Network'. There are various layers in the deep learning models. Both Machine and deep learning models is in the domain of artificial intelligence. In figure 2.9 the relationship between AI, ML and DL is provided.

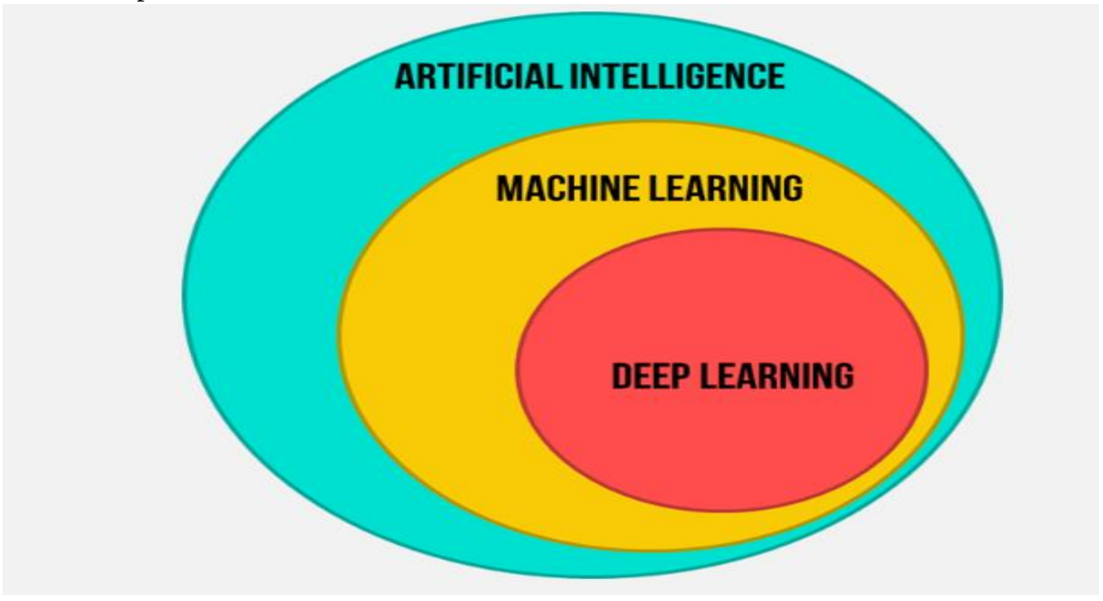


Figure 2.8 Relationship between Artificial Intelligence, Machine Learning, Deep learning [13]

Deep learning generates artificial neural network and perform complex computation on large dataset. An artificial neural network has similar structure like human brain and the basic processing units of it are artificial neurons, also known as nodes. These nodes are stacked on each other forming connected layers. Three types of layers present in a neural network. These are input, hidden, and output layer. Input data is provided to each nodes of the input layer. The nodes multiply the input with the weights, add the multiplication results, adds the bias value. Then by using a nonlinear function or known as activation functions are applied to determine whether the value is greater than the threshold value. If the value is greater than threshold value, then the neuron will be activated. If the value is less than the threshold value, then neuron will not be activated. In figure 2.9, basic Artificial Neural Network is shown.

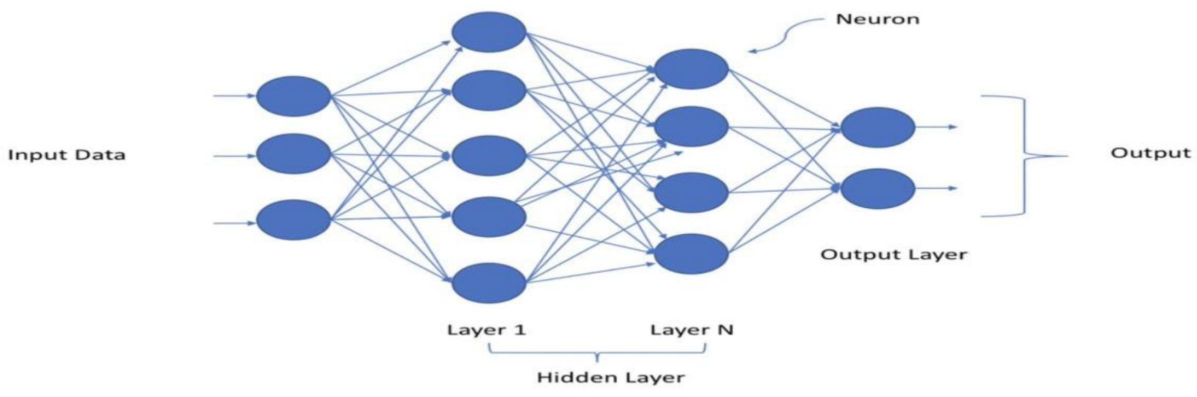


Figure 2.9: Artificial Neural Network [14]

Deep learning models represent self-learning and rely on artificial neural network which mimics the way brain computes. During training process, Deep learning models extract features from input data, similar objects are grouped together, the patterns are found out. There are many Deep learning models suitable for particular tasks. No network is considered as perfect. I have used Convolutional Neural Network (CNN), and a concatenation of Bidirectional Long Short Term Memory (BiLSTM) and Convolutional Neural Network (CNN), called as Conv-LSTM.

2.6.1 Convolutional Neural Network: Convolutional neural network, often known as CNN, a subset of deep learning is developed to process ordered arrays of data, such as representations. It is frequently used to analyze visual images since it can identify and categorize certain characteristics from images. CNN identifies the design components in the input image, such as lines, gradients, circles, or even eyes and features. This property makes convolutional neural networks highly reliable. Convolutional, polling, and fully connected layers are the components of a convolutional neural network (FC). When we process an image, we apply filters, and each of them produces a product we refer to as a feature map. To extract characteristics from an image, CNN utilizes different filters. Features are extracted using the convolution and pooling layers, while classification is done using the fully connected or dense layers. filters are formed in the convolution layer. Convolution is a mathematical procedure carried out by sliding these filters across the input picture and feature maps are produced. Convoluted feature map size is reduced by pooling layer. There are different types of pooling techniques. In Max Pooling, the biggest component is picked from the feature map. The average of the components in a predetermined sized Image portion is determined via average pooling. The FC layer receives a flattened version of the input picture from the preceding levels. Numerous mathematical operations are performed, along with categorization. Different activation functions are used to add non-linearity to the networks such as ReLU, tanH, sigmoid function, softmax etc. In figure 2.10, basic architecture of CNN model is shown.

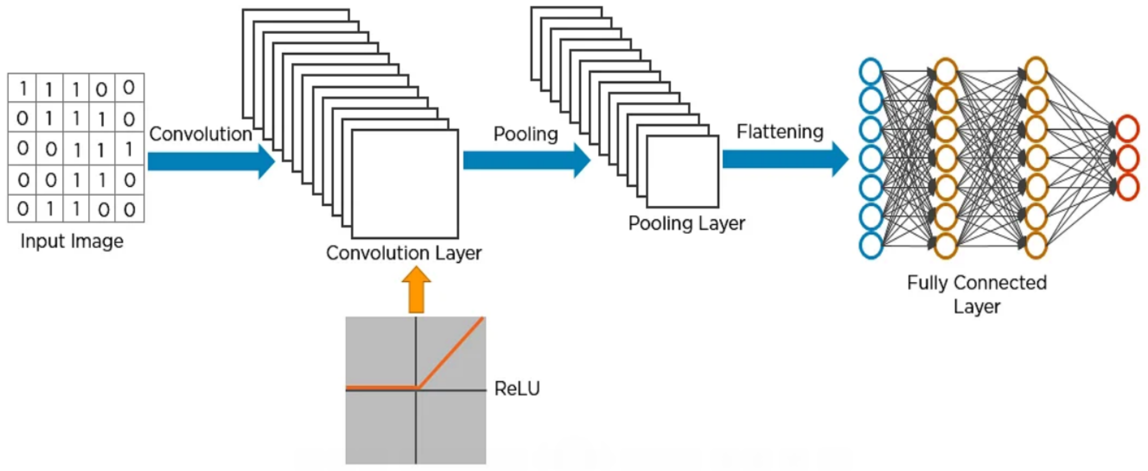


Figure 2.10: Basic Architecture of CNN model [15]

Although CNN model is best suited for image processing, CNN model can also be used in Natural Language processing if the feature extraction can be done properly. I have implemented a CNN model. In which, the textual data is provided to the input embedding layer. By using keras embedding, the input textual data is converted into numeric data. Then, the numeric data is provided as input to one dimensional convolution layer. After the convolution operation, the next layer input is the output of this convolution layer. The next layer is global_max_pooling layer. Here, the largest among the nonboring pixel values are chosen. Then the next layer is dropout layer. Dropout layer is used to drop the connection between some neuron to next layer to void the contribution of some neuron to reduce the effect of overfitting. Then the output of this layer is fed to the input as next dense layer. Dense layer is a fully connected layer. Then the following layers are two pairs of dropout and dense layer. Then the output is flattened in the flatten layer. Flatten layer converts the output into one dimensional array. Then at last output is taken from the fully connected dense layer. The output is provided as probability. For binary classification the sigmoid function is used, whereas for ternary classification, the softmax function is used to select the highest probability class. The architecture of implemented CNN model is provided in figure 2.11. In figure 2.12 the parameters of CNN model is shown.

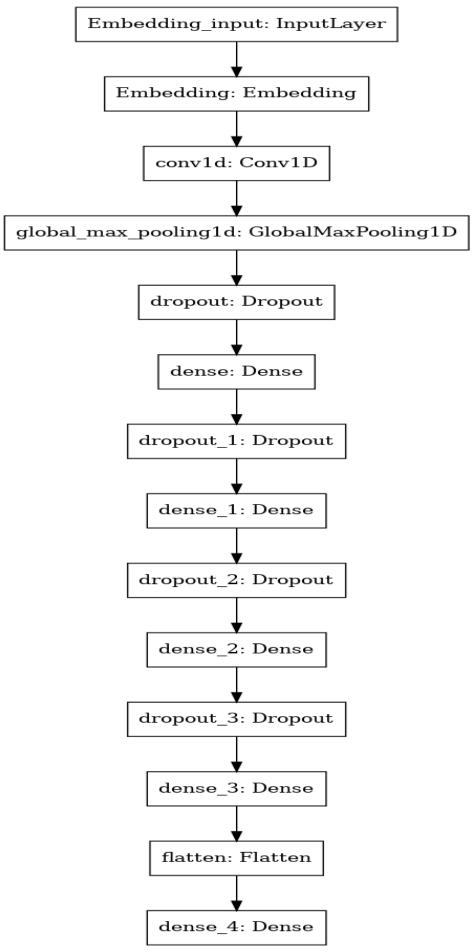
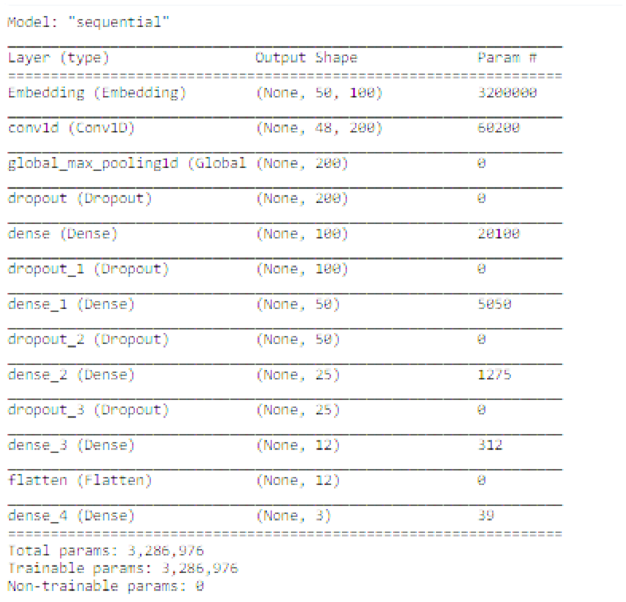


Figure 2.11: Implemented CNN model architecture



```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
Embedding (Embedding)        (None, 50, 100)           3200000
_________________________________________________________________
conv1d (Conv1D)              (None, 48, 200)           60200
_________________________________________________________________
global_max_pooling1d (Global (None, 200)               0
_________________________________________________________________
dropout (Dropout)            (None, 200)               0
_________________________________________________________________
dense (Dense)                (None, 100)               20100
_________________________________________________________________
dropout_1 (Dropout)          (None, 100)               0
_________________________________________________________________
dense_1 (Dense)              (None, 50)                5050
_________________________________________________________________
dropout_2 (Dropout)          (None, 50)                0
_________________________________________________________________
dense_2 (Dense)              (None, 25)                1275
_________________________________________________________________
dropout_3 (Dropout)          (None, 25)                0
_________________________________________________________________
dense_3 (Dense)              (None, 12)                312
_________________________________________________________________
flatten (Flatten)            (None, 12)                0
_________________________________________________________________
dense_4 (Dense)              (None, 3)                 39
=================================================================
Total params: 3,286,976
Trainable params: 3,286,976
Non-trainable params: 0
```

Figure 2.12: Implemented CNN model parameters

In figure 2.13, the accuracy curve of implemented CNN model is shown on validation and test data.

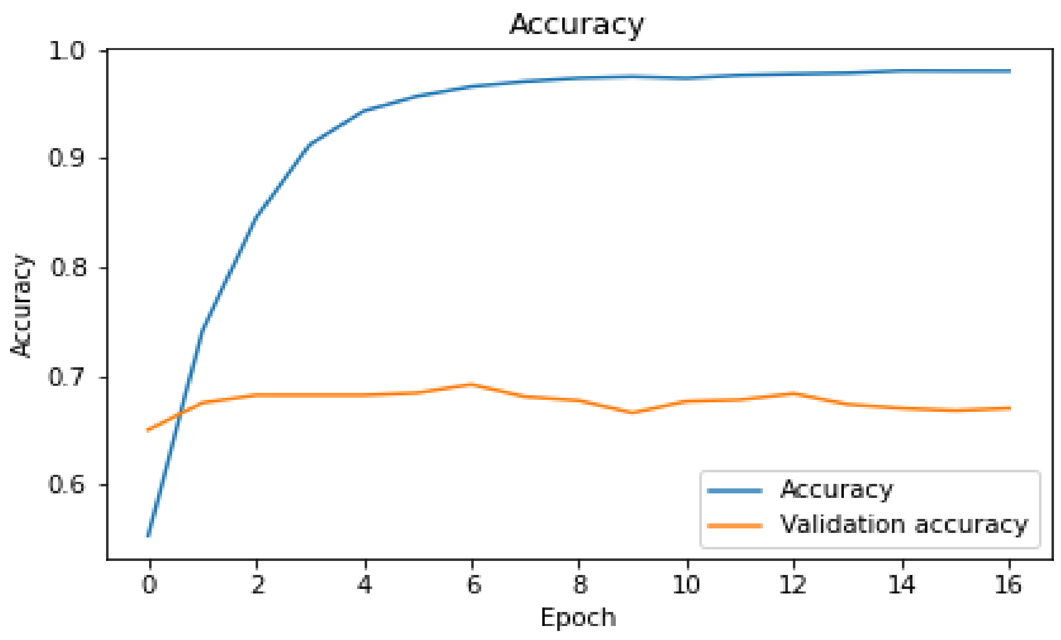


Figure 2.13: Accuracy curve of Implemented CNN model

In figure 2.14, the loss curve of implemented CNN model is shown on validation and test data.

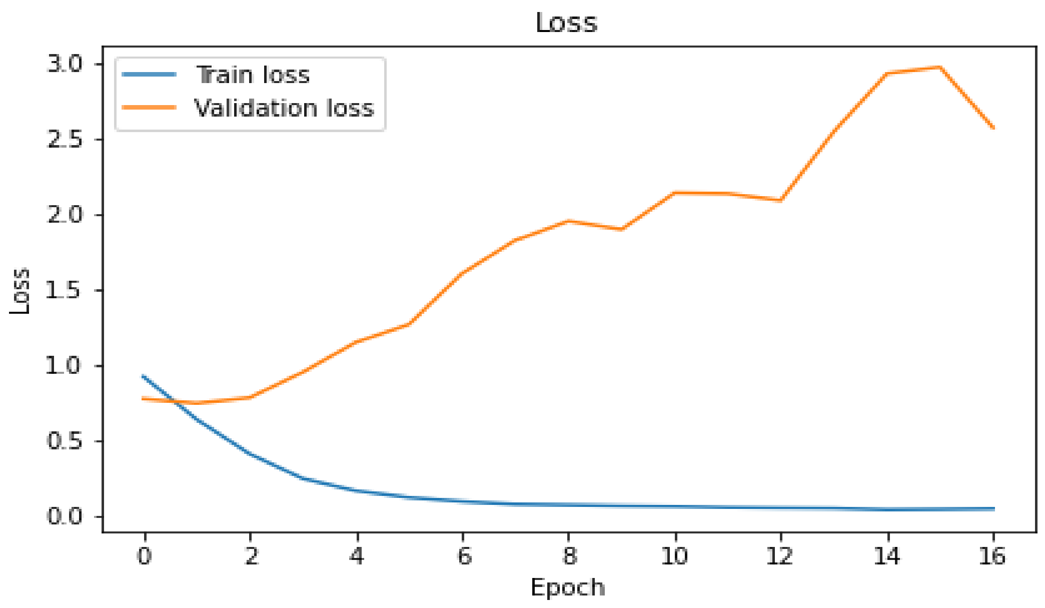


Figure 2.14: Loss curve of Implemented CNN model

In table 2.1, Confusion Matrix of the implemented CNN model is provided.

Table 2.1: Ternary class Confusion Matrix of Implemented CNN model

-	Neutral	Positve	Negative
Neutral	169	94	95
Positve	103	458	82
Negative	66	55	451

2.6.2 Recurrent Neural Network (RNN): One kind of Artificial Neural Network (ANN) is the recurrent neural network, which is utilized in speech recognition and natural language processing (NL) applications. An RNN model is made to identify the sequential properties of data, using the patterns to forecast future events. All of the inputs and outputs of classic neural networks depend on one another. Recurrent neural networks, however, use the output from earlier phases as the input for the present state. For instance, it is necessary to retain and store the previous letters or words in some type of memory in order to anticipate the following letter of any word or the following word of any phrase.

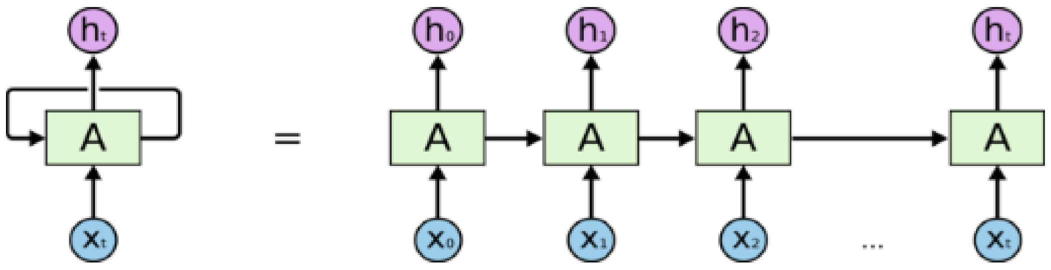


Figure 2.15: RNN Model [16]

The RNN models have an internal memory that constantly recalls the results of calculations made in earlier phases. All of the inputs are subjected to the same job, and the RNN applies the same parameter to each input. They are more complex than RNN since traditional neural networks include distinct sets of inputs and outputs. In figure 2.15 the basic architecture of implemented RNN model is provided.

Recurrent neural networks (RNNs) are the most advanced algorithm for sequential data and are the foundation of Google voice search and Apple's Siri. Due to its internal memory, it is the first algorithm to recall its input, making it ideal for machine learning issues involving sequential data. It is one of the algorithms that helped deep learning accomplish some incredible successes over the past several years. In this article, we'll go over the fundamental ideas behind recurrent neural networks, as well as the main problems they face and how to fix them.

Advantage:

- An RNN preserves all information throughout time, which is a benefit. It only helps with time series prediction if the user can remember previous inputs. This is known as long short-term memory.
- To boost the useful pixel neighborhood, convolutional layers and recurrent neural networks are even merged.

Disadvantage:

- Problems with explosions and gradient disappearance are a drawback.
- If tanh or relu are employed as the activation function, an RNN cannot handle very long sequences.
- It is extremely difficult to train.

2.6.3 Long Short-Term Memory (LSTM): Recurrent neural networks (RNN) are extended by LSTM networks, which were primarily developed to address RNN failure scenarios. Such "long-term dependencies" are completely beyond the capabilities of RNNs. Second, there is no finer control over how much of the past should be "lost" and how much of the context should be carried forward. Exploding and disappearing gradients, which happen when a network is being trained via backtracking, are another problem with RNNs. Long Short-Term Memory (LSTM) was introduced as a result. The LSTM hidden layer is a gated unit, which is the primary differentiator between LSTM and RNN systems. The output and state of the specific cell are decided by the interaction of its four levels. These two pieces are then transferred to the following buried layer. In contrast to RNNs, which only feature one logistic sigmoid gate, LSTMs also have a tanh layer. Gates have been developed to regulate the quantity of information traveling through the cell. They make judgements about what data should be ignored and what data the cell after them will need. The range for the product is 0 to 1, with 0 meaning "reject all" and 1 meaning "include all." In figure 2.16, the basic architecture of LSTM model is provided.

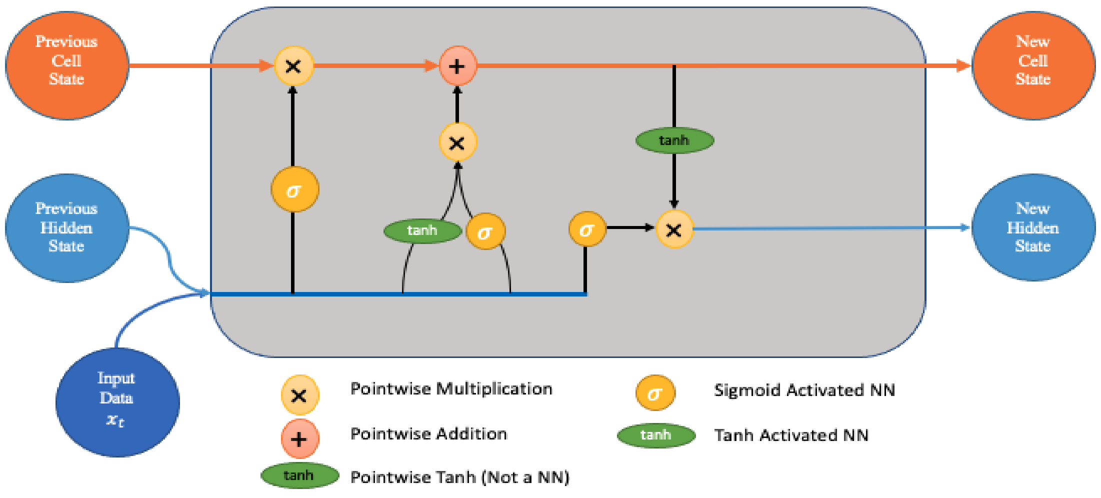


Figure 2.16: LSTM network [17]

The forget gate is the initial stage of the procedure. The new memory network and the input gate are involved in the next phase. This step's objective is to decide what new information, in light of the prior concealed state and the incoming input data, has to be added to the network's long-term memory (cell state). We may proceed to the output gate, the last stage, once we have finished updating the network's long-term memory.

Advantage:

- We may choose from a wide range of LSTM parameters, including learning rates and input and output biases.
- Thus, there is no need for precise modifications.
- With LSTMs, updating each weight is simpler than with Back Propagation Through Time (BPTT).
- It reduces complexity to O(1).

Disadvantage:

- Training LSTMs takes longer.
- To train LSTMs, additional RAM is needed.
- It is simple to overfit LSTMs.
- In LSTMs, dropout is far more difficult to implement.
- LSTMs respond differently to initializing random weights.

Bidirectional LSTM is equivalent to two LSTM running parallelly into opposite direction. One LSTM from forward to backward, another one from backward to forward. Bidirectional LSTM capture more information than unidirectional LSTM. I have used a LSTM model along with CNN. The implemented model understands the context as well as meaning of the sentence with higher accuracy score. In figure 2.17 and 2.18, model architecture and parameters of implemented Conv-LSTM is provided. In the Conv-LSTM model, textual input is provided in the input layer. By keras embedding layer, textual is converted into numeric input. Then the output of the previous layer is fed into the input of the next layer. The next layer is conv1d layer. In the conv1d layer one dimensional convolution is performed on the data. The next layer is bidirectional LSTM layer. Followed by dropout layer for avoiding overfitting. Then again bidirectional LSTM and dropout layer was implemented. Subsequently, two pairs of dense and dropout layer were implemented. Lastly, in between two dense layer, one flatten layer was implemented.

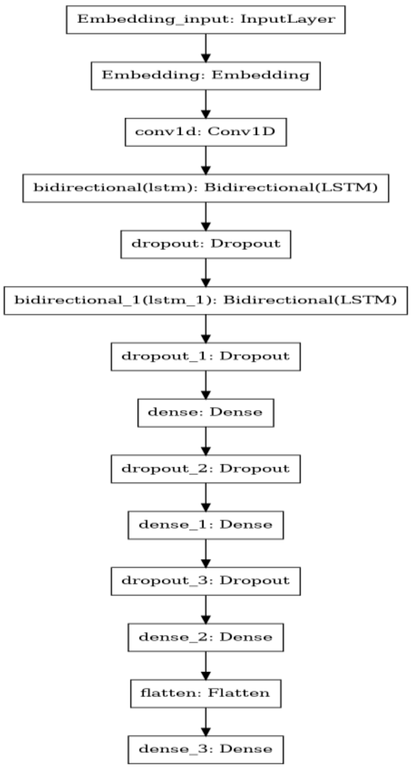
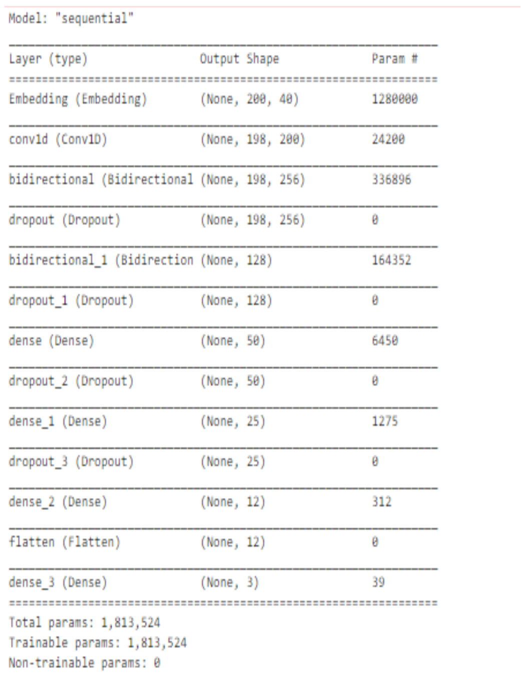


Figure 2.17: Implemented Conv-LSTM model architecture



```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
Embedding (Embedding)        (None, 200, 40)           1280000
_________________________________________________________________
conv1d (Conv1D)              (None, 198, 200)          24200
_________________________________________________________________
bidirectional (Bidirectional (None, 198, 256)          336896
_________________________________________________________________
dropout (Dropout)            (None, 198, 256)          0
_________________________________________________________________
bidirectional_1 (Bidirection (None, 128)               164352
_________________________________________________________________
dropout_1 (Dropout)          (None, 128)               0
_________________________________________________________________
dense (Dense)                (None, 50)                6450
_________________________________________________________________
dropout_2 (Dropout)          (None, 50)                0
_________________________________________________________________
dense_1 (Dense)              (None, 25)                1275
_________________________________________________________________
dropout_3 (Dropout)          (None, 25)                0
_________________________________________________________________
dense_2 (Dense)              (None, 12)                312
_________________________________________________________________
flatten (Flatten)            (None, 12)                0
_________________________________________________________________
dense_3 (Dense)              (None, 3)                 39
=================================================================
Total params: 1,813,524
Trainable params: 1,813,524
Non-trainable params: 0
```

Figure 2.18 Implemented Conv-LSTM model parameters

In figure 2.19, the accuracy curve of implemented Conv-LSTM model is shown on validation and test data.

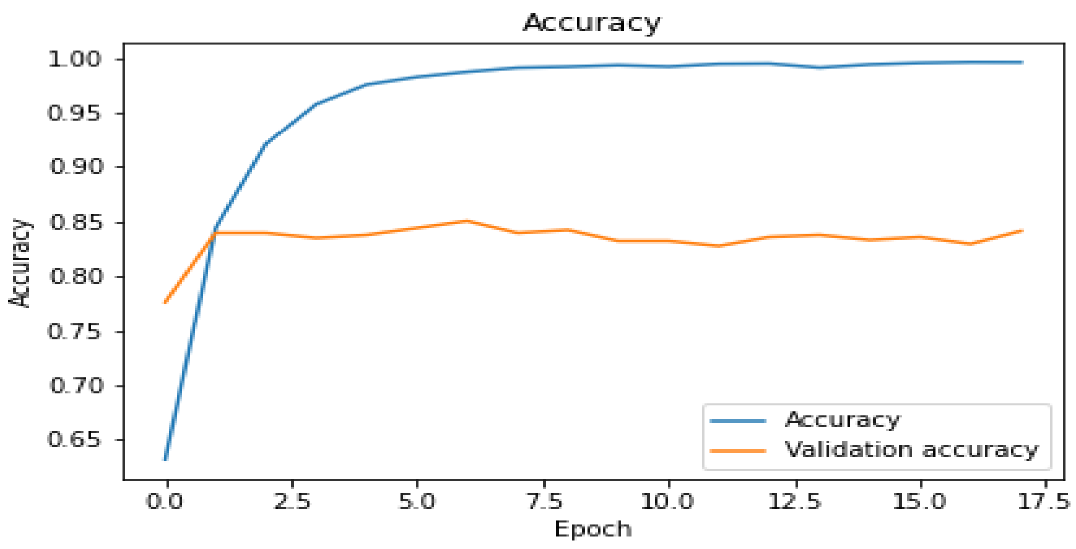


Figure 2.19: Accuracy curve of Implemented Conv-LSTM model

In figure 2.20, the loss curve of implemented Conv-LSTM model is shown on validation and test data.

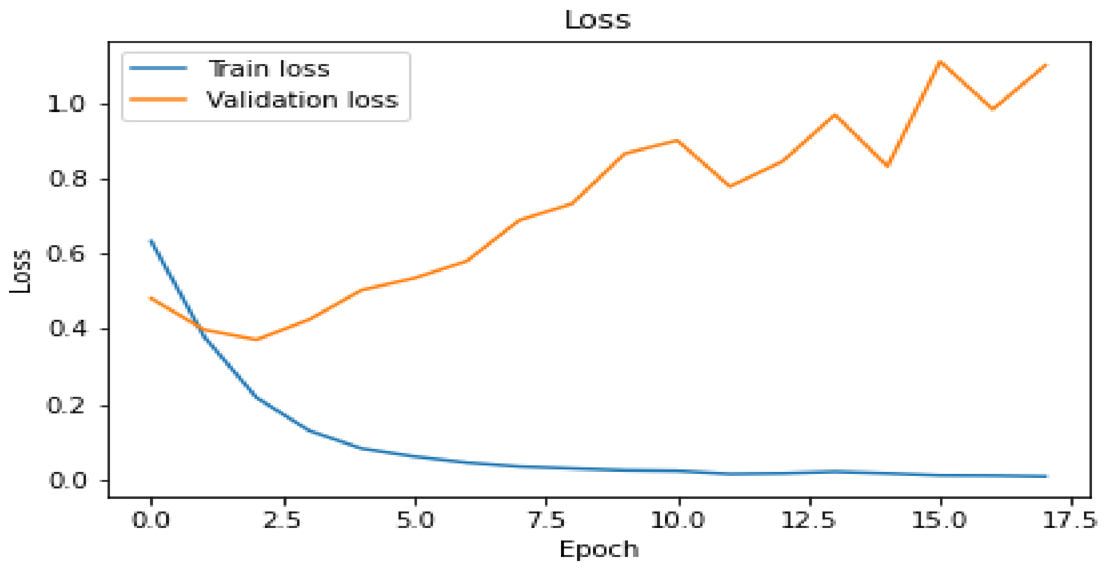


Figure 2.20: Loss curve of Implemented Conv-LSTM model

In table 2.2, Confusion Matrix of Conv-LSTM model is provided.

Table 2.2: Binary Class Confusion Matrix of Conv-LSTM model

	Positive	Negative
Positive	531	76
Negative	130	475

2.7 BERT: BERT, stands for Bidirectional Encoder Representations from Transformers, is based on Transformers which is a deep learning model, and. In Transformers, every output element is connected to every input element, and the weightings between them are dynamically determined based upon their connection. (In NLP, this procedure is referred to as attention.).It is an open-source machine learning framework for natural language processing (NLP) made to assist computers in deciphering ambiguous language in text by exploiting context provided by the text immediately around the confusing text. A question and answer dataset may be used to fine-tune the BERT framework, which was pre-trained using text from Wikipedia. It is intended to simultaneously condition on both left and right context to pre-train deep directional representations from unlabeled text. With just one extra output layer, the pre-trained BERT model may be fine-tuned to provide state-of-the-art models for a variety of NLP tasks. The whole of Wikipedia (which contains more than 2500 million words!) plus the Book Corpus make up the enormous corpus of unlabeled text on which BERT is pre-trained (800 million words). In order to provide users a deeper understanding of language, BERT is deeply bi-directional, meaning it examines the words that come before and after things as well as context learned from Wikipedia. The process of BERT pre-training and Fine-tuning is shown in the figure 2.21.

This new approach to solve NLP tasks has 2 steps:

- Utilize a sizable unlabeled text corpus to train a language model (unsupervised or semi-supervised).
- Use the vast knowledge base our model has amassed by fine-tuning it to do particular NLP tasks (supervised).

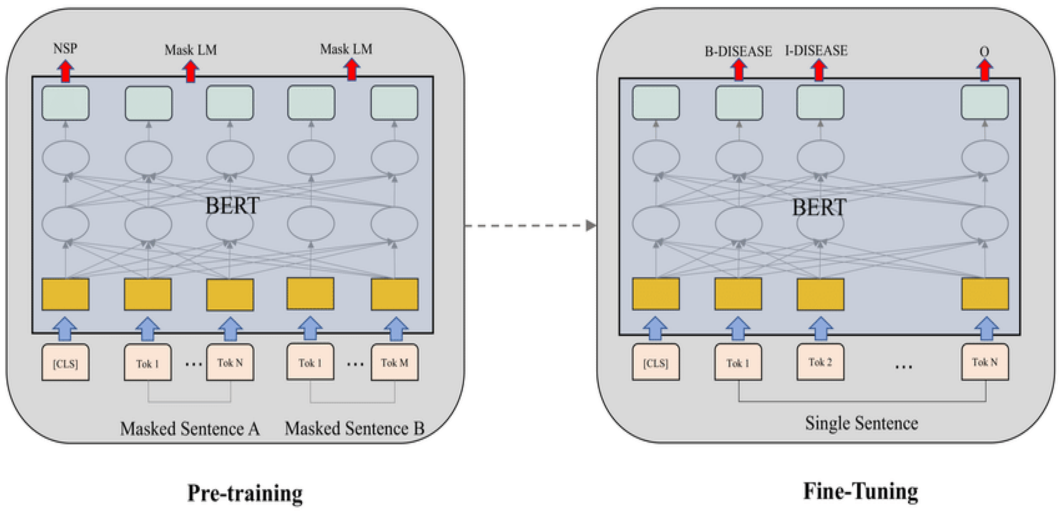


Figure 2.21: BERT Pre-Training and Fine-tuning [18]

BERT’s Architecture: Transformer serves as the foundation for the BERT architecture.

There are now two variations available:

• BERT Base: 110 million parameters, 12 Layers (transformer blocks), and 12 Attention Heads.

• BERT Large: 340 million parameters, 16 attention heads, and 24 Layers (transformer blocks).

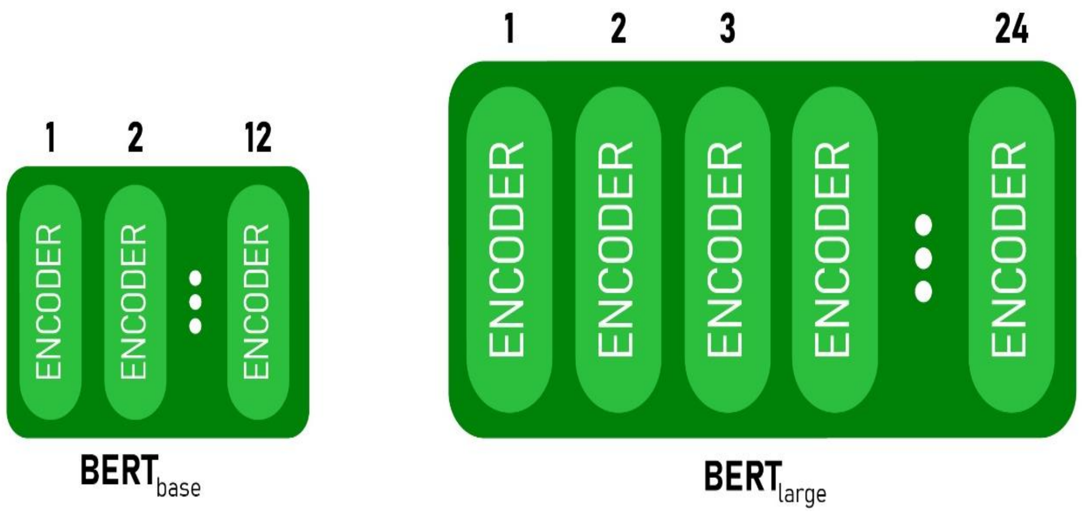


Figure 2.22: BERT Base and BERT Large [19]

The LARGE model generates state-of-the-art findings that were presented in the study article, whereas the BASE model is utilized to compare the performance of one architecture to another. The model is trained for a particular purpose that enables it to comprehend the linguistic patterns. Once trained, the model (BERT) has the ability to process language, which may be utilized to strengthen other models that we create and train using supervised learning. The architecture of BERT base and BERT large is shown in the figure 2.22.

BERT is just a transformer architecture with an encoder stack. An encoder-decoder network using self-attention on the encoder side and attention on the decoder side is known as a transformer architecture. The Encoder stack in BERTLARGE contains 24 layers compared to BERTBASE's 12 levels. These go beyond the Transformer design as it was originally defined in the article (6 encoder layers). In addition, LARGE and BASE BERT designs feature more attention heads (12 and 16 respectively) and larger feedforward networks (768 and 1024 hidden units) than the Transformer architecture proposed in the original study. It has 8 attention heads and 512 hidden units. While BERTLARGE has 340M parameters, BERTBASE only has 110M.

After the development of BERT at Google in 2018, BERT was the state-of-the-art transformer-based models used in a variety of Natural Language tasks. Though many research works have been conducted on other high resource language by training BERT models. Unfortunately, the number of BERT models trained on Bangla language is very negligible.

I have fine-tuned five BERT models trained on Bangla language from Hugging face library using ktrain wrapper. Due to requirement of GPU and high processing computers, I have used Kaggle platform to run the models. I have used following parameters in the table to fine-tune these models. The parameters used to fine-tune BERT models is provided in Table 2.3.

Table 2.3: Parameters used for fine-tuning BERT based models

Hyper parameter	**Value**
class_weight	balanced
Learning Rate	.00003
Epochs	7,10,30
preproc	trans
Max lenght	50

Chapter 3

Results and Discussions:

3.1 Introduction: In this chapter, I am going to discuss and analyze the results obtain in my thesis work. I have implemented the models on Google Colab and Kaggle by using scikit-learn library, keras library, and ktrain wrapper. At first, I have installed and import necessary required. Then load the dataset to the module. After that I have preprocessed and tokenize the data. Then the textual tokens are converted into numeric form by feature extraction. Then,the models were implemented on train data.

3.2 Result Discussion: For the evaluation of the results of the implemented models, Accuracy and Macro Averaged Precision, Recall and F1 scores were used.

Accuracy measures the accurate prediction of any instance out of all the data.

$$\text{Accuracy} = \frac{\text{True Positive} + \text{True Negative}}{\text{True Positive} + \text{True Negative} + \text{False Positive} + \text{False Negative}} \ldots\ldots\ldots\ldots (3.1)$$

Precision determines the chances of true positive prediction among true positive and false positive classes.

$$Precision\ = \frac{True\ Positive}{True\ Positive\ +\ False\ Positive} \ldots\ldots\ldots (3.2)$$

Recall measures the ratio of true positive among true positive and false negative.

$$Recall = \frac{True\ Positive}{True\ Positive\ +\ False\ Negative} \ldots\ldots\ldots (3.3)$$

The average of Precision and Recall is known as F1 Score.

$$F1\ Score\ =\ 2\ \times \frac{Precision\ \times\ Recall}{Precision\ +\ Recall} \ldots\ldots\ldots\ldots (3.4)$$

The contribution of my thesis is that automatic finding of the best parameters of the machine learning models. For this, I used GridSearchCV method. GridSearchCV method tries all the combinations possible for the parameters provided and gives the best parameter which provides highest accuracy for the given dataset.

In binary classification among ML models SVM (unigram) achieved the highest 86% F1 score which is surprisingly equal to CNN and ’csebuetnlp/banglabert’ score. In ternary classification SVM(unigram) scored 67% F1-score which is equal to CNN but ’csebuetnlp/banglabert’ achieved. The experimental data obtained by running the programs is provided in the table 3.1, 3.2, 3.3, 3.4, 3.5, 3.6, 3.7, 3.8.

Table 3.1: Evaluation of the ML models’ performance on test data for ternary class feature extracted by BOW

Model	**Best parameters**	**Precision**	**Recall**	**F1-score**	**Accuracy**
svm	{'C': 10, 'decision_function_shape': 'ovo', 'gamma': 'scale', 'kernel': 'rbf'}	0.573254048	0.570265473	0.569966374	0.604577241
decision tree	{'criterion': 'entropy', 'max_depth': 900, 'max_features': 'sqrt', 'min_samples_split': 15}	0.496345245	0.494369265	0.49461259	0.522250477
logistic_regression	{'C': 1}	0.522903624	0.514303619	0.499109656	0.569612206
multinomial naive bayes	{'alpha': 0.9}	0.520778071	0.513854463	0.501453402	0.567069294
k nearest neighbors	{'algorithm': 'ball_tree', 'n_neighbors': 7, 'weights': 'distance'}	0.529521153	0.529198095	0.528682226	0.550540369
Stochastic Gradient Descent	{'alpha': 0.0005, 'loss': 'log', 'penalty': 'l2'}	0.534852632	0.512474794	0.493150026	0.569294342

Table 3.2: Evaluation of the ML models' performance on test data for ternary class feature extracted by TF-IDF unigram

Model	**Best parameters**	**Precision**	**Recall**	**F1-score**	**Accuracy**
svm	{'C': 20, 'decision_function_shape': 'ovo', 'gamma': 'scale', 'kernel': 'rbf'}	0.691751	0.674937	0.676743	0.718055
decision tree	{'criterion': 'entropy', 'max_depth': 700, 'max_features': 'sqrt', 'min_samples_split': 15}	0.572802	0.570278	0.570956	0.597266
logistic_regression	{'C': 5}	0.666818	0.653257	0.653555	0.698029
multinomial naive bayes	{'alpha': 0.25}	0.666882	0.624229	0.617454	0.682136
k nearest neighbors	{'algorithm': 'ball_tree', 'n_neighbors': 7, 'weights': 'distance'}	0.623911	0.619867	0.620239	0.657025
Stochastic Gradient Descent	{'alpha': 0.0001, 'loss': 'log', 'penalty': 'l2'}	0.661202	0.612873	0.601547	0.675461

Table 3.3: Evaluation of the DL and BERT models' performance on test data for ternary class

Feature	Model	Precision	Recall	F1-score	Accuracy
Embedding	CNN	0.67	0.67	0.67	0.70
Embedding	Conv-LSTM	0.63	0.62	0.62	0.66
-	csebuetnlp/banglabert	0.69	0.69	0.69	0.72
-	Kowsher/bangla-bert	0.66	0.66	0.66	0.69
-	monsoon-nlp/bangla-electra	0.59	0.58	0.58	0.60
-	bert-base-multilingual-cased	0.69	0.69	0.69	0.71
-	sagorsarker/bangla-bert-base	0.66	0.66	0.66	0.68

Table 3.4: Evaluation of the ML models' performance on test data for Binary class feature extracted by BOW

Model	Best parameters	Precision	Recall	F1-score	Accuracy
svm	{'C': 10, 'decision_function_shape': 'ovo', 'gamma': 'scale', 'kernel': 'rbf'}	0.737880562	0.73847428	0.737824	0.738036
decision tree	{'criterion': 'entropy', 'max_depth': 900, 'max_features': 'sqrt', 'min_samples_split': 15}	0.683564335	0.682875917	0.683076	0.684406
logistic_regression	{'C': 1}	0.716996012	0.717461779	0.717061	0.717409

multinomial naive bayes	{'alpha': 0.75}	0.732419253	0.732512048	0.732463	0.733086
k nearest neighbors	{'algorithm': 'ball_tree', 'n_neighbors': 6, 'weights': 'distance'}	0.702167577	0.70261421	0.702209	0.702558
Stochastic Gradient Descent	{'alpha': 0.0001, 'loss': 'log', 'penalty': 'l2'}	0.720177111	0.72012076	0.718234	0.718234

Table 3.5: Evaluation of the ML models' performance on test data for Binary class feature extracted by TF-IDF unigram

Model	**Best parameters**	**Precision**	**Recall**	**F1-score**	**Accuracy**
svm	{'C': 10, 'decision_function_shape': 'ovo', 'gamma': 'scale', 'kernel': 'rbf'}	0.860259726	0.858464	0.859075	0.859736
decision tree	{'criterion': 'entropy', 'max_depth': 700, 'max_features': 'sqrt', 'min_samples_split': 55}	0.752720406	0.751199	0.751623	0.752888
logistic_regression	{'C': 10}	0.852166161	0.85156	0.851816	0.85231
multinomial naive bayes	{'alpha': 0.4}	0.843157269	0.842337	0.842664	0.843234
k nearest neighbors	{'algorithm': 'ball_tree',	0.829349831	0.827983	0.828461	0.829208

	'n_neighbors': 6, 'weights': 'distance'}				
Stochastic Gradient Descent	{'alpha': 0.0001, 'loss': 'log', 'penalty': 'l2'}	0.831332472	0.829388	0.830002	0.830858

Table 3.6: Evaluation of the DL and BERT models' performance on test data for Binary class

Feature	**Model**	**Precision**	**Recall**	**F1-score**	**Accuracy**
Embedding	CNN	0.86	0.86	0.86	0.86
Embedding	Conv-LSTM	0.83	0.83	0.83	0.83
-	csebuetnlp/banglabert	0.86	0.86	0.86	0.86
-	Kowsher/bangla-bert	0.84	0.84	0.84	0.84
-	monsoon-nlp/bangla-electra	0.78	0.77	0.77	0.77
-	bert-base-multilingual-cased	0.82	0.82	0.82	0.82
-	sagorsarker/bangla-bert-base	0.83	0.83	0.83	0.83

Table 3.7: Evaluation of the ML models' performance on test data for Binary class feature extracted by TF-IDF Bigram

Model	**Best parameters**	**Precision**	**Recall**	**F1-score**	**Accuracy**
svm	{'C': 1, 'decision_function_shape': 'ovo', 'gamma': 'auto', 'kernel': 'linear'}	0.856654	0.855363	0.855839	0.856436
decision tree	{'criterion': 'entropy', 'max_depth': 700, 'max_features': 'sqrt', 'min_samples_split': 15}	0.759604	0.75684	0.757425	0.759076
logistic_regression	{'C': 10}	0.861416	0.859767	0.860342	0.860974
multinomial naive bayes	{'alpha': 0.4}	0.86666	0.864564	0.865254	0.865924
k nearest neighbors	{'algorithm': 'ball_tree', 'n_neighbors': 7, 'weights': 'distance'}	0.811363	0.809535	0.8101	0.811056
Stochastic Gradient Descent	{'alpha': 0.0001, 'loss': 'log', 'penalty': 'l2'}	0.843861	0.839868	0.840858	0.841997

Islam et al. [1] implemented BiLSTM on the SentNoB dataset by using various grams as feature extraction techniques and found highest F1 score of 64.03% for ternary class. Whereas, I have found highest F1 score of 69% by fine tuning 'csebuetnlp/banglabert'.

Table 3.8: Comparison with previous works

Model	Feature Extraction	Precision	Recall	F1-score	Accuracy
BiLSTM + Attention [1]	FastText	0.5224	0.6309	0.5715	-
BiLSTM + Attention[1]	Random	0.5616	0.6497	0.6025	-
mBERT [1]	-	0.4958	0.5643	0.5279	-
Implemented SVM	BOW	0.5732	0.5702	0.5699	0.6045
Implemented SVM	Unigram	0.6917	0.6749	0.6767	0.7180
Implemented CNN	Keras Embedding layer	0.67	0.67	0.67	0.70
csebuetnlp/banglabert	-	0.69	0.69	0.69	0.72

Overall BERT based models performed better than ML and DL models in my observation but the difference is less compared to other researches.The best performance was showed by ’csebuetnlp/banglabert’ for both binary and ternary classes. The confusion matrix of ’csebuetnlp/banglabert’ is given in the following figures.

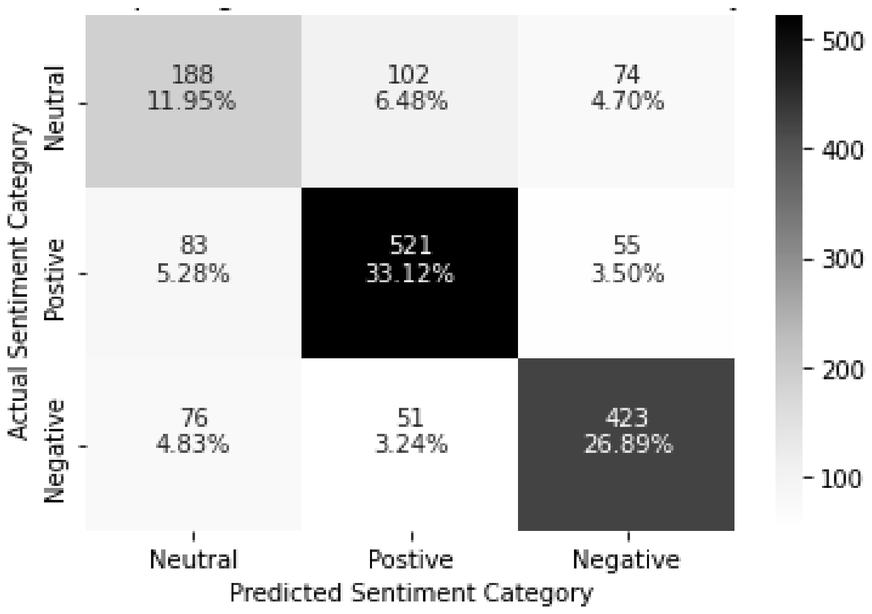


Figure 3.1: Confusion Matrix of ’csebuetnlp/banglabert’ model (Ternary class)

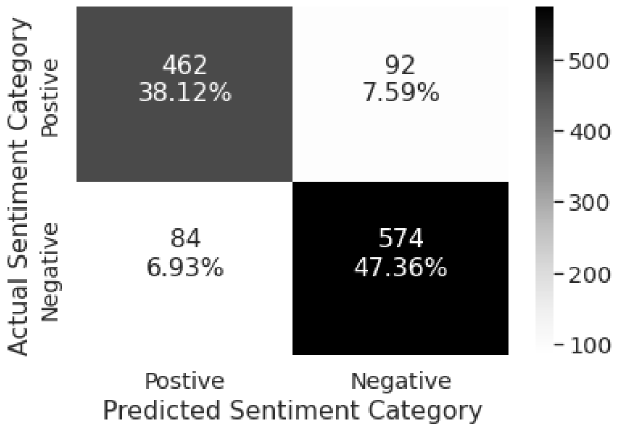


Figure 3.2: Confusion Matrix of 'csebuetnlp/banglabert' model (Binary class)

The models used in this paper struggled to predict the neutral class correctly which dragged down the models' scores. We also found out that as the number of sentiment classes increased from binary to ternary class, the overall performance of all models decreased. Thus, we can claim that an inverse relationship exists between the number of classes and the performance of the models.

Chapter 4:

Conclusion & Future Work

4.1 Introduction: In this chapter, a brief concluding remarks has been provided on my thesis work, the limitations and drawbacks and the future direction of my thesis research also discussed here.

4.2 Conclusion: In my research, after collection of SentNoB dataset, I have done the required preprocessing by removing unwanted words. Then, the tokenization of the Bangla sentences has been. Next phase was the feature extraction. I have extracted features by BOW, TF-IDF unigram and bigram, Keras embedding layer. After splitting the dataset into train, validation, split I have trained the models by using train data. The performance of the models is evaluated using test data. I have found that when I have increased the number of classes from two to three, the performance of all the models decreased. I have also implemented by models to five and six classes dataset. There I also found that accuracy decreased drastically with increasing the number of classes.

4.3 Limitations: The number of datasets available on Bangla Language is very negligible. Initially, I intend to show relative comparison between binary, ternary, five and six multiclass on same dataset. But due to lack of dataset, I could get any dataset of this sort in Bangla Language. I have tried my models on different multiclass datasets, but models predict the classes very poorly. Nowadays, different word embedding techniques are available, like Glove, Word2vec, Fasttext. But I have only used old feature extraction techniques like BOW, TF-IDF, keras embedding layer for converting textual data into numeric data. The performance of ML models is lower compare to DL and BERT models. But I didn't find any such significant performance difference.

4.4 Future works: In future, I will create a Multiclass Bangla dataset. By using different state of the word embedding technique, I will extract the features and train different models. Then I will train a variant of BERT model to predict the sentiment of the multiclass dataset. I am also working 'Next Day Stock Price Prediction' dataset to predict the next day stock price by analyzing the Bangla newspaper headline. Also I am trying different word embedding techniques for unsupervised learning of Bangla text.

References:

[1] K. Islam, S. Kar, M. Islam and M. Amin, "SentNoB: A Dataset for Analysing Sentiment on Noisy Bangla Texts", *Findings of the Association for Computational Linguistics: EMNLP 2021*, 2021. Available: 10.18653/v1/2021.findings-emnlp.278 [Accessed 30 September 2022].

[2] M. Karim et al., "DeepHateExplainer: Explainable Hate Speech Detection in Under-resourced Bengali Language", *2021 IEEE 8th International Conference on Data Science and Advanced Analytics (DSAA)*, 2021. Available: 10.1109/dsaa53316.2021.9564230 [Accessed 30 September 2022].

[3] Bhattacharjee, A., Hasan, T., Samin, K., Islam, M. S., Rahman, M. S., Iqbal, A.,Shahriyar, R. (2021). BanglaBERT: Combating Embedding Barrier in Multilingual Models for Low-Resource Language Understanding. arXiv preprint arXiv:2101.00204.

[4] Das A, Sharif O, Hoque MM, Sarker IH. Emotion classification in a resource constrained language using transformer-based approach. arXiv preprint arXiv:2104.08613. 2021 Apr 17.

[5] N. Irtiza Tripto and M. Eunus Ali, “Detecting multilabel sentiment and emotions from Bangla YouTube comments,” 2018 International Conference on Bangla Speech and Language Processing (ICBSLP), 2018.

[6] Sharif, O., Hossain, E., Hoque, M. M. (2021). NLP-CUET@ DravidianLangTechEACL2021: Offensive language detection from multilingual code-mixed text using transformers. arXiv preprint arXiv:2103.00455.

[7] M. R. Karim et al., "DeepHateExplainer: Explainable Hate Speech Detection in Under-resourced Bengali Language," 2021 IEEE 8th International Conference on Data Science and Advanced Analytics (DSAA), 2021, pp. 1-10, doi:10.1109/DSAA53316.2021.9564230.

[8] N. Irtiza Tripto and M. Eunus Ali, “Detecting multilabel sentiment and emotions from Bangla YouTube comments,” 2018 International Conference on Bangla Speech and Language Processing (ICBSLP), 2018.

[9] “Logistic regression in machine learning - javatpoint,” www.javatpoint.com. [Online]. Available: https://www.javatpoint.com/logistic-regression-in-machine-learning. [Accessed: 05-Oct-2022].

[10] “Support Vector Machine (SVM) algorithm - javatpoint,” www.javatpoint.com. [Online]. Available: https://www.javatpoint.com/machine-learning-support-vector-machine-algorithm. [Accessed: 11-Oct-2022].

[11] “Machine learning decision tree classification algorithm - javatpoint,” www.javatpoint.com. [Online]. Available:https://www.javatpoint.com/machine-learning-decision-tree-classification-algorithm. [Accessed: 12-Oct-2022].

[12] “K-Nearest Neighbor(KNN) algorithm for Machine Learning - Javatpoint,” www.javatpoint.com. [Online].Available:https://www.javatpoint.com/k-nearest-neighbor-algorithm-for-machine-learning. [Accessed: 13-Oct-2022].

[13] G. Regunath, “Understanding the difference between AI, ML, and DL: Using an incredibly simple example,” Advancing Analytics, 16-Dec-2021. [Online]. Available: https://www.advancinganalytics.co.uk/blog/2021/12/15/understanding-the-difference-between-ai-ml-and-dl-using-an-incredibly-simple-example. [Accessed: 14-Oct-2022].

[14] J. J. Moolayil, “A layman's Guide to Deep Neural Networks,” Medium, 30-May-2020. [Online]. Available: https://towardsdatascience.com/a-laymans-guide-to-deep-neural-networks-ddcea24847fb. [Accessed: 14-Oct-2022].

[15] A. Biswal, “Convolutional Neural Network tutorial [update],” Simplilearn.com, 21-Sep-2022. [Online]. Available: https://www.simplilearn.com/tutorials/deep-learning-tutorial/convolutional-neural-network. [Accessed: 14-Oct-2022].

[16] A. Mittal, “Understanding RNN and LSTM,” Medium, 26-Aug-2021. [Online]. Available: https://aditi-mittal.medium.com/understanding-rnn-and-lstm-f7cdf6dfc14e. [Accessed: 15-Oct-2022].

[17] R. Dolphin, “LSTM networks: A detailed explanation,” Medium, 12-Dec-2021. [Online]. Available: https://towardsdatascience.com/lstm-networks-a-detailed-explanation-8fae6aefc7f9. [Accessed: 15-Oct-2022].

[18] “Overall pre-training and fine-tuning procedures for bi-directional ...” [Online]. Available: https://www.researchgate.net/figure/Overall-pre-training-and-fine-tuning-procedures-for-bi-directional-encoder_fig3_347939321. [Accessed: 15-Oct-2022].

[19] “Explanation of Bert Model - NLP,” GeeksforGeeks, 20-Jun-2022. [Online]. Available: https://www.geeksforgeeks.org/explanation-of-bert-model-nlp/. [Accessed: 15-Oct-2022].