

TWITTER SENTIMENTAL ANALYSIS OF COVID TOWARDS MENTAL DEPRESSION USING DEEP LEARNING

A PROJECT REPORT

Submitted By

NASEELA PARVEZ [Reg No: RA1711008010096]

ADITYA AGARWAL [Reg No: RA1711008010343]

Under the Guidance Of

Dr S. SURESH

(Professor, Department Of Information Technology)

In partial fulfillment of the Requirements of the Degree

Of

BACHELOR OF TECHNOLOGY



DEPARTMENT OF INFORMATION TECHNOLOGY

FACULTY OF ENGINEERING AND TECHNOLOGY

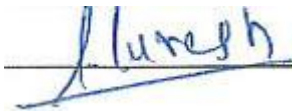
APRIL 2021

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

KATTANKULATHUR – 603203

BONAFIDE CERTIFICATE

Certified that this project report titled “**TWITTER SENTIMENT ANALYSIS OF COVID TOWARDS MENTAL DEPRESSION USING DEEP LEARNING**” is the Bonafide work of “**NASEELA PARVEZ [Reg No: RA1711008010096], ADITYA AGARWAL [Reg No:RA1711008010343]**, who carried out the project work under my supervision. Certified further, that to the best of my knowledge the work reported herein does not form part of any other thesis or dissertation on the basis of which a degree or award was conferred on an earlier occasion for this or any other candidate.



Dr.S.SURESH

GUIDE

Professor

Dept. of Information Technology Dept. of Information Technology

Signature of Internal Examiner



Dr. G. VADIVU
Head
Department of Information Technology
Faculty of Engineering & Technology
SRM Institute of Science and Technology
SRM Nagar, Kattankulathur - 603 203.
Kancheepuram Dist., Tamil Nadu, India.

Dr. G. VADIVU

HEAD OF THE DEPARTMENT

Professor

Signature of External Examiner

ABSTRACT

The coronavirus pandemic hit the worldwide population to a large extent. The economy of the most secured nations plummeted, lives were lost and a larger portion of the population lost their jobs. But one of the subtle effects of the COVID-19 pandemic is the depletion of the mental health of the people. A person without a sound mental capacity resembles an animal wandering the roads to find food. As responsible citizens and inhabitants of the world, we must contribute to improving the mental health of those affected.

Our research is based on Twitter sentiment analysis. According to psychiatrists, 20% of the people affected by the coronavirus developed mental illness within 90 days. Apart from this, staying at home during the lockdown has depleted the mental health of the normal population. Social media has become an efficient platform to express oneself. And Twitter is one of the most used platforms. The project focused on predicting depression in a person using tweets. The project employed the Deep Learning as well as Natural language processing model for predicting the depression in individuals which included depressed, non-depressed and even neutral. The dataset that is required to train the model was acquired using the Twitter API, Tweepy library in python, and online hydrators were additionally employed to collect the tweets. Some of the prospective models that were implemented are Bi-LSTM, BERT and XLNET. The performance of the model were evaluated during lockdown and post lock down.

ACKNOWLEDGEMENT

The success and the final outcome of this project required guidance and assistance from different sources and we feel extremely fortunate to have got this all along the completion of our project. Whatever we have done is largely due to such guidance and assistance and we would not forget to thank them.

We would like to thank our Project Guide (Engineering & Technology), Dr. Suresh Sankaranarayanan for his constant support and facilities in department of Information Technology for completing the project towards B.Tech degree and also for taking keen interest in our project work and guided us all along, till the completion of our project work by providing all the necessary information for developing a good system.

Secondly we express our sincere thanks to Dr. G. Vadivu, Professor and the Head of the Department for all the help and infrastructure provided to us to complete this project successfully.

We are thankful to all the teaching and non-teaching staff of our Department for helping us in bringing this project to a success.

TABLE OF CONTENTS

CHAPTER:	PAGE NO:
ABSTRACT	III
LIST OF TABLES	VIII
LIST OF ABBREVIATIONS	IX
LIST OF FIGURES	X
1.INTRODUCTION	11
1.1 THE RISE OF COVID-19 ACROSS THE GLOBE	11
1.2 MENTAL HEALTH	12
1.3 TWEET SENTIMENTAL ANALYSIS FOR MENTAL DEPRESSION	13
1.4 MOTIVATION	13
1.5 PROPOSED WORK	14
1.6 OBJECTIVES	15
2. LITERATURE REVIEW	16
2.1 TRADITIONAL APPROACHES TO TWITTER SENTIMENT ANALYSIS	16
2.2 DEEP LEARNING FOR TWITTER SENTIMENT ANALYSIS	17
3. SYSTEM ANALYSIS	20
3.1 PROBLEM DEFINITION	20
3.2 DATA PRE-PROCESSING	21
3.3 LABELLING THE TWITTER DATA	23
3.4 TF-IDF VECTORS	23

3.5 SUPPORT VECTOR CLASSIFIER	24
3.6 DEEP LEARNING ALGORITHMS FOR TWITTER SENTIMENT ANALYSIS	25
3.6.1 BIDIRECTIONAL LONG SHORT-TERM MEMORY (BI-LSTM)	25
3.6.2 BIDIRECTIONAL ENCODER REPRESENTATIONS FROM TRANSFORMERS	27
3.6.3 XLNET	33
4. SYSTEM DESIGN	37
4.1 SYSTEM ARCHITECTURE	37
4.2 DATA FLOW DIAGRAM	38
4.3 FLOW CHART	39
5. CODING AND TESTING	41
5.1 PROGRAMMING ENVIRONMENT	41
5.2 PROGRAMMING LANGUAGE	42
5.3 PYTHON LIBRARIES USED DURING IMPLEMENTATION	43
5.4 TESTING	45
5.4.1 UNIT TESTING	45
6. RESULTS AND ANALYSIS	46
6.1 DATASET	46
6.2 METRICS FOR MODEL EVALUATION	49
6.2.1 PRECISION AND RECALL:	49
6.2.2 ACCURACY	50
6.2.3 F1 SCORE	50
6.3 PERFORMANCE OF DEEP-LEARNING MODELS	50
6.3.1 Bi-LSTM	51
6.3.2 BERT MODEL	53

6.3.3 XLNET	54
6.4 EXPLORATORY DATA ANALYSIS	60
6.5 POST- VACCINATION AND POST-LOCKDOWN ANALYSIS	61
7. CONCLUSION AND FUTURE WORK	65
7.1 CONCLUSION	65
7.2 FUTURE ENHANCEMENTS	66
REFERENCES	67
APPENDIX	69
PLAGIARISM REPORT	102

LIST OF TABLES

TABLE NO:	TABLE NAME:	PAGE NO:
1.	Bi- LSTM results	44
2.	BERT results	44
3.	XLNET Results	45

LIST OF ABBREVIATIONS

ML	MachineLearning
BERT	Bi-directional EncoderRepresentations
SVM	Support VectorClassifier
Bi-LSTM	Bi-drectional Long Short TermMemory
TF-IDF	Term Frequency Inverse Document Frequency

LIST OF FIGURES

FIGURE NO:	TITLE	PAGE NO:
Fig 1.1	Increasing Number of COVID-19 Cases	2
Fig 1.2	Indirect Consequences On Mental Health	5
Fig 3.1	Hyperplane separating the different classes	14
Fig 3.6.1	Architecture of LSTM	16
Fig 3.6.3	Architecture diagram of BERT	19
Fig 3.6.4	Multi-head attention mechanism	21
Fig 3.6.5	BERT model for classification of tweets	23
Fig 3.6.6	working of XLNET	25
Fig 4.1	System Architecture Diagram	27
Fig 4.2	Data Flow Diagram	28
Fig 4.3	Flow Chart	29
Fig 5.1	Google Colab environment	32
Fig 6.1	Raw twitter dataset with all the columns	37
Fig 6.2	Pre-processed twitter dataset after labelling	37
Fig 6.3	Word cloud visualization of tweets during lockdown	39
Fig 6.4	Word cloud visualization of tweets post lockdown and post vaccination	39
Fig 6.5	ROC curve for neutral tweets in Bi-LSTM	46
Fig 6.6	ROC curve for depressed tweets in Bi-LSTM	46
Fig 6.7	ROC curve for non-depressed tweets in Bi-LSTM	47
Fig 6.8	Accuracy of Bi-LSTM through the epochs	47
Fig 6.9	Training and validation loss of Bi-LSTM through the epochs	48
Fig 6.10	Loss of BERT through the epochs	48
Fig 6.11	Loss of XLNET through the epochs	49
Fig 6.12	Accuracy of the models implemented	49
Fig 6.13	Precision of the models implemented	50
Fig 6.14	Recall of the models implemented	50
Fig 6.15	F1 Score of the models implemented	51
Fig 6.16	Distribution of tweets during lockdown	53
Fig 6.17	Distribution of tweets post-vaccination	53
Fig 6.18	Word distribution in tweets during lockdown	54
Fig 6.19	Word distribution in tweets post-vaccination	54

CHAPTER 1

INTRODUCTION

1.1 THE RISE OF COVID-19 ACROSS THE GLOBE

The coronavirus pandemic affected a considerable portion of the world's population and has been one of a kind in many ways. From the virus being new to mundane human stories spread like a wild forest fire, COVID-19 is an unforgettable experience. According to [1], the world's first case of coronavirus was reported in China on Nov 17, 2019 and by March 2020, it was a global pandemic. The nationwide lockdown was imposed in every nation. One thing that the virus impacted directly was human lives. As of Mar 22, 2021, 2.71 million deaths have been recorded worldwide.

The currencies of the most secure nations collapsed, lives were lost, and a more significant proportion of the people lost their employment. The secondary effects of COVID-19 were depleting economies, unemployment, impacted food systems, etc. At the same time, the primary one has been on the mental health of people. Besides COVID-19, loneliness, anxiety, and lack of money are also causing or exacerbating mental health problems. Many individuals may be seeing an uptick in alcohol and opioid use. COVID-19, on the other hand, can cause neuro-logical and psychiatric issues like delirium, pressure, and stroke.

Psychological wellness is getting more common as time passes. As indicated by the World Wellbeing Association, one out of each four individuals will have psychological well-being issues. Besides the danger of infection transmission, the Coronavirus pandemic has brought about many social outcomes, including monetary difficulty, work misfortune, and rising joblessness, to give some examples. The individuals who work in essential consideration are bound to experience the ill effects of mental pressure. Notwithstanding, to discover a fix, we should initially understand the enthusiastic state. Individuals utilize online media locales like Facebook, Instagram, Twitter, and Reddit to share their considerations and perspectives.

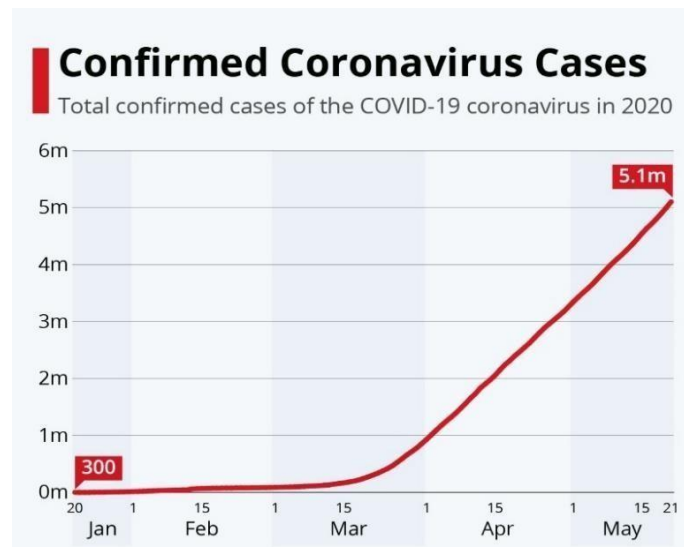


Fig 1.1: Increasing Number of COVID-19 Cases

1.2 MENTAL HEALTH

The mental health of individuals has depleted during the lockdown. The loss of people's mental health is one of the more indirect consequences of the COVID pandemic. [2] has put forward how the pandemic has affected the mental health of people. We were probably bound to four walls for months for the first time in our lives. As the COVID pandemic spread throughout the planet, it is causing inescapable frenzy, stress, and worry among the overall population.

The schools, colleges, and offices were shut in every country. The daily routine of people was altered, inhibiting them from doing what they want. Apart from that, being at home throughout the lockdown has harmed the general public's mental health. Apart from this, the terror created by the rising number of cases impacted people's mental health. The number of deaths recorded every day is the leading cause of people's mental health deterioration.

Besides, mental and actual prosperity are inseparably connected. Psychological sicknesses, for example, despondency and uneasiness, can make it hard for people to participate in better propensities. Fundamental prosperity issues will likewise make it more challenging for patients to get prescriptions for mental diseases. Expanded emotional wellness screening can help patients in accepting the consideration they need.

During the pandemic, a higher-than-normal extent of youthful grown-ups (ages 12-24) experienced uneasiness and sorrow. Young adults were more likely than all grown-ups to encounter drug use and self-destructive contemplations.

Youngsters were likewise in great danger of insufficient mental prosperity and narcotic use issues before the pandemic; however, many didn't go through prescription.

Numerous youngsters have encountered uplifted dread, discouragement, rest aggravations, and self-destructive usings because of the pandemic. They've effectively managed a large number of pandemic-related impacts, including college terminations, movements to distant positions, and the absence of payer business, all of which could unleash destruction on their psychological wellness. After a few examinations, we have discovered that all through the pandemic, a considerable portion of youthful grown-ups have clear manifestations of nervousness as well as burdensome issues.

An individual with poor cognitive ability is akin to an animal roaming the streets looking for food. People with pre-existing psychiatric, neurological, or substance-abusing conditions are often more vulnerable to more severe infections, and they may face a greater risk of severe consequences, including death.

1.3 TWEET SENTIMENTAL ANALYSIS FOR MENTAL DEPRESSION

Sentiment analysis is a hot subject that has been studied for decades to determine the meaning of the text and label it as positive, negative, or neutral. Today's advanced technologies include many methods for efficiently extracting, analyzing, and categorizing text data through various categories. In today's digital world, social media networks such as Facebook, Twitter, and others provide a vast amount of data for sentiment analysis of both image and text; however, using data from specific social media networks poses privacy concerns; hence, Twitter is the best social media network for obtaining enough data while still protecting us from privacy laws.

There has been work where Machine and Deep learning employed for Tweet Sentimental analysis for different applications. In terms of Covid-19 pandemic related tweets, Bag of Words were used as features and support Vector classifier for classification of tweets. There has been other work pertaining to mental depression tweets where deep learning models like RNN and GRU models used.

1.4 MOTIVATION

Emotional and mental welfare is vital because it affects your feelings, attitudes, and emotions and may be an elementary part of your life. Emotional well-being will boost potency and effectiveness in tasks like work, education, and caregiving. It helps you go with changes in your life and modify hardship, which is essential for your relationships' welfare.

The motive for this work was to show how irrationally people behave in the face of a pandemic. This particular pandemic seems to have taken millions of lives and is causing the global economy to collapse. It would've been more accessible for the perpetrators to collect structured information from the internet at this time. This necessitates a rigorous quality control of tweets to ensure that relevant content is shared on these widely used social networking sites. This project aims to check on social media sites before they are widely disseminated, so that false and inappropriate information does not spread among netizens.

Sentiment analysis may reveal important details about a community's risk perceptions. This cutting-edge computational method examines textual data to interpret and categorize public sentiment with the depletion in mental health. Indeed, Twitter has been widely used for public health surveillance, ranging from tracking and forecasting public health. According to [3], Psychiatrists estimate that 20% of individuals infected with the coronavirus experience psychotic disease within 90 days. These technical advancements have proven beneficial in offering illuminating information that can help public health organizations and politicians enhance their interventions by better understanding the cause for depleting mental health. For example, it would give an idea about the increase in suicidal cases during the pandemic. Also, help the government officials to increase the social awareness camps.

In addition, researchers have used lexical-based approach to do twitter sentiment analysis. There hasn't been much use of deep learning or machine learning techniques. Second, the data from Twitter is unlabeled. Twitter has been labelled using clustering strategies. These strategies, however, are ineffective. In addition, most Twitter sentiment analysis are focused towards binary grouping. The tweets have been divided into two categories: positive and negative. The neutral tweets, on the other hand, are entirely overlooked. The use of hashtags to categorize content is standard practice. It is, however, not a real solution. A single tweet, for example, will contain several hashtags.

1.5 PROPOSED WORK

We in this project have proposed towards employing deep learning models in analyzing the Tweets towards mental depression. In addition, towards labelling the tweets, active learning approach was used. Active learning is a recently created method in which a portion of a dataset is manually labelled while a classifier labels the remaining data values. In addition, the model has been deployed in multi-classification of tweets as Positive, Negative and even neutral tweets too. We in this project have employed advanced deep learning models like Bi-LSTM, BERT

and XLNET towards classifying the level of depression from tweets and validate the best model. We hope that by creating a best deep learning model based on an individual's tweets, individuals who have mental illness can be identified quickly and handled appropriately. We also propose to reduce the number of suicide cases that have taken place because people cannot control their mental stress, loneliness, and hardship. With this model by sentimental analysis of tweets, there is possibility towards arranging awareness camps to help people deal with depression and anxiety with proper analysis.



Fig 1.2: Indirect Consequences On Mental Health

1.6 OBJECTIVES

The major objectives accomplished in the proposed work are as follows:

- Assessing the factors associated with the increased levels of mental health among people during COVID-19 outbreak.
- Labelling the Tweets using Active learning approach and classification as depressed, not depressed and neutral using classifier.
- Deploy and Validate deep learning models Bi-LSTM, BERT and XLNET on mental depression using tweets pertaining to COVID-19 data during lockdown
- Validate the model deployed with best accuracy on post Covid-19 lock down period for classification of depression using tweets.

CHAPTER 2

LITERATURE REVIEW

2.1 TRADITIONAL APPROACHES TO TWITTER SENTIMENT ANALYSIS

Twitter sentiment analysis is not a new idea. Twitter has been a widely used platform and thus the data has been used for a variety of research purposes. Although the twitter sentiment analysis has been used widely for many purposes, the approach towards the same has been ever-changing. In the world of technology, Machine Learning (ML) and Deep Learning (DL) has been dominating every sector and aspect. Twitter sentiment analysis is also being carried out using the ML and DL. ML models have been previously employed in twitter sentiment analysis. However, using deep learning models for determining the sentiment of tweets is new. In this section, the work that has used lexicon-based approach and ML beendiscussed.

In [4], the authors have tried to determine the mood of the public based upon the tweets that are made. The moods are categorized into six parts- tension, depression, anger, fatigue, confusion and vigor. Twitter sentiment analysis in this paper has been carried out based upon ‘profile of mood states’ which stands for POMS. It is a psychometric approach that consisted of a questionnaire. However, generating a vector of the above six sentiments using PMOS scoring limited and hindered the twitter sentiments. In reality, there are tweets that are happy and positive. Although the vector will have low values of the labels but we do not get a proper insight of the sentiment represented in the tweet.

Depression campaign has prompted many twitter users to come forward and express their mental state. We are focused on Covid-19 pandemic related tweets. One such work that has been situation specific was discussed in [5]. The tweets of this work were focused on ‘bell let’s talk’ campaign- a social awareness campaign for mental illness. For classification purpose, Bag of Words were used as features and the classifier used was SVC (Support Vector Classifier). The work also addressed the problem of imbalanced datasets. Oversampling technique- SMOTE- was applied to deal with class imbalance. SMOTE has been employed for oversampling. The oversampling technique generates random data so as to balance the data points of various classes. In a way, oversampling corrupts the dataset by introducing

random data. If powerful deep learning models were employed, normalization of data is carried out effectively with the help of the loss functions.

Machine learning models sometime outperform deep learning models in case of twitter sentiment analysis. This can be attributed to the data not being cleaned properly. In [6], the author has used the twitter data to get a customer review for the flight experience. Also, the complaints of the various airline companies have been jotted down. A strong feature extraction technique, TF-IDF has been used in the project. The work has used machine learning model called voting classifier. The classifier uses logistic regression and stochastic descent classifier. The classifier has outperformed a deep learning model, LSTM. A final accuracy of 0.791 has been recorded in this work. This accuracy can be attributed to a smaller dataset.

2.1 DEEP LEARNING FOR TWITTER SENTIMENT ANALYSIS

In the previous section we have discussed the various machine learning and lexicon-based approaches for twitter sentiment analysis. However, recently deep learning has been a new force in the world of technology. Deep learning has been utilized in many sectors for developing technology. Health sectors relied on AI for many procedures. In the near future, everything we do will be carried out by deep learning models. Twitter, as mentioned previously, has been used to gather the opinions on many matters. Be it a release of a movie or presidential elections, twitter data has been used to draw the conclusions on how the people feel.

In [7], an attempt has been made to identify racist tweets. It was done using the deep learning models like CNN and GRU. Also, the text embedding technique that has been employed in the paper is GloVe embeddings. GloVe embedding is one of the most powerful text embedding techniques. The GloVe embeddings stand for global vectors. The word embeddings are based on global as well as local statistics of corpus for creating the embedding matrix. This work aimed at predicting hate speech. It addressed the problem as binary classification. However, the twitter data might have tweets that are neutral with respect to hate speeches. The work has ignored considerably large neutral tweets.

One of the challenges that have been faced with producing desired results in twitter sentiment analysis, is labelling the twitter data. Ready to use twitter data are not available due to privacy issues. And the data required for our project was a recent one, that is, tweets made in pandemic. The data is not automatically labelled as depressed, non-depressed and neutral. Labelling the

dataset manually is time consuming and require human annotators. In [8], this issue has been addressed. The paper has used the hashtags for the labelling of a tweet. If a tweet has a hashtag 'happy', it is classified as a happy tweet and if a tweet has a hashtag 'sad' it is labelled as sad tweet. However, in many cases, a tweet has more than one hashtag. A single tweet can have both 'happy' and 'sad' as hashtags. And hence, the authenticity of the label is questionable. Also, the work has mentioned techniques like SVM, Naïve Bayes etc. for building the classifiers. Although machine learning has been used a lot in many sectors, deep learning is better with textual data because they have been used for many textual problems like sentence prediction, handwriting etc.

One of the papers that focused on using twitter data for depression was [9]. The paper used word based RNN and GRU models. The achieved accuracy for the model was 97% and 98% respectively. Although the model had high accuracy, as mentioned in the work, this was due to a smaller dataset of 13,385 rows. A smaller dataset has a smaller number of examples and thus might perform well on the data but will not work well with the other data. Also, the neutral tweets were ignored in this case.

There have been numerous drawbacks in the previously mentioned projects which are as follows:

- Twitter sentiment analysis has been carried out using the lexical based approach.
- . Labelling of twitter has been carried out using approaches like clustering. However, these approaches are not very effective.
- The twitter sentiment analysis carried out is usually a binary classification. The tweets are classified as positive and negative. However, the neutral tweets are completely ignored.
- Classification on the basis of hashtag has been a common approach. However, it is not an authentic approach. A single tweet can have more than one hashtag.
- Deep learning has been employed in the past but there has been a drawback pertaining to smaller dataset and also focused on binary classification only. Neutral tweets been ignored.

So based on the above-mentioned drawbacks, we here aim to bring up new and improved techniques throughout the steps carried out. Firstly, we have aimed to perform twitter sentiment analysis using purely deep learning models. The deep

learning algorithms are robust, efficient and most effective to solve a classification problem. The twitter dataset has been labelled using the novice approach of active learning. Active learning is a newly developed approach in which a part of a dataset is labelled manually and the rest of the data values are labelled using a classifier. Also, the project has kept a space for the tweets that are not related to our problem statement. These tweets are the neutral tweets. Hence, we have a multi-class classification problem at our hands. The project that has been described aims to overcome the drawbacks of previous work done as well as employ strong new classification techniques employing deep learning. These would be discussed in detail in the forthcoming chapter.

CHAPTER 3

SYSTEM ANALYSIS

3.1 PROBLEM DEFINITION

For any society to prosper, a population with a good mindset is of utter importance. A society where people are mentally sound has a better future than a society where people suffer from mental-illness of any kind. However, throughout the years we have noticed that mental health of people is given least importance. Mental-health issues like anxiety, depression, bipolar etc. are completely overlooked by the people. We have noticed that in almost all the countries, even government contributes very less to a depleting mental health.

2019 marked the onset of the COVID-19 pandemic. During the pandemic, lockdown was announced across the nations. The people had to be confined to four walls. However, adjusting to this lifestyle was quite challenging. Not stepping out in public and staying at their home straight for months proved to be quite challenging. In addition to this, people were witnessing mass deaths. Every day hundreds of thousands of deaths were reported worldwide. This affected the mental health of any people. Many companies had to lay off their employees to cover up for the loss they were facing. Unemployment also added to the decreasing mental health. All these factors contributed to the depleting mental health of people worldwide.

So accordingly, we in this work have aimed towards detecting the depleting mental health of people during the pandemic and also give an insight about how the mental health of people has improved post vaccinations and when the lockdown has been eased. To get an understanding of how the mental health of people has been affected during the COVID-19 pandemic, we concentrated on the twitter data. The tweets that were made during the pandemic were used as a source to understand how the mental health of people has depleted.

We are focusing on two major datasets in our project- the tweets that were made during the pandemic and the tweets that were made post vaccination and when lockdown was eased. The first part of dataset collected was focused between the months of March to December 2021. The second dataset was a very small dataset collected for 15 March, 2021. This dataset was used to just evaluate our model.

Using these datasets, we aimed to form a perspective of how has the mental health of people been during the lockdown. The mental health of the people has been a concern of all the people. People have been mentally pressurized by lack of jobs, witnessing deaths etc. We are

completely aware that every depressed person would not express himself/herself using the twitter. But we are taking a step to work for a section of people who have chosen to put themselves out there.

The project here focused on identifying the depressed, non-depressed as well as the neutral tweets of the users. Most of the classification models or twitter sentiment analysis ignore the neutral tweets, that is, tweets completely off-topic. Secondly, identify the section of the population that has raised awareness about mental health using twitter platform. This might include people who might have been under utter mental pressure and wanted to reach out to public for help. These might also include sections of people who were not able to visit their therapists and might have faced severe anxiety in the lockdowns. Finally develop a perspective about the tweets made in the month of March through December (when the pandemic was on peak) and how the mental health has been after vaccinations were started and the lockdowns were eased.

3.2 DATA PRE-PROCESSING

Twitter sentiment analysis is a text-classification problem. The efficiency of the text classification is co-dependent on how clean our text is. The tweets that people make have punctuations, links to other pages, different cases as well as emoticons. Also, the words like the, is, a, an etc. do not contribute to the sentiment of the tweet. Therefore, for an effective classification of our tweets into the depressed, non-depressed and neutral category we had to first pre-process our dataset. [10] has put together numerous steps that are followed during the pre-processing of twitter data.

NLTK stands for Natural Language Toolkit. It is a python library that has a wide range of lexical and corpora resources. The pre-processing of our data was entirely done using the NLTK package. It consists of suite of text pre-processing libraries for tokenization, parsing, stemming, tagging, stop words etc.

In accordance with [11], the first step for pre-processing our twitter dataset was to include the tweets that were made in English language. The 'lang' column in our dataset specified the tweet language and we filtered out tweets in which the lang column had value other than 'en'. After we had set of only English tweets, we further proceeded to remove anything in the tweet or 'text' column that could contribute to the noise for sentiment classification.

After we filtered out only the English tweets, we proceeded to remove the stop words that

contributed as noise. Stop words consists of the words that are common and do not contribute to overall sentiment of the sentence. The NLTK package consists of the stop words library that contains all the stop words. We removed the stop words from the text column of our data set to get a fairly clean tweet. The stop words that were removed from our tweets contained only English language stop words because we filtered out non-English tweets.

Besides removing the stop words, the punctuation and links were removed from the sentence using regular expressions. That which consists of links that direct to any other pages and the links start with 'https'. The punctuation as well as the links were removed from the 'text' column of the dataset.

Retweets are defined as the tweets that have been treated again by a different user. If we include retweets in our dataset, we will have redundant copies of a single tweet. Therefore, for the purpose of tweets sentiment analysis, we removed retweets from our dataset. The retweet starts with RT and therefore we removed any 'text' column that started with RT.

Stemming as well as lemmatization are the techniques to link the word to their root words. It is the way toward decreasing words to their stems. Most of the stemming calculations are based on some predefined rules. For example, they improve the entirety of the previously mentioned 'Regreted' conversations to 'Regret.'

A Lemmatizer is like a Stemmer, yet it keeps the word's phonetics in context. For example, 'running,' 'ran,' 'runs' comes under the same category 'run.' Words like 'do,' 'does' come from a common word 'do'.

In text-classification stemming and lemmatization help us to link the word to their link words. This helps to reduce the noise that will be created by similar words and also reduce the length of the text. The difference between stemming and lemmatization is that in stemming the root word might not be the actual word while as lemmatization the root word is actually an actual word. In our project we have used "PorterStemmer" for stemming and "Wordnetlemmatizer" for lemmatization. Both these are a part of NLTK package.

Tokenization is another important part of text pre-processing. Breaking a text into tokens ease the text-classification problem. For tokenizing tweets, NLTK package has tokenize library which gives access to various tokenizers. For tokenizing the tweets, we have used the "TweetTokenizer".

Finally, the last step of pre-processing is dropping all the columns that did not contribute to the sentiment of our tweet. We only kept the 'id' and 'text' column for this purpose. Our final

dataset consisted of two columns- 'id' for the id of tweet and 'text' for the content of the tweet.

3.3 LABELLING THE TWITTERDATA

The datasets that have been used for this project were built from the scratch. The models that we have implemented for our problem statement required our dataset to be labelled. To label the pre-processed dataset, “active learning” has been employed. Active learning is a subset of the machine learning. The core theory of active learning is that if a learning algorithm can select the data from which it needs to learn, it can outperform conventional approaches with significantly less instruction. The algorithm used in active learning interactively labels the unlabeled data. In active learning, a part of the dataset is selected and labelled manually. This labelled dataset is then required to label the rest of the dataset. However, the ‘text’ column containing tweets cannot be used by any classifier. Any classifier needs a vector or numerical data to work with. Therefore, labelling the dataset was divided into three major tasks –

- 1) Label the part of dataset
- 2) Creating the ‘text’ column to a vector and
- 3) Choose a classifier to classify unlabeled tweets.

3.4 TF-IDF VECTORS:

TF-IDF stands for term frequency inverse document frequency. TF-IDF is a statistical measure for the relevance of a term in a document. It is a way to convert a text into a vector form. TF-IDF has an edge over other techniques because it considers term frequency as well as document frequency into consideration. While recovering any data, TF-IDF, is a mathematical measurement planned to reflect how important a word is to an archive in an assortment or corpus. It is regularly utilized as a weighting factor in data recovery searches, text mining, and client displaying. [12] has described in detail what TF-IDF actually stands for and. The TF-IDF esteems build relatively to the occasions a word shows up in the report. It is balanced by the number of records in the corpus that contain the word, which assists with adapting to the way that a few words show up more now and again when all is said in done. The TF-IDF is quite possibly the most famous term-weighting plan today.

Term frequency is the number of times a word will occur in the document. The inverse document frequency is number of times a word has occurred in a set of documents. If a word occurs many times in the documents, the IDF approaches zero. A word that is very common and has occurred in many documents has TF-IDF approaching zero.

$$TF-IDF(t, d, D) = TF(t, d) * IDF(t, D)$$

$$TF(t,d) = \log(1 + \text{frequency}(t,d))$$

$$IDF(t,D) = \log(N/df+1)$$

$df(t)$ = occurrence of t in document

The 'text' column was created into TF-IDF vectors. A new column 'Vectorized' was added to the dataset containing the TF-IDF vectors of the tweets. These were used to label the tweets.

3.5 SUPPORT VECTOR CLASSIFIER:

The support vector machine classifier aims to discover a hyperplane in an N -dimensional space that finds and classifies the fine data points for the classifier. [13] has explained the working of a support vector machine which is a basis for the support vector classifier. To label the tweets based on the manually labelled part of dataset, we needed to select a classifier. A classifier tries to understand the class of a data point after learning from the training data. We have used the tf-idf vectors generated earlier for the purpose of classifying the tweets as depressed, non-depressed and neutral. The classifier uses this tf-idf representation to link a tweet to a category.

For classification, **Support vector classifier** (SVC) has been used in this project. A support vector classifier aims to find a hyperplane in an X -dimensional space that partitions the different classes. There are many choices of hyperplanes that can separate the various classes. However, the aim of SVC is to find the plane with maximum distance between the classes.

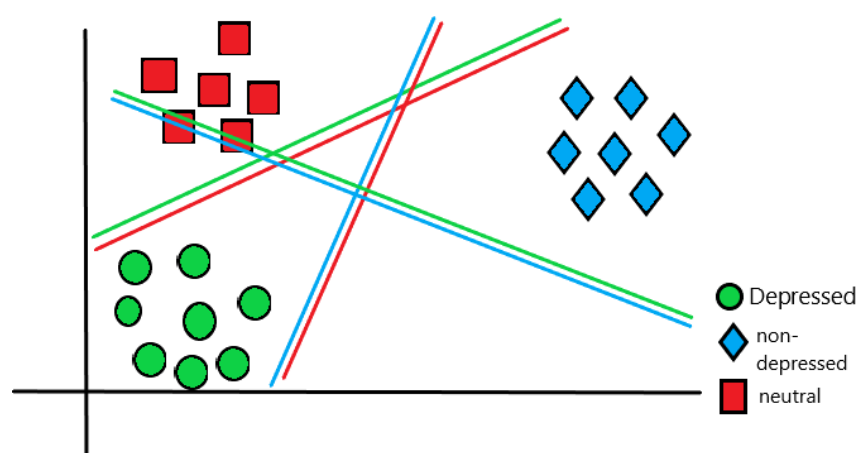


Fig 3.1: Hyperplane separating the different classes

SVC was used as a part of active learning algorithm. Linear kernel was selected to separate the depressed, non-depressed and neutral data points. The classifier was trained using the manually labelled dataset and the unlabeled data points were predicted using the trained classifier.

3.6 DEEP LEARNING ALGORITHMS FOR TWITTER SENTIMENT ANALYSIS

In the project, the aim was to get an insight about how the mental-health of the population has been affected during the pandemic and how vaccination might have improved it. So, for this purpose, we used the social media platform “twitter”. Our focus is on English language tweets. Following the pre-processing stages and the labelling of our dataset, the problem of classifying tweets as depressed, non-depressed and neutral was addressed. In our dataset, we have labelled the tweets into three classes- neutral (0), depressed (1), non-depressed (2). We have employed three major deep learning models which are Bi-LSTM, BERT, XLNET. The working as well as architecture of these algorithms have been described in detail in this section.

3.6.1 BIDIRECTIONAL LONG SHORT-TERM MEMORY (Bi-LSTM)

RNN (recurrent neural network) are a step ahead in neural network to clone our human brain. Recurrent neural networks are able to retain the information. This means that while dealing with a new input they try to give the output in accordance of what has been encountered before. e.g., in sentence generations we need to predict the next word. For this purpose, our neural network needs to remember the previous words. RNN remembers every detail through time with the help of the hidden layers.

In RNN the network is created with loops in them which retain the information and the input is presented as sequence to these loops. The RNNs provide equal weights and biases to all the layers creating dependent activations. The complexity is also reduced as every layer does not have to memorize. All the layers are then joined together into single recurrent layer.

LSTM stands for Long Short-Term Memory. LSTM are a type of RNN. However, unlike RNN they can choose to forget some information. RNN are not able to retain old information. Through loops they learn and remember all the new information whether or not that information is relevant. However, in some cases the information we require might be old. Since all the information is not relevant, we might need our system to forget that information. This

means that the information that is old needs to be remembered and the information that serves little to our problem needs to be dropped. The information that is not necessary or has less relevance towards providing desired outputs are removed. The architecture helps drop the information that does not serve our purpose. Therefore, LSTM can be defined as an altered version of the RNN. LSTMs are preferred for classification and prediction problems.

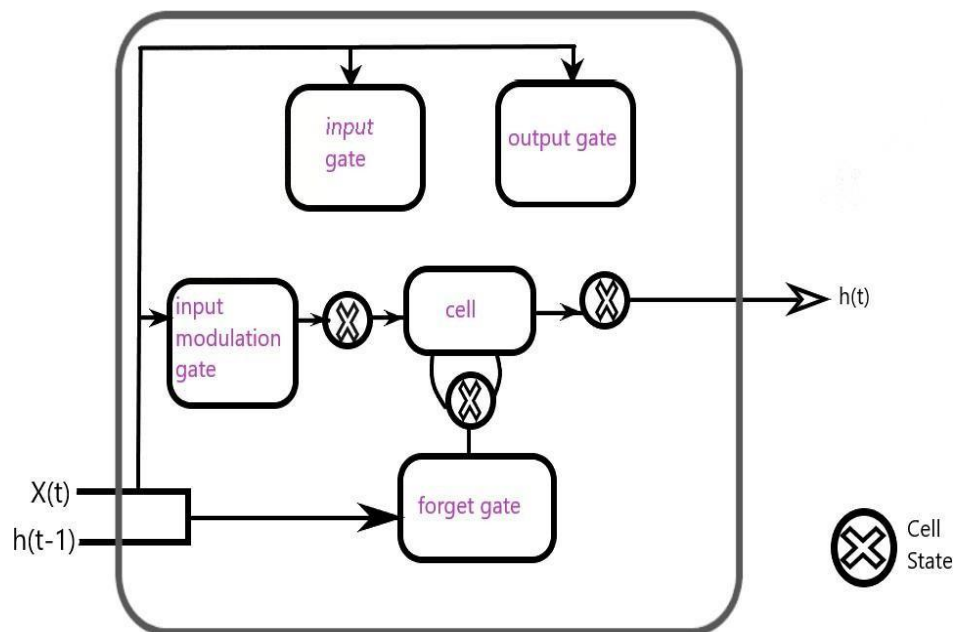


Figure 3.6.1 – Architecture of LSTM

The architecture of LSTM as described in [14] has cell state, input gate, forget gate, output gate and cell state. The cell state is the long short-term memory in the architecture. This means that all the information is stored in the cell state. The cell state is altered by the forget gate. The input first passes through the forget gate. Forget gate in other words is called the remember vector. When the result of the forget gate is 1, the information is to be retained. However, if the result is 0, the information is to be dropped. The input gate determines the information in the cell state. The input gate decides which value from the input should alter the memory. The input gate is nothing but an activation function. An input gate is a sigmoid function. The sigmoid function has a range 0 – 1. The input gate decides which information can be added to the cell state. It cannot determine which information is to be dropped. The sigmoid function is then joined by a tanh activation function. The output gate gives the results

of the cell state in a modified way. There are two major activation functions—sigmoid having values from 0 through 1 and tanh having from -1 through 1. The forget gate is responsible to find the details that needs to be dropped from the memory. The sigmoid function is used for this purpose. The forget gate considers the previous state as well as the input and gives a value between 0 and 1. The values closer to 0 are forgotten and the values approaching 1 are kept. The output gate decides the output considering the memory as well as the input block. Sigmoid and tanh functions decide the values and weightages to the values respectively. Below are the equations for the LSTM gates:

$$\text{input}_t = \sigma(W_{\text{input}} \cdot [h_{t-1}, X_t] + b_{\text{input}})$$

$$C_t = \tanh(W_c \cdot [h_{t-1}, X_t] + b_c)$$

$$\text{forget}_t = \sigma(W_{\text{forget}} \cdot [h_{t-1}, X_t] + b_{\text{forget}})$$

$$\text{output}_t = \sigma(W_{\text{output}} \cdot [h_{t-1}, X_t] + b_{\text{output}})$$

$$h_t = \text{output}_t * \tanh(C_t)$$

where, h_t = cell state

input_t = input gate

output_t = output gate

forget_t = forget gate

C_t = initial cell state

Bidirectional LSTM is a transitional form of the traditional LSTM. Bi-LSTM is simply combining two LSTM networks Bi-LSTM improves the sequence classification as compared to LSTM. This architecture provides way to have both forward as well as backward information about the sequence. Bidirectional LSTM improves the context or the sequence prediction power because they have access to more information as compared to LSTM. [15] has put forward the concept of using the Bi-LSTM for the purpose of twitter sentiment analysis. We have also employed Bi-LSTM for the same purpose in the project presented in this document.

3.6.2 BIDIRECTIONAL ENCODER REPRESENTATIONS FROM TRANSFORMERS

In the evolving science of deep learning, transformer model is a robust and recent approach to

sequence-to-sequence learning. Sequence-to-sequence learning allows a sequence of input from one domain to produce an output in another domain. Transformer models are sequence-to-sequence. A transformer architecture has stack of encoders and decoders. In a transformer architecture, self-attention is used on encoders side and attention on decoders side. Self-attention allows an input to interact with the attention of the other inputs while as attention puts focus on the inputs to produce the outputs. Sequence-to-sequence learning was developed as a part of converting one language to another e.g., English to German or French to German.

A sequence-to-sequence model maps the input to a n-dimensional vector in the encoder. The decoder then takes this vector and transforms it into an output sequence. BERT stands for Bidirectional Encoder Representations from Transformer. BERT is a transformer model that utilizes the transformer architecture of attention mechanism. BERT is used to build a language model and hence, focuses more on the encoder working.

Transformer models were introduced to the world of deep learning in the year 2017. This model architecture became famous for its robustness. Transformer model performs Seq2Seq operations in an efficient manner. The basic architecture of a transformer model consists of two units- encoder and decoder. What makes a transformer model different is that it uses long term dependencies to solve sequence-to-sequence problems.

The problem of long-term dependency is an everyday problem. Long term dependency refers to output depending upon the previous examples. RNN are a step towards solving the problem of long-term dependency. However, they have not been very successful. Transformer models are the most successful models till date that enable us to solve the problem of long-term dependency. Apart from the advantage that is gained by the encoder-decoder architecture, the transformer models solve this problem with the concept of attention mechanism.

An attention mechanism is a method of informing a system that particular words are important with respect to the output. In our case, e.g., sad is an important word with respect to detecting depression. This means that the word, 'sad' will be marked as a keyword. The attention mechanism also enables the decoder to look back at the sequences before giving an output. In case of BERT, an encoder passes the keyword to the decoder. This helps solving a sequence-to-sequence problem quicker and with much efficacy. The BERT model has been documented in [6] where the underlying architecture as well as working is put forward.

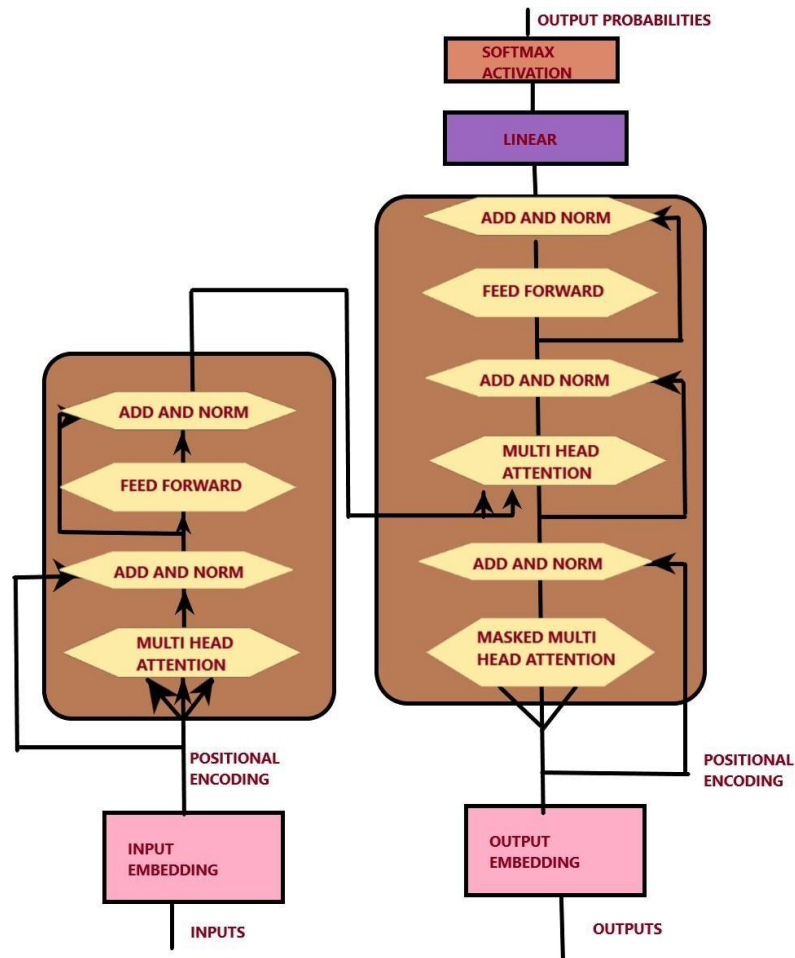


Figure 3.6.3- Architecture diagram of BERT

Fig 3.6.3 gives an insight about the architecture of the BERT. Right from looking at the diagram, we can get an idea about the stacked nature of this architecture. The left side of the diagram represents an encoder while the right side of the diagram represents a decoder. In both the encoder as well as the decoder, the different modules are stacked upon one another. The modules in encoder are – Multi head attention, Add and norm and Feed forward. In decoders there is an additional module masked multi head attention. One of the exclusive features of the architecture is the positional encoding. In BERT model, there are no recurrent neural networks.

Therefore, remembering the information is impossible for the model. The positional encoding is a vector representation that has the same dimensions as that of the embeddings. The positional embedding value is appended to our embedding value and then passed as input to the encoder as well as decoder. Hence the positional embeddings keep track of the location of the values in the data.

The add and norm layer in the BERT architecture is a normalization layer. After passing the inputs of encoder and decoder through the various modules, they pass through the add and

norm layer. The input of this layer is first passed through a residual connection. This prevents passing an input through a non-linear activation. Non-linear activations bring about erratic nature in the gradients. Therefore, the gradients are passed through an extra or residual connection. Next, they are passed through a normalization layer which is present as a LayerNorm function in python. When a network learns the weights as well as the biases, they are updated. This might cause changes in the model outputs. These changes are referred to as 'internal covariate shift'. This phenomenon occurs while passing the data from one layer to another layer. Hence, this calls for the normalization of the input.

A feed-forward module enhances the working of the BERT model. A feed forward layer helps the model to better understand the context of the word in respect to its surrounding. The feed forward layer is fully connected or the dense layers in the BERT architecture. In a simple neural network, the attention layer would be a hidden layer and the feed forward layer would be an output layer. Although the purpose of feed forward layer is not clear. But there are certain points. First, is that the feed forward layer looks at a particular word and not the entire weight. 2/3 of the weights of the encoder in the feed forward layer and just 1/3 in the attention layer.

Bert uses multi-head attention layer in its architecture. This means that many attention mechanisms are running in parallel. In BERT, various attention mechanisms are learnt by the model and these attention mechanisms run in parallel to one another. This is known as multi-head attention layer. In the decoder, an additional attention mechanism called, masked multi-head attention mechanism is added. This means that the data is masked or hidden in this case.

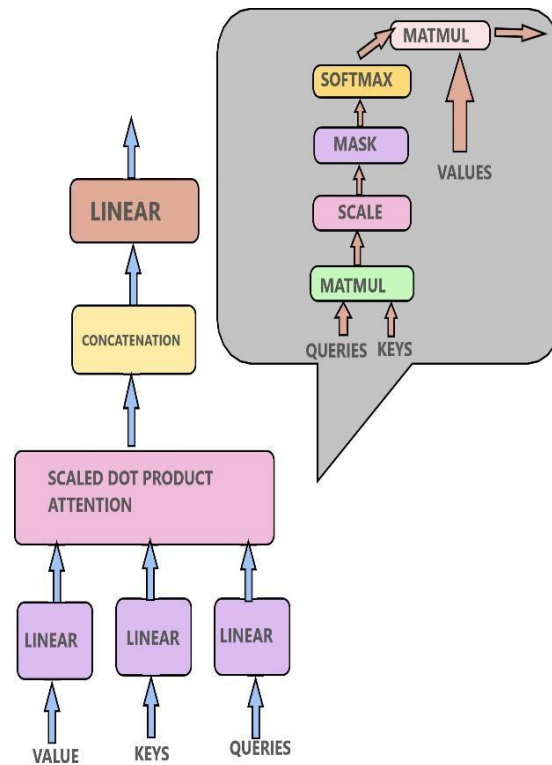


Fig 3.6.4 Multi-head attention mechanism

Based on the number of encoder and decoder arranged in a BERT model, there are two models – BERT_{base} and BERT_{large}. BERT_{base} has 12 encoder-decoder sequences stacked while as BERT_{large} has 24 encoder-decoder sequences stacked. The BERT model is pre-trained on a large set of unlabeled data.

In the previous model discussed, LSTM, the input is read in one direction. However, to define BERT model, we can say that it is un-directional. The BERT model reads the input at once. Therefore, the BERT model gets a complete context of the words in text. The model can understand the relation of the word with a word that comes before and that comes after.

The attention is mathematically defined as follows:

$$\text{Attention}(q,k,v) = \text{softmax}[(q * k^t) / \text{sqrt}(dk)] * V$$

Where, q = matrix containing set of queries

k = keys

v = values

d_k = dimensions of keys

BERT uses multiheaded attention and for the mathematical representation for same is:

$\text{MultiheadAttention}(q, k, v) = \text{concatenate}(\text{head}_1, \text{head}_2, \dots, \text{head}_n) W^o$

Where, $\text{head}_i = \text{Attention}(qW_i^q, kW_i^k, vW_i^v)$

q = matrix containing set of queries

k = keys

v = values

W = weight

The principle of self-attention is the foundation of the BERT transformer model. The concept of self-attention as well as attention is the same- to find what part of input is important to get the output. However, in case of self-attention, the data is to be looked at itself (self). For example, while converting an English sentence to a French sentence, in case of self-attention, the sentence in question looks for the words that are important for providing the results. As mentioned earlier, BERT model has multiheaded attention mechanism. This means that in BERT architecture, a number of self-attentions run in parallel at the same time.

In reference with [16], BERT is a masked language model. During training the model, some of the words in a text are masked and the model is aimed to guess these words. For the purpose of tweet classification, we use pre-trained BERT model. The BERT models are trained on a large set of unlabeled data. For the purpose of text classification- a) the text data is converted into tokens b) a classification layer is added on top of the model. Every input data in BERT model has a CLS token at the start of a sentence and a SEP token at the end of the sentence.

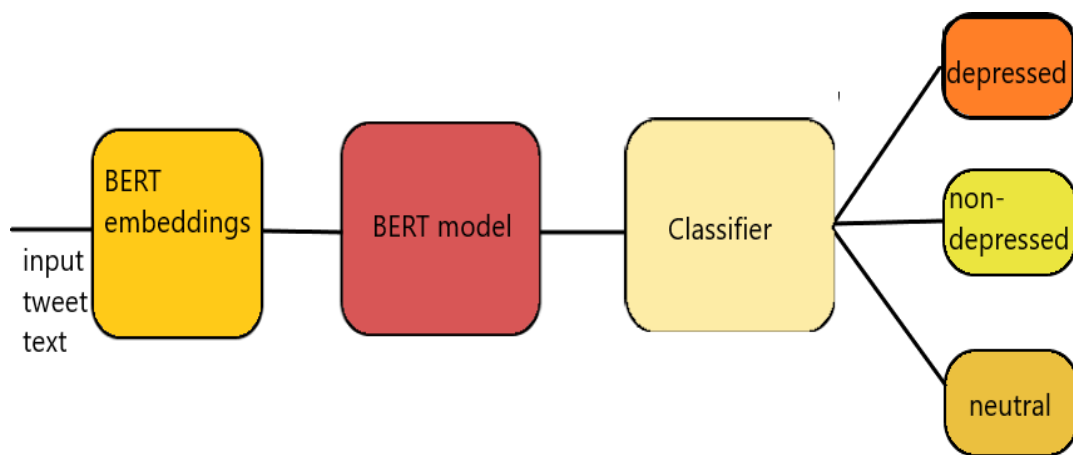


Fig 3.6.5: BERT model for classification of tweets

Looking at the architecture of BERT as well as the above figure, we know that the textual data has to undergo embedding before passing through encoder as well as decoder. Bert embedding is a type of word embedding in which a word is represented as a vector. There are two main tokens in the BERT embedding – SEP and CLS token. The SEP token is used to separate the two sentences and the CLS token is present at the start of the text. For the classification problem, like in the project that is presented, the CLS token is crucial. The BERT embeddings can separate every word or sentences into tokens or break a word into parts represented by “##” in the beginning. The last resort is to break the word into characters and also represented by “##” in the beginning.

3.6.3 XLNET

Bert is an autoencoder language model. As mentioned previously, some of the tokens in Bert model are masked during fine-tuning. Although BERT is one of the most effective models but it has its own disadvantages. One of the major disadvantages of BERT model is that, it considers the masked tokens independent of one another. e.g., in a sentence “Maurine drinks a lot of alcohol and so she has liver sorosis” if words liver and sorosis are masked the BERT model tries to predict liver and sorosis independent of one another.

As explained in [17], XLNet is quite similar to the BERT model. It is defined as a generalized autoregressive model. An autoregressive model aims to predict the next word in the sentence using the context of the previous word. XLNet has been successful in using the plus points of Autoencoder as

well as autoregressive models while avoiding their downsides. As we know a word can depend on what word comes before as well as after it. XLNet is an autoregressive model that is non-directional in nature. This means that the predicted word depends on the words around it. The non-directionality of XLNet autoregressive model is achieved by introducing a new concept- “permutations”. Let us assume that a sequence has tokens as $[t_1, t_2, t_3, t_4, t_5]$. Now there are $5!$ i.e., 120 Permutations that are possible for this sequence. Now, if we want to predict the t_5 token, there are 5 scenarios t_5 present at 1,2,3,4 and 5 positions. Using these scenarios, the model retains the information necessary for the prediction. Using this approach, an attempt to build **permutation language model** has been made.

During the training of the XLNET architecture, two major concepts of the Transformer-XL model are focused upon. As explained in the previous section, a Transformer architecture was a fresh approach towards solving the problems bound to sequence to sequence. Previously, RNN were used to solve the Seq2Seq problems. RNN networks operate on the principle of learning whatever passes through them. However, the transformer models are based upon the concept of attention mechanism. Attention mechanism is a concept of alerting the architecture that a word is important and requires attention. Coming to XLNET, it follows the architecture of the transformer model but unlike BERT masking of the input is not present in the model.

XLNET draws up two concepts from the BERT model- 1. Positional encoding and 2. Segment recurrence. As we know, XLNET does not have a memory or a loop architecture to retain the information as in case of the RNN networks. The model however has to keep a track of the tokens in a sequence. Since, XLNET was developed to solve the problem of Seq2Seq, the input of the model is a sequence. Positional encoding is a technique to keep a track of the tokens in a sequence. Positional encoding is a vector that keeps a track of the position of tokens in a sequence. When a sequence is to be passed through a XLNET architecture, it has to undergo word embeddings first. The word embeddings are performed by the tokenizers. After, the input is embedded, a positional embedding vector is appended to the word embedding and after this, the data is input to the XLNET architecture.

Segment recurrence is a technique to reuse the memory for every segment. The XLNET architecture caches the hidden state of the very first segment in the memory. And this is repeated for all the layers of the architecture. This allows the system to update the attention at regular intervals. If a system has a depth D , the segment recurrence takes place D times for

every level. Segment recurrence can increase the contextual understanding of the architecture and helps achieve the longest dependency possible for a model.

Passing the permutations during the model training allows XLNet to gather more dependencies in the data as compared to the BERT and other models. The model parameters are passed on to all permuted factorization orders. This allows the model to get a non-directional context of the input data as well as remove the discrepancy caused by masking data in BERT. The encoding scheme as well as the architecture of the XLNet resembles BERT. XLNet has two main types with respect to their architecture- XLNet-base have 12 encoder-decoder sequences and XLNet-large have 24 encoder-decoder sequence.

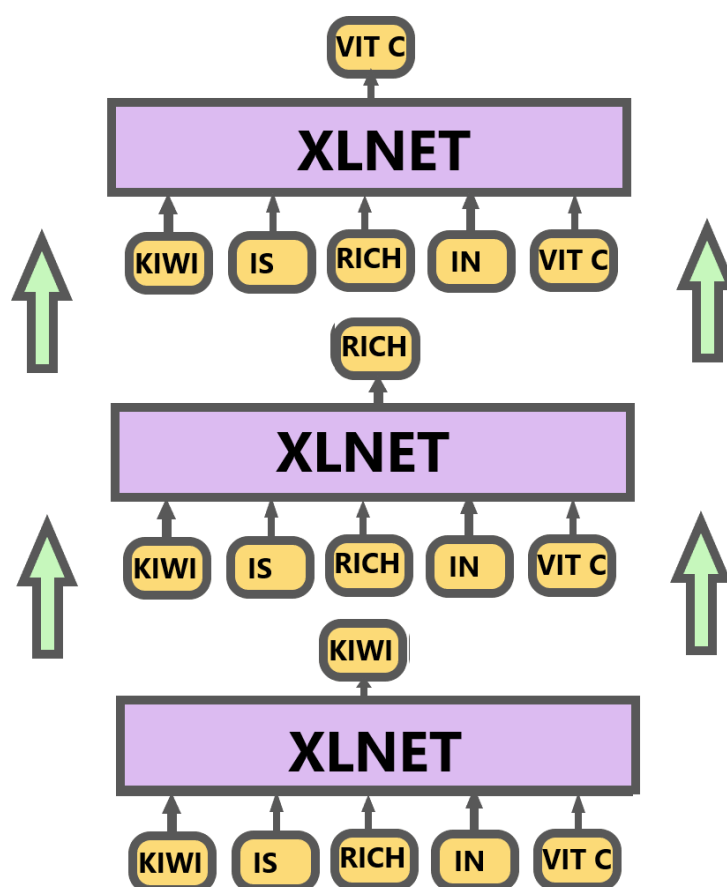


Fig 3.6.6: Working of XLNET

The working of XLNET for the sequence is explained in the figure above. As discussed in the article [18], the XLNET is a permutation-based model. In this model, the probability

is maximized by taking into consideration all the possible permutations in the factorized order. When we take all the possible permutation into order, we can get a context of the tokens that are prior to a token as well as tokens that follow a particular token.

The following mathematical equation explains the idea behind the XLNet model:

$$\Omega = \operatorname{argmax}_{\Omega} [\mathbb{E}_{z \sim Z} [\sum_{t=1}^T \log [\Pr(x_{z[t]} | x_{z[<t]})]]]$$

Where, Ω = model parameter

x = token

T = sequence length

Z = permutations

$z_t = t^{\text{th}}$ element in permutations Z

The equation that is given above explains the principle that is used in XLNET model and that has enabled it to outperform the rest of the models, that is, the permutation language model. However, as we are already aware XLNET is a generalized autoregressive model. The mathematical equation governing the AR model is given below

$$\max_{\theta} \log p_{\theta}(\mathbf{x}) = \sum_{t=1}^T \log p_{\theta}(x_t | \mathbf{x}_{<t}) = \sum_{t=1}^T \log \frac{\exp(h_{\theta}(\mathbf{x}_{1:t-1})^{\top} e(x_t))}{\sum_{x'} \exp(h_{\theta}(\mathbf{x}_{1:t-1})^{\top} e(x'))},$$

Where, $h_{\theta}(\mathbf{x}_{1:t-1})$ = context representation generated by neural networks

$e(x)$ = embedding of x

\mathbf{x} = text sequence

The main aim of the equation mentioned above is to maximize the likelihood during the forward autoregressive factorization.

CHAPTER 4

SYSTEM DESIGN

4.1 SYSTEM ARCHITECTURE

This chapter discusses the System Architecture of Mental Depression during COVID-19 using Twitter Sentiment Analysis. Also, the chapter discusses Data Flow Diagram and Use Case Diagram of the system developed

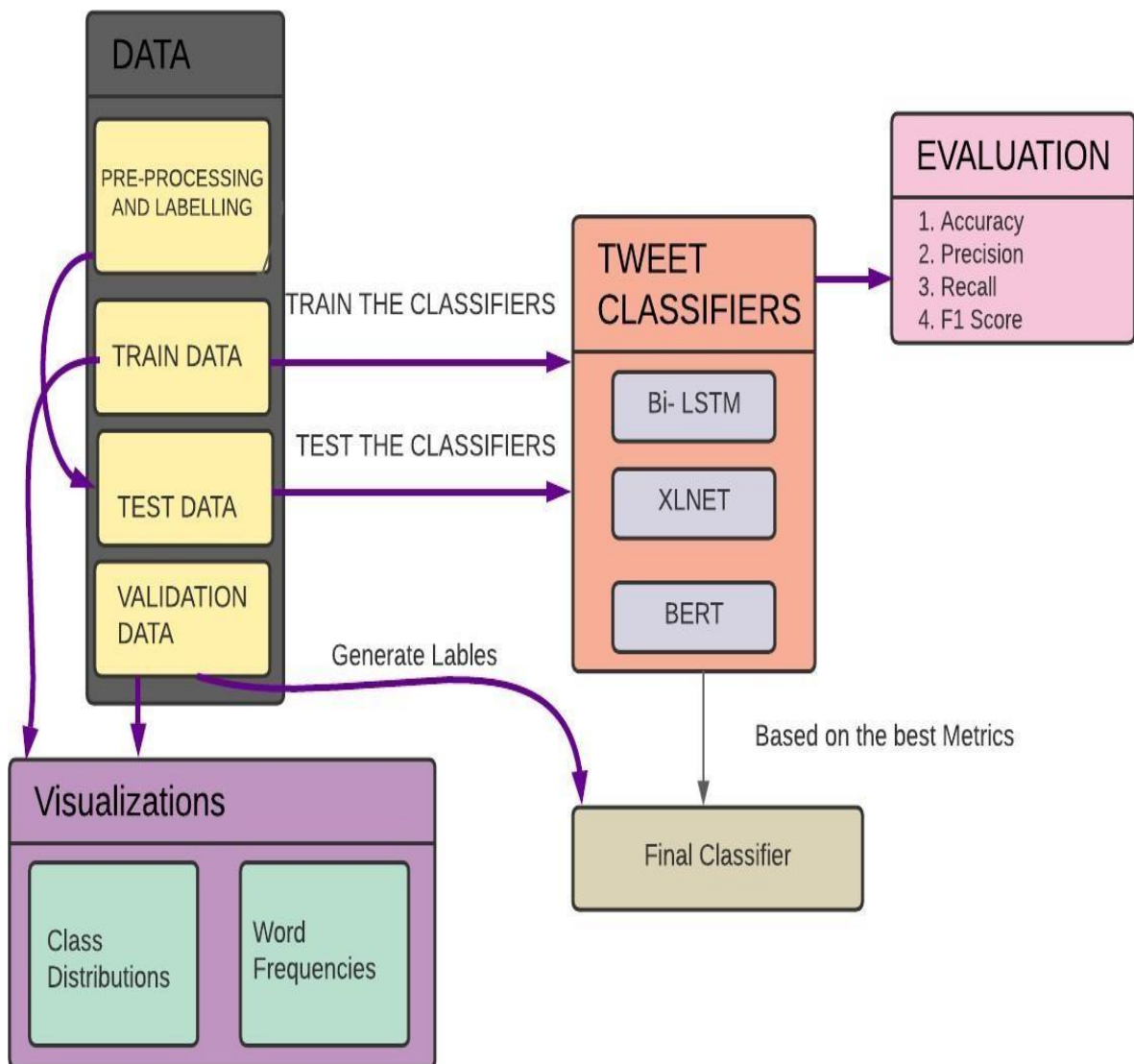


Figure 4.1: System Architecture Diagram

The first and foremost step is to analyse and understand the problem at the ground level to create the system architecture. This step is crucial and is the basis of any project which is to be completed. With the help of this, we will be able to define and execute our system more effectively. Once we have understood our data, the next step is to find a raw dataset that is then cleaned and pre-processed to improve our model's accuracy. This cleaned data is split into the training and testing data. Using this trained and testing data, we will train and test the classifiers on our Bi-LSTM, BERT, and XLNET model, respectively. The next step will be to run the evaluation metrics and calculate the accuracy, precision, recall value, and F1 score. Based on these metrics, we will opt for the classifier which has the highest accuracy. After this, we will create a Word Cloud visualization and analyse the model.

4.2 DATA FLOW DIAGRAM

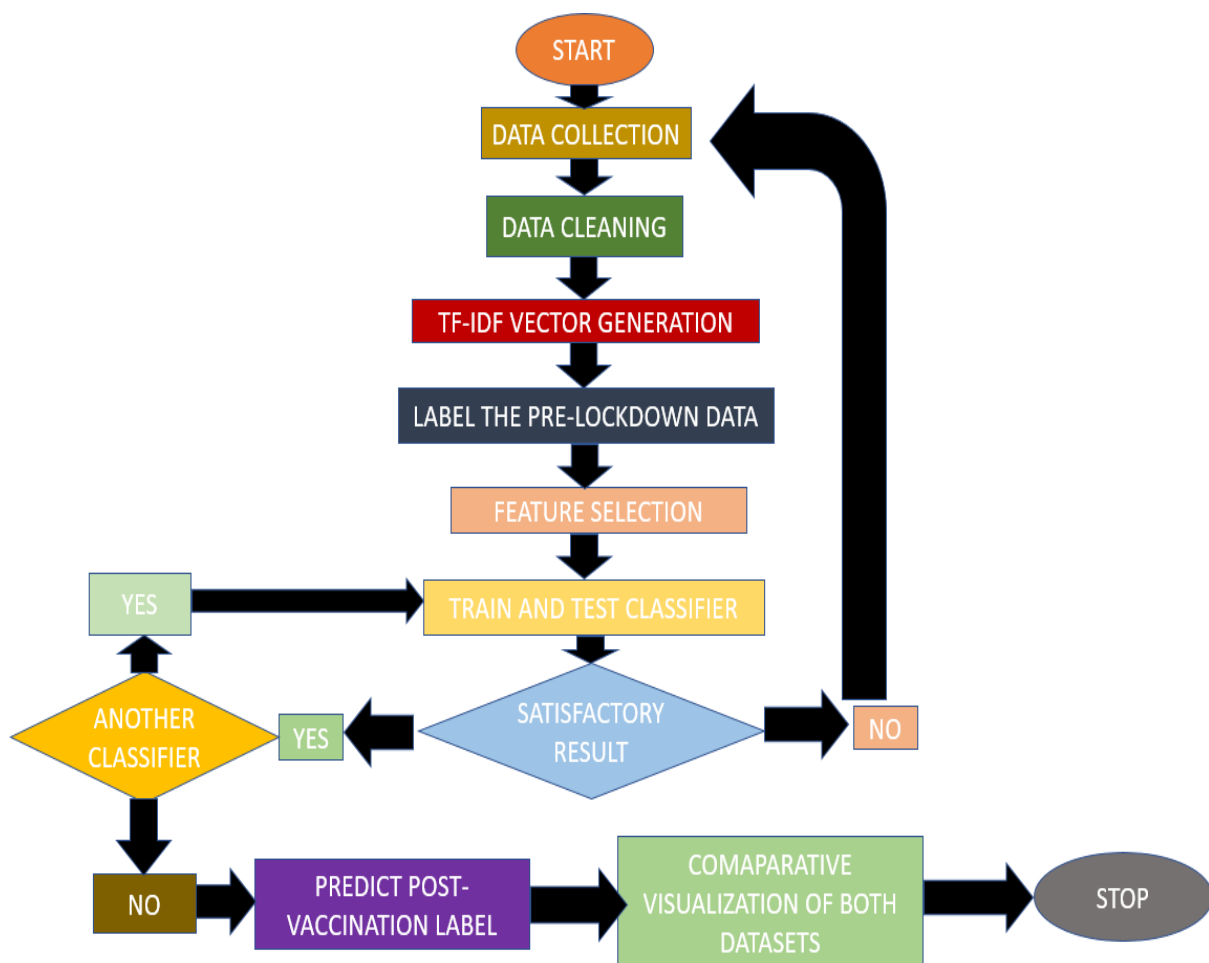


Figure 4.2: Data Flow Diagram

The first step is to use Tweepy Library and collect two Twitter datasets- a) The tweets that were made when the pandemic was at the peak, i.e., from March 2020 through December 2020 and b) The tweets that were made post the vaccination around the world, i.e., in March 2021. The next step is to pre-process the dataset using the NLTK package. It consists of a suite of pre-processing text libraries for tokenization, parsing, stemming, tagging, stop words, etc. The next step is to convert a text into a vector form using TF-IDF vectorization and then label the dataset to create a classifier and classify the un-labelled dataset. Now, we would train and test the classifier on our model and check if satisfactory results are achieved or not. If not, then we would run the classifier on another model. If yes, we would create a visualization and compare it with the post-vaccine dataset and conclude.

4.3 FLOWCHART

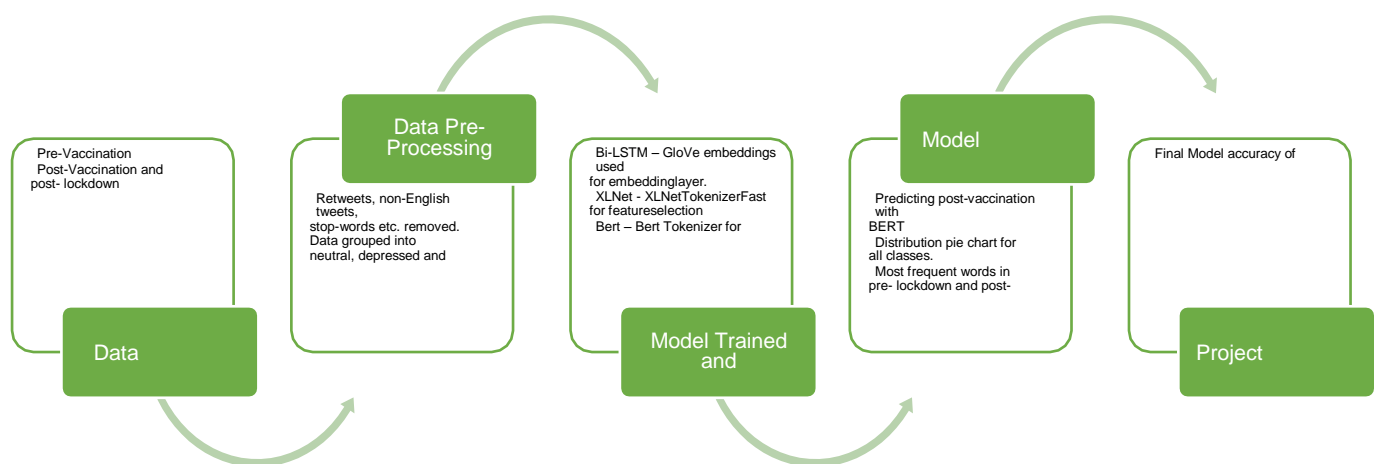


Figure 4.3: Flow Chart

The primary step is to collect both the pre-vaccination and post-vaccination datasets using the Tweepy library. The next step is to pre-process and label the dataset. Here we remove the retweets, non-English tweets, stop-words, etc. Also, the dataset is categorized as a neutral, depressed, and non-depressed dataset. The next step is to test and train the model and fine-tune it. For Bi-LSTM, we use GloVe's embedding layer; for XLNET, we use XLNETTokenizerFast

for extracting its features, and for Bert model, we use Bert tokenizer for feature selection. Now the next step is to evaluate our model. We predict the post-vaccination dataset with the Bert model, create a pie chart for all the classes and analyse the most frequent words used in the pre-lockdown and post-lockdown dataset. The final step is to select the model with the highest accuracy.

CHAPTER 5

CODING AND TESTING

The project that is being discussed in this document is aimed to detect the mental state of people during the pandemic. For the same, the focus has been on tweets and twitter sentiment analysis has been carried out to get a perspective about how the mental health of people has been affected during the covid-19 pandemic. As discussed previously, the twitter sentiment analysis has been carried out using the deep learning models. The deep learning models that we have implemented are Bi-LSTM, BERT and XLNET. The machine learning as well as deep learning architecture implementation is relied upon two main things – implementation language and implementation environment.

5.1 PROGRAMMING ENVIRONMENT

For the data collection, data pre-processing, the labelling of dataset as well as generating word cloud, we have used Spyder. Spyder is a scientific open-source environment and is written in python. Spyder is considered one of the best environments to work with scientific related stuff. It is easy to use and is robust. Spyder is a part of the Anaconda. Using Spyder we can edit the scripts, debug our code as well as test our code. However, for the implementation of the deep learning algorithms we required higher processing power. A CPU will take a long time in case of deep learning models. For the same, we have used Google Colab. Colab is a cloud-based runtime environment that resembles Jupyter notebook. Colab allows free access to GPU in the cloud. Although it has limitations for the GPU usage but for our project it sufficed.

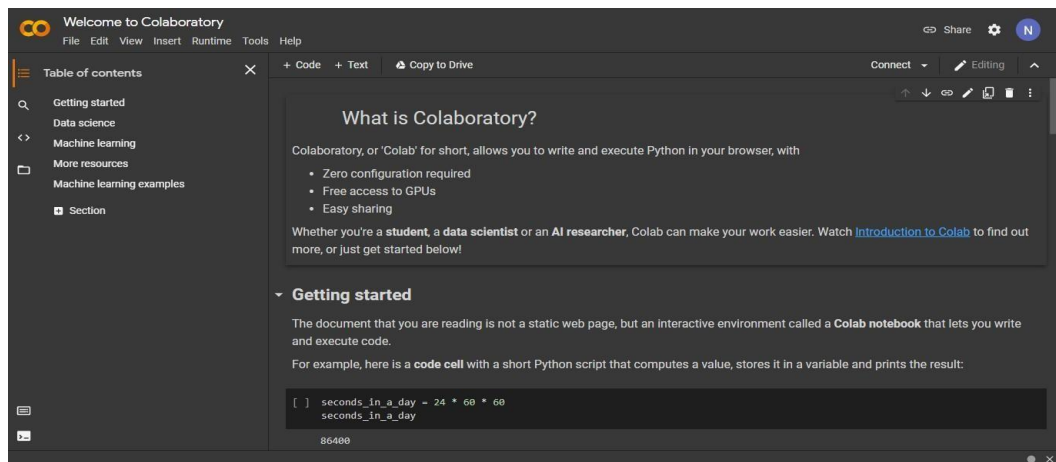


Fig 5.1: Google Colab environment

The above figure shows the google colab environment. It allows easy editing, debugging and testing of the code. The output of the cells can be saved to be viewed later. The colab provides CPU, GPU as well as GPU runtime settings. The files can be mounted on to the drive or can be uploaded/downloaded from the system. The google colab allows easy sharing of the code and makes it accessible at all points.

5.2 PROGRAMMING LANGUAGE

While implementing any technology related products, one of the basic questions is 'In what language should be implemented it?'. Various technologies have various languages that are suitable for it. Let us consider mobile applications. The languages that are most suited for developing a mobile application are java and Kotlin. We have experienced very few chances where coroutines are used to develop mobile applications. Likewise, data science and field of ML and DL also require a programming language.

Artificial Intelligence has a vast number of algorithms. Therefore, the programming language to be used for the same should be user friendly and easy to use. One of the most laid-back programming language is Python. Whenever, deep learning is heard, python is accompanied with it. Python has been used in deep learning since the very beginning. It is considered as a breakthrough in the programming languages.

Python is a general-purpose programming language. It is everything that a data scientist would ask for. It is used very commonly nowadays. Python has a very simple syntax which saves up a lot of time debugging the syntax while implementation. One of the things that attracts users

towards the language is a wide variety of libraries. Python has a large number of libraries which can be used by the users whenever required for the tasks.

In this project, the programming language that has been used is python. Owing to its advantages it was the best choice available. The version of python that has been used is 3.8. As mentioned previously in this section, one of the attractions of python is its libraries and packages. We have used a large number of these packages at every step right from the beginning.

‘Tweepy’ package has been used for the collection of tweets. The dataset that consists of tweets made post vaccination were collected using the tweepy package. The package allows us to access the tweets that are in a week’s time frame. During the pre-processing a lot of things were to be kept in mind. The package employed during the pre-processing stage was the NLTK package. The NLTK stands for natural language toolkit. The package has enabled us to perform the various steps included in the pre-processing stage. Using the package, we have done activities like removing the non-English tweets, punctuations, stop words etc. for implementing the deep learning models two primary libraries have been used one being keras and the other being Transformers library. The Bi-LSTM model was implemented using the keras library. The keras is a framework that is built on top of the tensorflow library. The keras allow easy addition of dense layers to the model network. Both BERT as well as XLNET are a part of the transformers library and can be instantiated using the same.

In this section, we have explained the various libraries and packages of python that has been used and why python has been the selected language for the project implementation.

5.3 PYTHON LIBRARIES USED DURING IMPLEMENTATION

As mentioned previously, the language that has been used for implementation of the project is python. The advantages of python are many in respect with the field of data science as well as deep learning. However, as stated in [19] one of the major advantages of using the python language has been the access to various packages and libraries. This helps in reducing the labor to write down codes for every single tasks. Some of the major libraries that have been used during the implementation are described here in detail.

TWEEPY

Tweepy is a python library that provides us access to the twitter API. Since, our data required the twitter dataset, we are very well informed about the lack of public datasets. The twitter

API enables the collection of tweets. The filtering out of tweets can also be done using the tweepy library. If we want tweets in any language, we can filter them out and our dataset will contain tweets in that particular language.

For accessing the twitter API, we need to have a developer account. A twitter developers account provides us with a secret key and the access tokens. These enable the tweepy library to authorise and authenticate our developers account followed by providing all the data we need. The data can be saved as a .csv or .json file for easy manipulation during the implementation of any project.

MATPLOTLIB

The best way for a human to understand things in a pictorial representation. We have various animated and graphical representations that help us understand things better. To understand our data as well as to understand the results of our data, we need to represent them in visual forms. Matplotlib is another python library that has been extensively used while implementing the project presented in this document. The graphs that help in comparative analysis of the deep learning models have been plotted using the library. Also, the word clouds are represented using the matplotlib python library.

In the final stage of the project, we needed to compare the results of lockdown vs post-vaccination twitter data. Now for helping the people understand how actually the mental health during the lockdown has been affected, we used the matplotlib library for exploratory data analysis to understand both the datasets.

TRANSFORMER (HUGGING FACE) LIBRARY

Transformer models have changed the approach towards sequence-to-sequence prediction. Also, their performance on text data has been a breakthrough. In the implementation of this project, we have used two transformer models- BERT and XLNET. Both the models have outperformed the keras model. Transformers library has been used to implement these models. The models have been further fine-tuned so as to give satisfactory results for our problem statement.

KERAS

Keras is not a package but it is a framework for one of the strongest python library. Keras acts like an interface to python library- TensorFlow. In [20] article, it is stated that keras is built on top of the tensorflow. If we build our machine learning model using keras, it utilizes the

backend of the tensorflow. Bi-LSTM model has been implemented in this project and it has been done using the keras framework. Keras enables and eases adding the various model layers and also has a robust nature in training the model.

5.4 TESTING

5.4.1 UNIT TESTING

According to [21], Unit testing is a testing strategy in which all the parts are tested in small units. e.g. in case of a machine, in unit testing every part of the machine is individually tested so as to see if it is performing its part fine. We have implemented a twitter sentiment analysis using deep learning in this project. Since the project is a complicated one, it has various modules that are implemented. e.g. data-collection, data-pre-processing, data labelling, model implementation and final analysis. All these modules are co-dependent. Any mistake in one step can bring about an error in the rest of the modules. Therefore, we had to perform unit testing. During unit testing, every module code was run several times to ensure if it is working correctly and the results that are produced are the same. If any error was found in one module, changes in rest of the modules were made accordingly.

The project unit testing was done to ensure that the data is coherent and the models have the same data to train as well as validate. Unit testing ensured that all the deep learning models work on a properly labelled data.

CHAPTER 6

RESULTS AND ANALYSIS

The aim of this project has been to achieve a perspective of the mental health of the people during pandemic. For this purpose, we have used the twitter data and taken into account three deep learning models- Bi-LSTM, BERT and XLNet. In the previous sections these models have been thoroughly described with their architecture as well as working explained in detail. In this section, the various metrics of these models will first be stated and described and then a comparative analysis of the three models will be carried out.

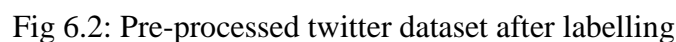
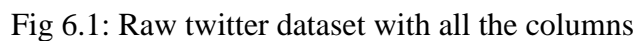
6.1 DATASET

Twitter data is one that is not available online or publicly. Publishing the twitter data present with you is considered a crime and does not come under the terms and conditions of twitter. To access any twitter data, the person must have a developers account and have the access tokens.

Tweepy library in python allows collection of twitter data. However, the only data that we can access is the one that is made in the past seven data. As far as our project was concerned, we needed data from March, 2020 through December, 2020. Tweepy library could therefore not be used for this dataset.

Tweetset is a website that consists of tweet-ids and can be collected for any time frame. From the website we collected the tweet-ids using the coronavirus filter and then these tweets were hydrated. A dataset of 105000 rows was collected.

The twitter data of the March 2021 was collected using the tweepy library. We just collected tweets of the few hours of 15 March 2021 for the final comparisons. This dataset was used to implement the final model and predict the labels. Further comparison of the two datasets were followed right after this.



The above figure is word cloud visualization of tweets made after the people were vaccinated. We can clearly see that words like panic, anxiety do not appear large in size like the previous word cloud. We can make a faint assumption that the tweets of these word clouds do not represent depressed tweets.

Now, for us to determine if the tweets represent depression or not, we have employed deep learning models to determine how the mental health of people has been during the pandemic.

6.2 METRICS FOR MODEL EVALUATION

6.2.1 PRECISION AND RECALL:

Precision is described as a metric for measuring the exactness of a class. It is mathematically described as true positive divided by sum of true positive and false positive. If the precision is really low, it indicates that the false positives are large in number.

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

Where, TP= True positives

FP= False positives

Recall is another metric for evaluating the performance of our model. It can be described as a metric that indicates the completeness of a classifier. Mathematically, recall is defined as true positives divided by a sum of true positives and false negatives. In a model with low recall value, the false negatives are large in number

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

Where, TP= True positives

FN= False negatives

6.2.2 ACCURACY

Accuracy is one of the most sought-after evaluation metrics for classification model. In common words, accuracy is a measure of the correctness of the model. Mathematically accuracy is defined as sum of true positives and true negatives divided by the sum of true positives, true negatives, false positives and false negatives.

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

Where, TP = true positives

TN = true negatives

FP = false positives

FN = false negatives

Although accuracy is one of the most commonly used metrics for evaluating deep learning classification models but other metrics should be well taken into consideration.

6.2.3 F1SCORE

F1 score is a classification metric that is derived from precision and recall. Mathematically, F1 score can be calculated as follows:

$$\text{F1 score} = 2 * (\text{precision} * \text{recall}) / (\text{precision} + \text{recall})$$

F1 score can thus be defined as a metric in between precision and recall. However, it does differ from accuracy. Accuracy takes into account the true negatives while F1 score takes into account false positive as well as false negative. It can also be argued that F1 score is a better metric than accuracy as it takes into account the business cost affecting values – false positive and false negative as opposed to accuracy that takes into account true negative that does not have much business cost.

6.3 PERFORMANCE OF DEEP-LEARNING MODELS

In the project, the dataset has been prepared from the scratch as the data is new and had to be collected carefully. The dataset was prepared best to our knowledge. Any data that is irrelevant

or could contribute as noise during the training was well removed. Three different models were trained and tested – Bi-LSTM, BERT and XLNet. The results of these models will be mentioned in this section before finalizing the model that performed the best.

6.3.1 Bi-LSTM

Bi-LSTM model is built using the keras library in python. A bidirectional wrapper in keras allows the normal LSTM to be operated in forward as well as backward direction. A Bi-LSTM has an embedding layer which passes on to the LSTM layer. The embedding layer required an embedded form of our input, in our case tweets. The tweets are embedded using GloVe embedding technique.

GloVe embedding is a technique of word embeddings. In word embedding technique, the words are represented as dense vectors.

In LSTM, embedding layer is the first layer of our deep neural network. Following the embedding layer, the LSTM layers are added using keras. Adding more bi-directional LSTM layers makes our deep neural network more complicated. The number of bi-LSTM layers to be added in our deep neural network depends upon how complicated our problem statement is. Our project aims to classify tweets into depressed, non-depressed and neutral classes. This means that our project is a classification problem. This means that after adding the LSTM layers, we need to add a classifier.

Word embeddings are a technique in which the words are represented in the form of dense vector. For any deep neural networks as well as machine learning models, text data cannot be an input. These models accept some form of integer vectors as input data. Word embedding is an approach to convert text data into vector forms. There are many word embedding techniques that have been employed for tweet classification. In building the Bi-LSTM model we employed a powerful technique of word embeddings known as **GloVe** embedding. According to [22] GloVe stands for “global vectors”. GloVe embeddings are built on the principle of matrix factorization technique in which the matrix is created by the word-content. The embedding dimensions can vary depending on the problem statement. GloVe embeddings are used to create a vector representation of the text. The GloVe embeddings use both the global as well as local statistic of a corpus to create a word embedding.

Loss functions are present in all the deep learning models. Loss functions are a way for us to understand if the chosen algorithm fits or models our data appropriately. While trying to build

a perfect model, the loss function gives us an insight if we are going the right way. If the model predicts the output as desired, i.e. the model fits the dataset well, the loss function inputs a lower value. However, if the algorithm does not model our dataset well, the loss function inputs a higher value. While building a model for any problem statement, we should aim at a lower loss function value. As mentioned earlier, our project is a multiclass classification problem. For this purpose, we have used “**Sparse Categorical Cross Entropy**” loss function. The sparse categorical cross entropy loss function works similar to the categorical cross entropy loss function. The only difference is that the output labels in categorical cross entropy loss function are encoded while the output labels in sparse categorical cross entropy loss function are not encoded. The mathematical formula for the loss function are as follows:

$$\text{Loss} = \sum_{i=1}^n y_i * \log \hat{y}_i$$

Where Loss = calculated loss

n = number of scalar values in output

\hat{y}_i = i^{th} scalar value in output of model

y_i = corresponding target value

While training our Bi-LSTM algorithm, the model can overfit if the training examples are passed repeatedly over the model. Overfitting is a situation, in which the model learns the values rather than learning from the values. To overcome the problem of overfitting in Bi-LSTM we have used a dropout of 0.2. Dropout is a regularization technique. In this, some of the outputs of the model are ignored or dropped while training. This helps regularize the dataset and prevent overfitting.

Optimizers are used in deep learning models to adjust the value of characteristics such as learning rate and weight decay to reduce the value of loss and simultaneously increase the accuracy of the model. Adam optimizer is one of the best optimizers. It can handle the noises and sparse gradients really well. It is a stochastic gradient descent optimizer. Adam optimizer computes the individual adaptive learning rates of the various parameters. The Twitter sentiment analysis presented in this work using a bi-directional LSTM used GloVe embeddings to create a vector representation of the tweet. Also, for building a Bi-LSTM model, Adam optimizer was used.

Table 1: Bi- LSTM results

Performance Metrics	Values
Precision	0.819
Recall	0.823
F1 Score	0.818
Accuracy	82.332

6.3.2 BERT MODEL

BERT model has been also employed for the tweet classification. The pre-trained model 'bert-base-uncased' has been used for sentiment analysis. The model is fine-tuned for better performance. Since, the text data has to be converted into tokens for the encoder input, the project has used BERT embeddings. The BertTokenizerFast is the Bert tokenizer that has been employed for tokenizing the input tweet texts. To classify the tweets as depressed, non-depressed and neutral, we have added a classification layer on top of the BERT model. AdamW optimizer has been used for the model. The learning rate of the optimizer has been set to improve the performance of the model. Also, linear scheduler with warmup has been used. The weight decay of the model also has been set to optimize the performance of the model. The AdamW optimizer separates the weight decay and the learning rate. This implies that both the learning rate and the weight decay can be optimized separately.

Table 2 | BERT results

Performance Metrics	Values
Precision	0.966
Recall	0.963
F1 Score	0.965
Accuracy	96.716

6.2.3 XLNET

XLnet is used for the purpose of text-classification and not for word prediction. For the same, the huggingface transformer library has pretrained model. We have used BertForSequenceClassification to predict the class of our tweets. For generating input ids and attention masks, XLNetFastTokenizer has been used to encode all the tweets. Also, like BERT, AdamW optimizer with a learning rate of $5e-5$ has been used for training the model. The learning rate enables the model to learn the information which helps improve the prediction. The weight decay has been set at 0.01 and also helps in the regularization of the input for training.

Following the implementation of the models, we continued to do a comparative analysis of all the three models so as to determine which model classified the sentiments of the tweets as depressed, non-depressed and neutral most effectively.

Table 3 | XLNet Results

Performance Metrics	Values
Precision	0.956
Recall	0.955
F1 Score	0.956
Accuracy	95.802

The first model implemented during the course of the project has been the Bi – LSTM model. For this particular model, we have implemented the metric of ROC curves as well. ROC stands for receiver operating characteristic. The ROC curve determines how well the classifier diagnosis. ROC curves are very common for binary classifier. However, as we know that the tweets are classified into three classes- neutral (0), depressed (1) and non-depressed (2), therefore the ROC curve to evaluate the same will not be a single plot. In fact the plot for every class will be different. The plots for the ROC in case of Bi-LSTM are as follows:

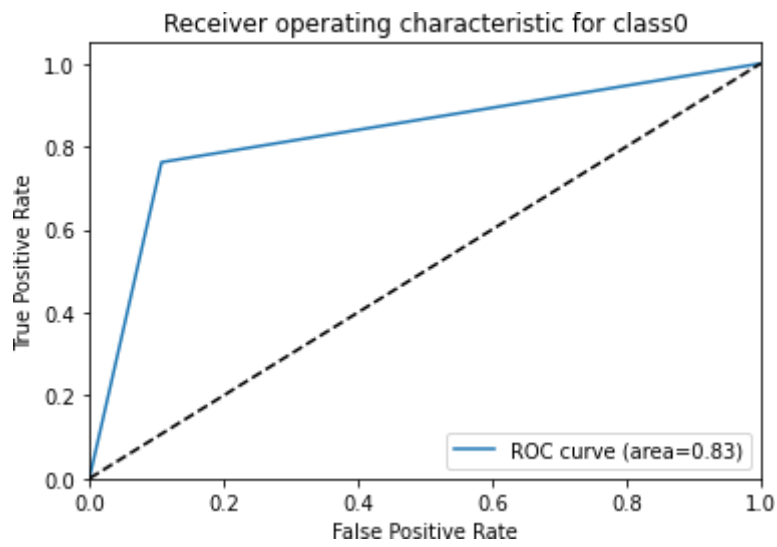


Fig 6.5: ROC curve for neutral tweets in Bi-LSTM

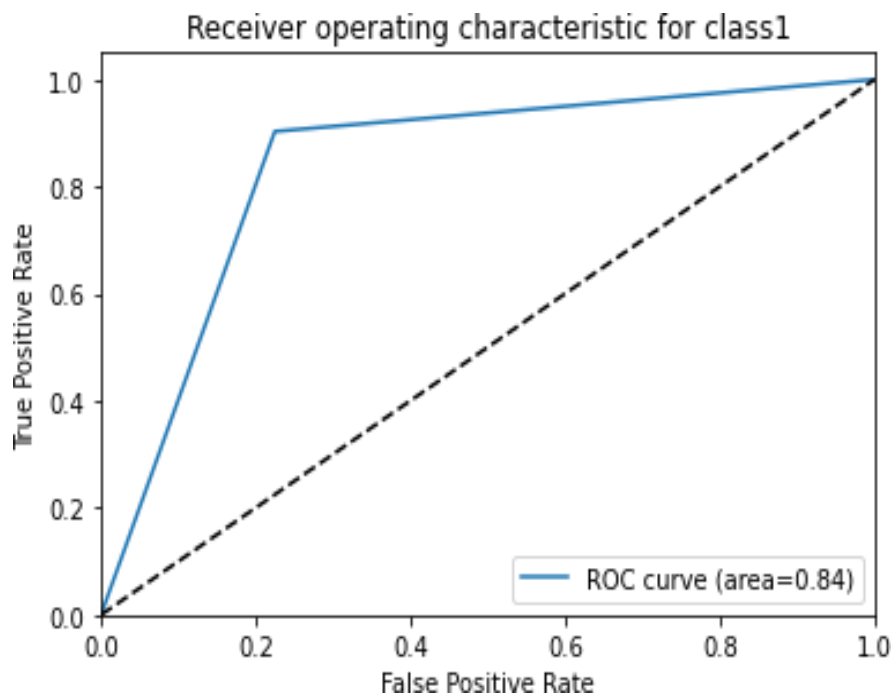


Fig 6.6: ROC curve for depressed tweets in Bi-LSTM

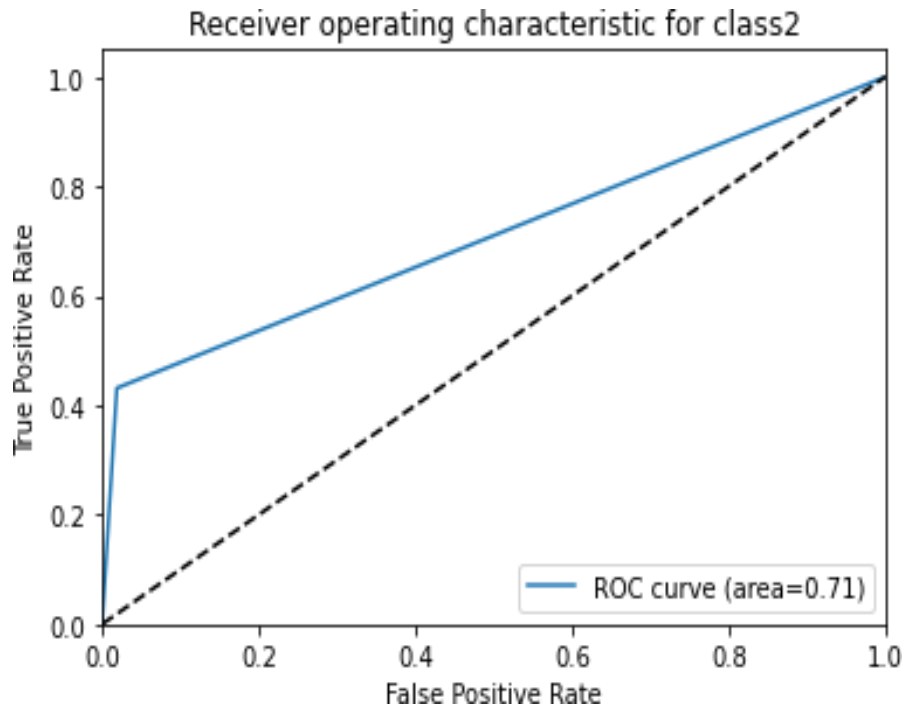


Fig 6.7: ROC curve for non-depressed tweets in Bi-LSTM

The graphs representing the accuracy and loss of Bi-LSTM model versus the epochs given below:

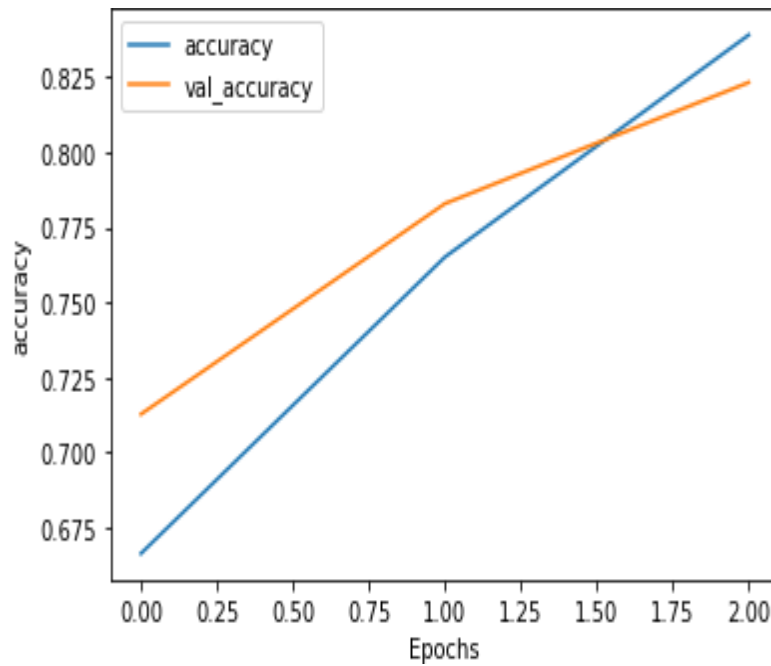


Fig 6.8: Accuracy of Bi-LSTM through the epochs

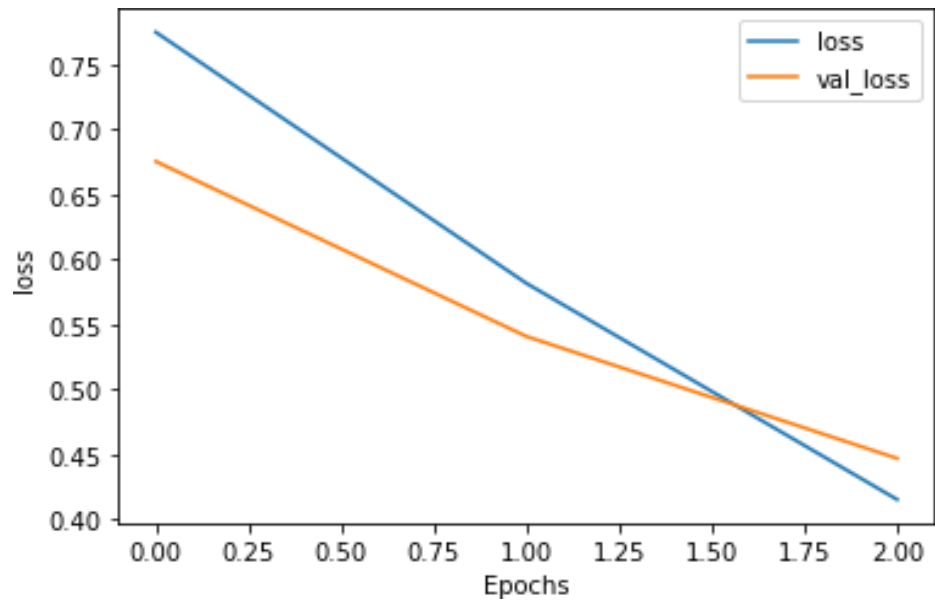


Fig 6.9: Training and validation loss of Bi-LSTM through the epochs

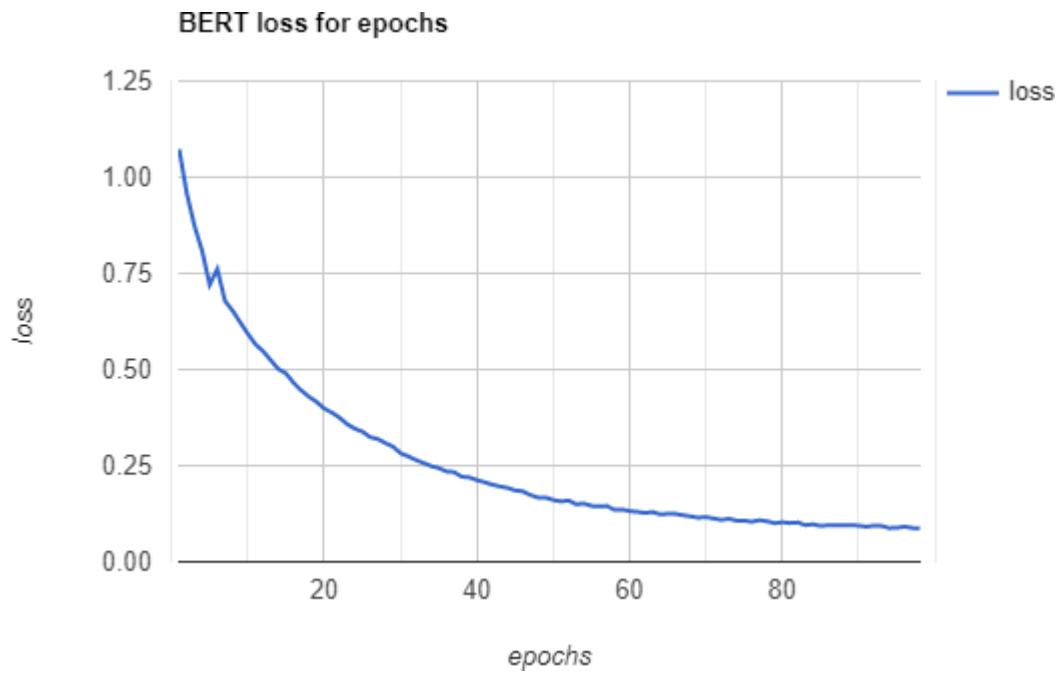


Fig 6.10: Loss of BERT through the epochs

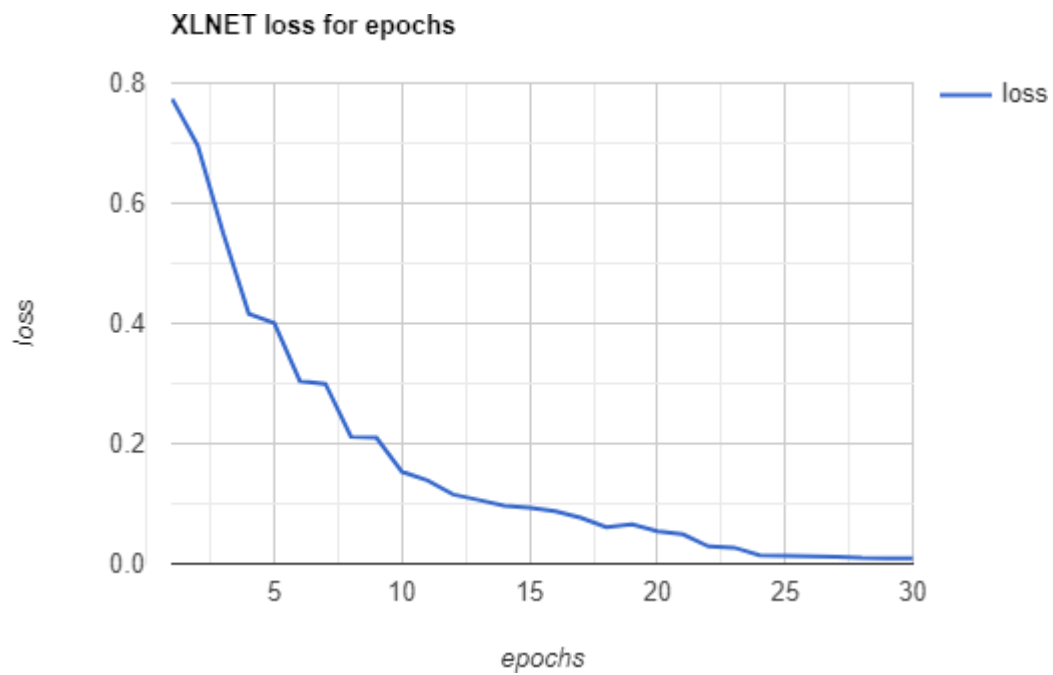


Fig 6.11: Loss of XLNET through the epochs

The graphs comparing the performance metrics of the three models have been given below

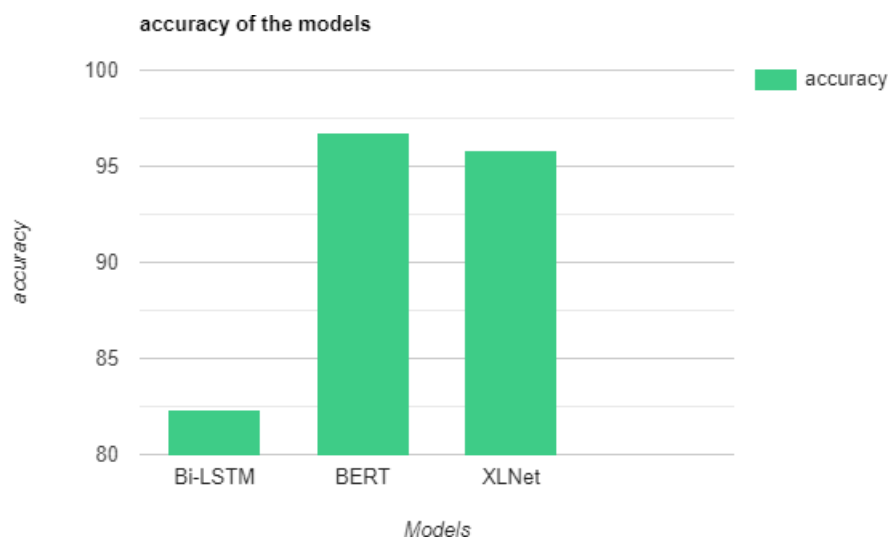


Fig 6.12: Accuracy of the models implemented

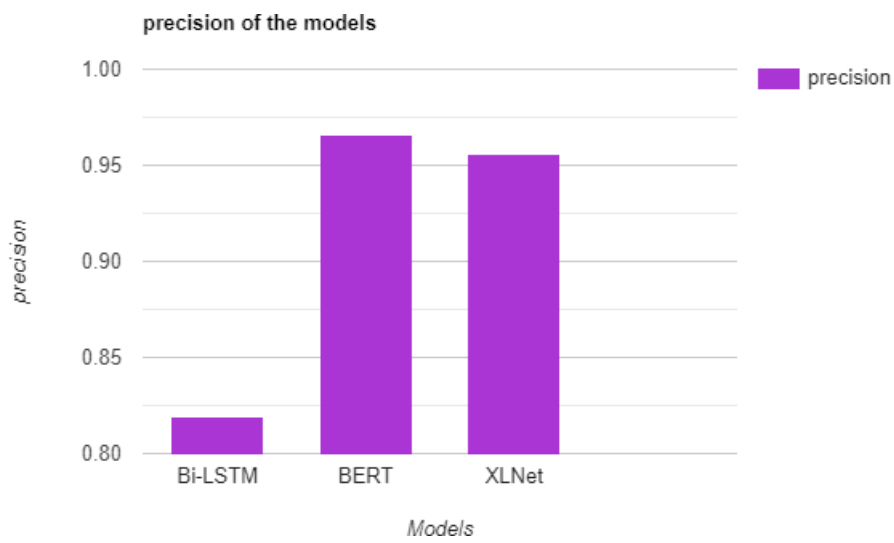


Fig 6.13: Precision of the models implemented

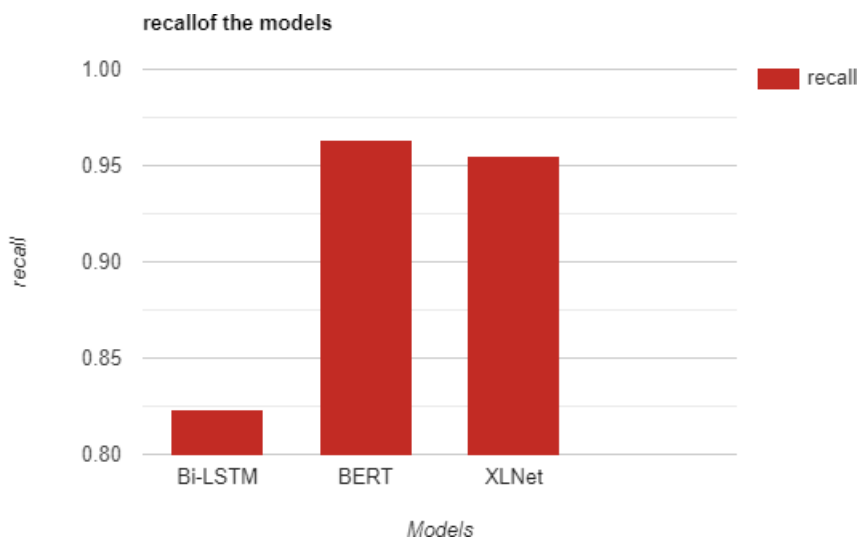


Fig 6.14: Recall of the models implemented

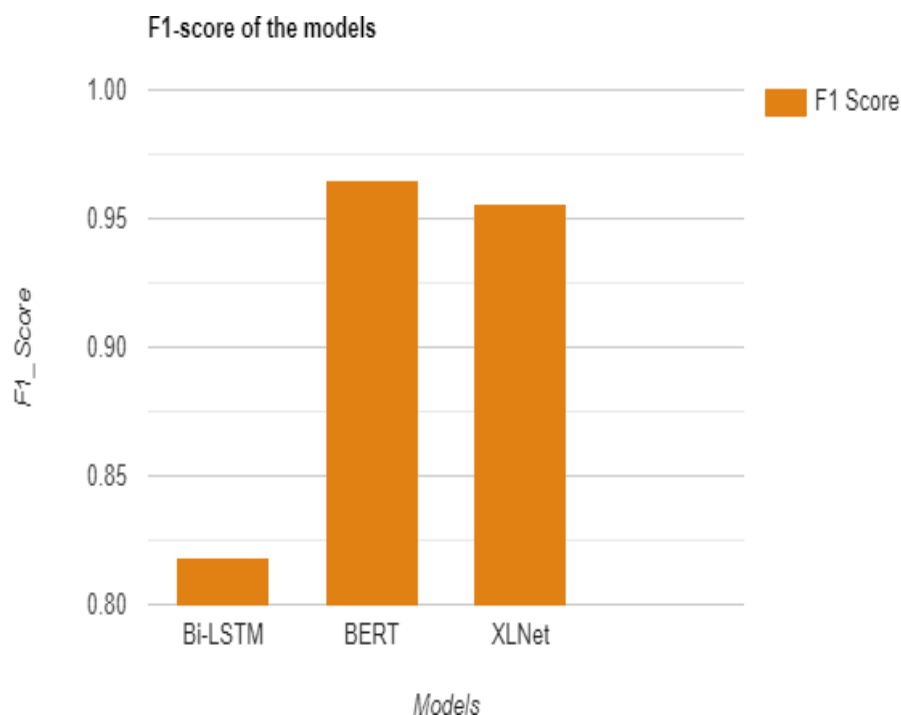


Fig 6.14: F1 Score of the models implemented

6.4 EXPLORATORY DATA ANALYSIS

The tweet dataset that has been collected was large and very difficult to interpret. Through our models, we are trying to detect the depression and alterations in the mental health of the people during the pandemic. However, analyzing such a large dataset is quite challenging. It will take time as well as a lot of human labor. This is in no way accessing the technologies available.

One of the best ways for humans to understand anything is through pictorial representations. In every field, presenting your work through graphs and pictures is considered a crucial step because this makes it easy for the rest to understand.

Exploratory data analysis (EDA) is a technique to analyze our data and get an understanding from it. This makes it quite easy to pick out the features and actually understand what is happening. Like, for this section of generating the most commonly used words, it becomes quite simple to draw conclusions from it. Besides plotting of graphs for ROC, accuracy as well as loss is included.

6.5 POST- VACCINATION AND POST-LOCKDOWN ANALYSIS

The deep learning models that are mentioned in the previous section were trained using the dataset that contained tweets during the time when covid-19 pandemic was on peak, that is, March 2020 through December 2020. The results of the model will be described in detail in the sections that follow. However, here we would point out that the BERT transformer model outperformed the other models. The trained model was then used to predict the labels of the dataset for the post-vaccination and post-lockdown tweets. This section provides a comparative analysis of how the mental-health of people has been following the vaccinations and when lockdown restrictions were eased.

In the last and final part of our implementation, we used the data collected post the vaccinations and when the lockdown had eased. After the lockdown restrictions were eased, people started getting back to their normal life. The vaccinations that have been introduced allowed people to give out a sigh of relief. The people were able to visit their therapist and the decreasing number of deaths surely has boosted the confidence of some.

Using the model that performed best, we predicted the class of a small post-vaccination dataset into depressed, non-depressed and neutral. Following this, we carried out the comparative analysis of the two datasets to get an insight.

Previously, the word cloud visualization has shown how words like happy, glad, hope have been used more in the data post vaccination. This gives us a faint perspective of improved mental health. But to actually draw an analysis, we need to know the distribution of the data into the various classes.

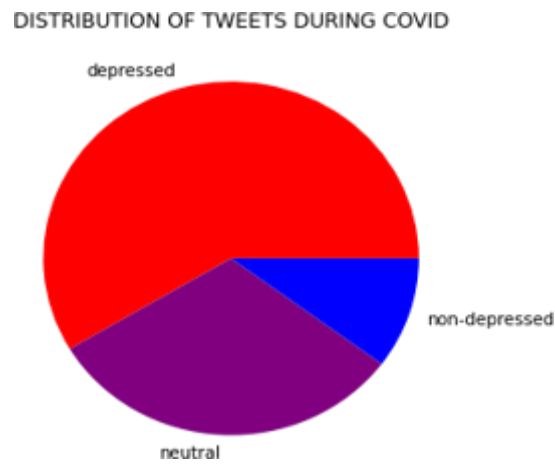


Fig 6.15: Distribution of tweets during lockdown

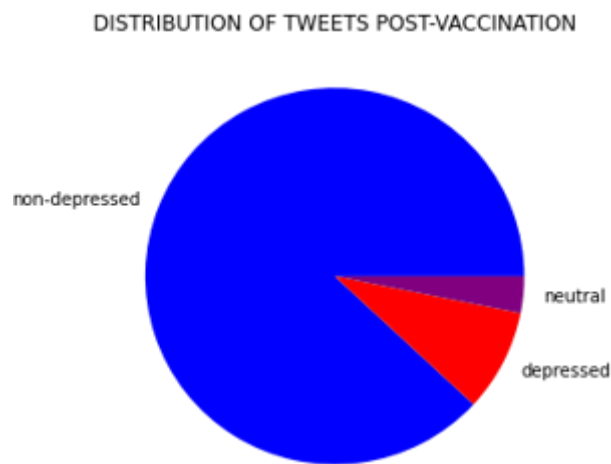


Fig 6.16: Distribution of tweets post-vaccination

From figures 6.14 and 6.15, we can clearly see that there has been more than 40% drop in the depressed tweets. In fact, the tweets that have been made post the vaccination are dominated into non-depressed category. Following this, a final comparative analysis of the words that have been used is carried out.



Fig 6.17: Word distribution in tweets during lockdown

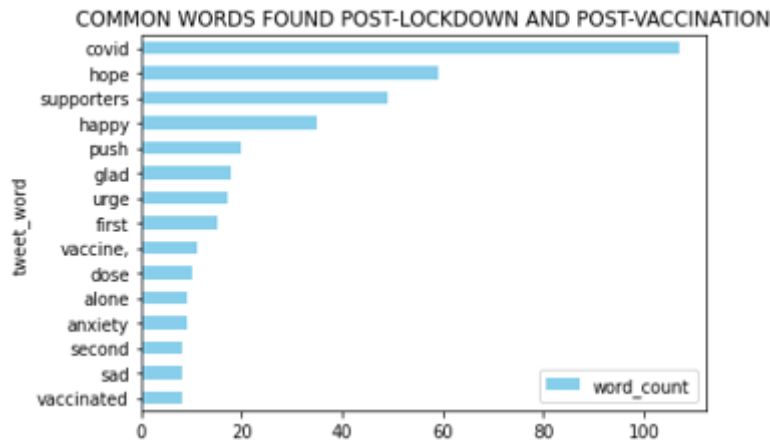


Fig 6.18: Word distribution in tweets post-vaccination

Figure 6.16 and 6.17 have shown the common words during the lockdown and post the vaccinations. When the pandemic was at peak, words like pandemic, hope, covid, anxiety, sad, mental health were used a lot. These words clearly pointed at depression and depreciated mental-health. However, post vaccination worlds like hope, supporters, happy, push, glad etc. were common. These words do not contribute into the set of depression pointing words.

In this section, we have aimed to showcase that.

From the results that have been mentioned above it is clear that Bi-LSTM has a lower performance as compared to BERT and XLNet taking into account all the evaluation metrics.

In case of BERT and XLNet the precision, recall, F1 score as well as the accuracy of the models is quite close. However, BERT has outperformed XLNet in case of our Twitter sentiment classification problem.

In the last section of the project, we used the BERT model to predict the class of the post-vaccination dataset and draw a comparative analysis of the mental-health during lockdown vs post-vaccination / post- lockdown as mentioned in the previous section.

CHAPTER 7

CONCLUSION AND FUTURE WORK

7.1 CONCLUSION

The project that has been summarized in this document has been a successful step towards contributing to a secured and better mental-health. The project has successfully used the deep learning algorithms to perform twitter sentiment analysis.

Depression and anxiety mark the onset of deteriorating mental health in a society. But as the tradition continues the mental health of the people has been overlooked. The people suffering from depression and anxiety are overlooked and not taken into account. This has led to suicides, increasing cases of bipolar as well as underperformance of people in jobs or schools.

Twitter is a popular microblogging platform where people can talk about whatever they want. Taking into account all the social media platforms, twitter has contributed most to the ongoing issues worldwide. One thing that makes the issue come out in public are the hashtags. There are hashtags that are associated with any ongoing issue that people want to bring to light. Twitter has started movements and also been a platform to protest against the wrong.

During the lockdown that was imposed in covid-19 pandemic, the people were made to sit at home for months. Even stepping out for daily needs had to be restricted. People got introduced to a lifestyle that was new to them. This affected the mental state of a lot of people. People lost their free will. Apart from this, unemployment rates started taking a hike as business of the associations was affected. And the death rates did leave a very deep effect on the brain of the people. To combat depression, people took help of counsellors or therapists. But the people were not able to do it. A lot of people actually put themselves out on twitter expressing how they have been battling a fight with anxiety and depression. Many people having suicidal thoughts also turned to social media looking for a glimmer of hope.

During the pandemic, very less was done to help people who developed mental-health issues. We in this project have taken a step toward improving the mental health of people and give an awakening call to everybody to improve mental health. The deep learning models have been used

to detect the depression among the users. The models that have been implemented are Bi-LSTM, BERT and XLNET. BERT outperformed the other two models which are Bi-LSTM and XLNET with an accuracy of 96.7% approximately. This model was then used to get an insight of how the vaccinations and relaxations in lockdown have brought an improvement in the mental health of people.

7.2 FUTURE ENHANCEMENTS

The project got a lot of scope for future enhancement. We in this project have focused on the tweets that were in the English language. We know the pandemic hit Europe the most. The strain of the virus there was dangerous and wiped away many lives. One of the worst hit countries during the pandemic was Italy. The economy and the livelihood of people is lost. But we know that people in Italy prefer to speak in Italian. English is not a very well-known language there. So, if we used the models to train in languages like Italian, French, Spanish as well as German we can also decide about the mental health in worst hit countries like Italy, France, Spain etc.

The post vaccination data is not abundant. The vaccination has started very recently and many countries have reimposed restrictions. In future when the vaccination drive would be carried out in mass and people are vaccinated, we can understand better about how the vaccination has impacted the mental health of the people.

REFERENCES

- [1] Covid map: Coronavirus cases, deaths, vaccinations by country - BBC News
- [2] Nimrita Panchal, Kabah Kamal, Cynthia Cox, Rachel Garfield (2020, Feb 10). KFF Filling the need for trusted information on national health issues[Online]. Available: <https://www.kff.org/coronavirus-covid-19/issue-brief/the-implications-of-covid-19-for-mental-health-and-substance-use/>
- [3] Paul Harrison (2020, November 10). Department of Psychiatry, University of Oxford[Online]. Available: <https://www.psych.ox.ac.uk/news/20-of-covid-19-patients-receive-psychiatric-diagnosis-within-90-days>
- [4] Bollen, J., Mao, H., & Pepe, A. (2011). Modeling Public Mood and Emotion: Twitter Sentiment and Socio-Economic Phenomena. Proceedings of the International AAAI Conference on Web and Social Media, 5(1).
Retrieved from <https://ojs.aaai.org/index.php/ICWSM/article/view/14171>
- [5] Jamil, Zunaira (2017) Monitoring Tweets for Depression to Detect At-Risk Users
- [6] Asish Vaswani, Noam Shazeer, Niki Parmar (2017). Attention Is All You need
- [7] A. Rane and A. Kumar, "Sentiment Classification System of Twitter Data for US Airline Service Analysis," 2018 IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC), Tokyo, Japan, 2018, pp. 769-773, doi: 10.1109/COMPSAC.2018.00114.
- [8] Maryam Hasan, Elke Rundensteiner, Emmanuel Agu (2014). EMOTEX: Detecting Emotions in Twitter Messages
- [9] Diveesh Singh & Aileen Wang (2020). Detecting Depression Through Tweets
- [10] Prateek Joshi (2018, July 30). Analytics Vidya[Online]. Available: <https://www.analyticsvidhya.com/blog/2018/07/hands-on-sentiment-analysis-dataset-python>
- [11] A. Krouska, C. Troussas and M. Virvou, "The effect of preprocessing techniques on Twitter sentiment analysis," 2016 7th International Conference on Information, Intelligence, Systems & Applications (IISA), 2016, pp. 1-5, doi: 10.1109/IISA.2016.7785373.
- [12] Bruno Stecanella (2019, May 11). Monkey Learn[Online]. Available: <https://monkeylearn.com/blog/what-is-tf-idf/>
- [13] Rohith Gandhi (2018, June 7). Towards Data Science[Online]. Available: <https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning->

algorithms-934a444fca47

- [14] Raghav Agarwal (2019, July 4). Medium(Online). Available:
<https://medium.com/@raghavaggarwal0089/bi-lstm-bc3d68da8bd0>
- [15] Md. Yasin Kabir, Sanjay Madria (2019). A Deep Learning Approach for Tweet classification and Rescue Scheduling for Effective Disaster management
- [16] Rani Horev (2018, Nov 10). Towards Data Science[Online]. Available:
<https://towardsdatascience.com/bert-explained-state-of-the-art-language-model-for-nlp-f8b21a9b6270>
- [17] Zhilin Yang, Zihang Dai, Yiming Yang, Jamie Carbonell, Ruslan Salakhutdinov, Quoc V. Le (2019). XLNet: Generalized Autoregressive Pretraining for Language Understanding
- [18] Aishwarya V Srinivasan (2019 , August 1). Towards Data Science[Online]. Available:
<https://towardsdatascience.com/xlnet-explained-in-simple-terms-255b9fb2c97c>
- [19] TechVidvan (September 20) [Online] Available:
<https://techvidvan.com/tutorials/python-advantages-and-disadvantages/>
- [20] Manpreet Singh Minhas (February 15). Towards Data Science [Online]. Available:
<https://www.infoworld.com/article/3336192/what-is-keras-the-deep-neural-network-api-explained.html>
- [21] Manpreet Singh Minhas (February 15). Towards Data Science [Online]. Available:
<https://towardsdatascience.com/unit-testing-in-deep-learning-b91d366e4862>
- [22] Wikipedia (2020, November 19) [Online]. Available:
Retrieved From: [https://en.wikipedia.org/wiki/GloVe_\(machine_learning\)](https://en.wikipedia.org/wiki/GloVe_(machine_learning)).

APPENDIX

DATA COLLECTION

Post lockdown and post-vaccination tweet mining:

```
import tweepy
import csv
from google.colab import files
files.upload()
API_KEY = "9LyDtW88uhyzYLcZBo3sLT3of"
API_SECRET = "YmZWfEw74MWaVCG0wmUczCleF7TUIId7mIsHydWRnwJ4vVqSQIO"
access_token = "1309932783414865920-g6BSNBOH6F5LBtQDeQDXv0ILBGdstq"
access_token_secret = "R3HXjkPHgsXuJEomORCgl7TuKKwCXFqLUlpaUnGYHT4aW"
authorization = tweepy.OAuthHandler(API_KEY,API_SECRET)
authorization.set_access_token(access_token,access_token_secret)
API = tweepy.API(authorization,wait_on_rate_limit=True)
from google.colab import drive
drive.mount('/content/drive')
post_lockdown_tweets = []
# API.user_timeline(screen_name=screen_name,count=200)
myFile = open('post_vaccination_tweets.csv','a')
csvWriter = csv.writer(myFile)
for tweet in tweepy.Cursor(API.search,q = "covid AND vaccine -filter:retweets", since = "2021-03-19",until = "2021-03-20",lang = "en",).items():
    csvWriter.writerow([tweet.created_at,tweet.id,tweet.text.encode('utf-8')])
    print(tweet.created_at,tweet.id,tweet.text)
myFile.close()
```

Tweet Cleaning:

```
#importing the necessary libraries
```

```
import pandas as pd
```

```
from nltk.stem.porter import *
```

```
from nltk.tokenize import TweetTokenizer
```

```
from nltk.stem import WordNetLemmatizer
```

```
from nltk.corpus import stopwords
```

```
from nltk.stem.porter import *
```

```
import nltk
```

```
import string
```

```
import re
```

```
pd.set_option('display.max_colwidth',100)
```

```
# Loading the dataset
```

```
def load_data_csv():
```

```
    #data_=pd.read_csv('predicted_post_covid.csv',names=['created_at','id','text'])
```

```
    #data_ = pd.read_csv('month.csv')
```

```
    data_ = pd.read_csv('tweet_data.csv')
```

```
    return data_
```

```
twitter_df=load_data_csv()
```

```

our_df=pd.DataFrame(twitter_df)

print(our_df.head())

# only include english tweets

our_df =our_df[our_df['lang']=='en']

#drop all retweets

our_df=our_df[~our_df['text'].str.startswith('RT')]

our_df['text']=our_df['text'].astype(str).str.replace('\d+',")

our_df=our_df[~our_df['text'].str.startswith('@')]

our_df['text']=our_df['text'].astype(str).str.replace('\d+',")

#remove the stop words... add few more

extended=['yr','year','woman','man','girl','boy','xe','x','xa','xm','b','still','let','know','case','new"due','t
ime','lot','say','make','go','take','im','due','via','find','way','may','full','X',]

stpwrds=set().union(stopwords.words('english'),extended)

our_df['text']=our_df['text'].apply(lambda x: ' '.join(word for word in x.split() if word not in
(stpwrds)))

#removing the hyperlinks

def remove_hyperlinks_retweets(txt):

    txt=' '.join([wrd for wrd in txt.split(' ') if 'http' not in wrd])

    return txt

our_df['text']=our_df['text'].apply(lambda x:remove_hyperlinks_retweets(x))

```



```

def remove_twitter_handles(txt):

    txt=' '.join([wrd for wrd in txt.split(' ') if '@' not in wrd])

    return txt

our_df['text']=our_df['text'].apply(lambda x:remove_twitter_handles(x))

#removing the punctuations

our_df['text']=our_df['text'].str.replace('[^\w\s]','')

our_df['text']=our_df['text'].str.replace("[^a-zA-Z#]",' ')

#Drop COLUMNS Keep Only Tweets

our_df=our_df[['id','text']]

stem_tweet=PorterStemmer()

our_df['text']=our_df['text'].apply(lambda x:" ".join(stem_tweet.stem(txt) for txt in x.split()))

our_df['text']=our_df['text'].astype(str).str.replace('\d+','')

#Lemmatization

ltizer=WordNetLemmatizer()

our_df['text']=our_df['text'].apply(lambda x:" ".join(ltizer.lemmatize(txt)for txt in x.split()))

our_df['text']=our_df['text'].astype(str)

our_df['text']=our_df['text'].apply(lambda y: " ".join([word for word in y.split() if word not in stopwords.words('english')]))

#Break into tokens

twk=TweetTokenizer(strip_handles=True,preserve_case=False,reduce_len=True)

```

```

data_f['text']=data_f['text'].apply(twk.tokenize)

data_f['text']=data_f['text'].astype(str).str.replace("\d+',"")

#save to a new file

our_df.to_csv('tweet_data_cleaned.csv')

```

Word Cloud Visualization:

```

#load Dataset

import pandas as pd

from wordcloud import WordCloud

import matplotlib.pyplot as plt

def load_data():

    data_=pd.read_csv('post_vaccination_tweets_final.csv')

    return data_

data_=load_data()

df_twitter= pd.DataFrame(data_)

def wordcloud_plot(wordcloud):

    plt.figure(figsize=(20,20))

    plt.imshow(wordcloud)

    plt.axis('off')

my_string=[]

for text in df_twitter['text']:

```

```
my_string.append(text)

my_string=pd.Series(my_string).str.cat(sep=' ')

wordcloud=WordCloud(width=1000,height=500).generate(my_string)

wordcloud_plot(wordcloud)
```

LabellingDataset:

```
import pandas as pd

from sklearn.feature_extraction.text import TfidfVectorizer

from sklearn import svm

from sklearn.cluster import KMeans

from sklearn.pipeline import Pipeline

from sklearn.svm import SVC

from sklearn.ensemble import RandomForestClassifier

from sklearn.preprocessing import LabelEncoder

from nltk.corpus import stopwords

def load_csv_file():

    #file_csv=pd.read_csv('dataset2.csv')

    file_csv=pd.read_csv('FINAL_DAY.csv')

    return file_csv

data_file=load_csv_file()

data_to_train=pd.DataFrame(data_file)
```

```

#data_file_test= pd.read_csv('month_clean_1.csv')

data_file_test= pd.read_csv('tweet_data_cleaned.csv')

data_to_label=pd.DataFrame(data_file_test)

#label encoder

encoder_label=LabelEncoder()

Y=encoder_label.fit_transform(data_to_train['label'].astype(str))

#VECTORIZATION TF-IDF

tfidfcon=TfidfVectorizer(ngram_range=(1,3))

X=data_to_train['text'].values

X = tfidfcon.fit_transform(data_to_train['text']).toarray()

data_to_train['vectorized']=list(X)

data_to_train.to_csv("FINAL_DAY_.csv")

X1=data_to_label['text'].values

X1=tfidfcon.fit_transform(data_to_label['text']).toarray()

data_to_label['vectorized']=list(X1)

#SVC classifier

classifier=SVC(kernel='linear')

for i in range(100):

    classifier.fit(X,Y)

prediction=classifier.predict(X1)

```

```
data_to_label['label']=prediction
```

```
data_to_label.to_csv('FINAL_.csv')
```

Bidirectional LSTM:

```
import keras
```

```
from sklearn.preprocessing import LabelEncoder
```

```
from keras.layers import Dropout, Dense, Embedding, LSTM, Bidirectional
```

```
from keras.preprocessing.text import Tokenizer
```

```
from keras.preprocessing.sequence import pad_sequences
```

```
from keras import Sequential
```

```
from sklearn.metrics import matthews_corrcoef, confusion_matrix, roc_curve, auc
```

```
from sklearn.model_selection import train_test_split
```

```
from sklearn import metrics
```

```
from sklearn.multiclass import OneVsRestClassifier
```

```
from sklearn.utils import shuffle
```

```
from sklearn.preprocessing import label_binarize
```

```
import numpy as np
```

```
import pickle
```

```
import matplotlib.pyplot as plt
```

```
from nltk.tokenize import TweetTokenizer
```

```
import warnings
```

```
import logging
```

```
import pandas as pd
```

```
from contextlib import
```

```
redirect_stdout, logging.basicConfig(level=logging.INFO)
```

```
#import dataframe and drop unnecessary columns
```

```

# mydata=pd.read_csv('FINAL_MONTH_.csv')
mydata=pd.read_csv('dataset.csv',skiprows=[0,1,2])
mydataframe=pd.DataFrame(mydata)
del mydataframe['Unnamed: 0']
del mydataframe['Unnamed: 0.1']
del mydataframe['Unnamed: 0.1.1']
#print(mydataframe.head())

#label encoder on the target variable
encoder=LabelEncoder()
mydataframe['label']=encoder.fit_transform(mydataframe['label'])

# the preprocessing function to prepare our text for training and testing
def model_preprocessing(train_data,test_data,MAX_WORDS=80000, MAX_SEQ_LEN=1000):
    np.random.seed(7)
    txt=np.concatenate((train_data,test_data),axis=0)
    txt=np.array(txt)
    tok=Tokenizer(num_words=MAX_WORDS)
    tok.fit_on_texts(txt)
    pickle.dump(tok,open('text_tokenizer_adam.pkl','wb'))
    seq=tok.texts_to_sequences(txt)
    word_index=tok.word_index
    txt=pad_sequences(seq,maxlen=MAX_SEQ_LEN)
    ind=np.arange(txt.shape[0])
    #np.random.shuffle(ind)

```

```

txt=txt[ind]
train_data_Glove=txt[0:len(train_data), ]
test_data_Glove=txt[len(train_data):, ]
dictionary_embedding={ }
func=open("glove.6B.50d.txt",encoding="utf8")
for l in func:
    val=l.split()
    wrd=val[0]
    try:
        coefficients=np.asarray(val[1:], dtype='float32')
    except:
        pass
    dictionary_embedding[wrd]=coefficients
func.close()
return(train_data_Glove,test_data_Glove,word_index,dictionary_embedding)

```

```

def model_building_blstm(word_index, dictionary_embedding,nclasses,
MAX_SEQ_LEN=1000, dropout=0.2, hidden_layer=3,
lstm_node=32,EMBEDDING_DIMS=50):
    blstm_model=Sequential() # model is sequential
    matrix_embedding=np.random.random((len(word_index)+1,EMBEDDING_DIMS)) # make
embedding matrix
    for wrd,i in word_index.items():
        vector_embedding=dictionary_embedding.get(wrd)
        if vector_embedding is not None:
            if len(matrix_embedding[i]) != len(vector_embedding):
                print("errorrrrr")
                exit(1)
            matrix_embedding[i]=vector_embedding

```

#Now add an embedding layer

```
blstm_model.add(Embedding(len(word_index)+1,EMBEDDING_DIMS,weights=[matrix_embedding],input_length=MAX_SEQ_LEN,trainable=True))
```

#Hidden layers - for every layer bi-lstm followed by dropout layer

for i in range (0,hidden_layer):

```
blstm_model.add(Bidirectional(LSTM(lstm_node,recurrent_dropout=0.2,return_sequences=True)))
```

```
blstm_model.add(Dropout(dropout))
```

```
blstm_model.add(Bidirectional(LSTM(lstm_node,recurrent_dropout=0.2,return_sequences=False)))
```

```
blstm_model.add(Dropout(dropout))
```

```
blstm_model.add(Dense(256,activation='relu')) #fully connected layer
```

```
blstm_model.add(Dense(nclasses,activation='softmax'))
```

```
blstm_model.compile(loss='sparse_categorical_crossentropy',optimizer='adam',metrics=['accuracy'])
```

```
return blstm_model
```

```
X=mydataframe['text']
```

```
Y=mydataframe['label']
```

```
train_data, test_data, train_data_y, test_data_y=train_test_split(X,Y,test_size=0.2)
```

```
train_data_Glove, test_data_Glove, word_index, dictionary_embedding=  
model_preprocessing(train_data,test_data)
```

```
blstm_model= model_building_blstm(word_index, dictionary_embedding,3)
```

```
#blstm_model=pickle.load(open('blstm_model.pkl','rb'))
```

```
blstm_model.summary()
```

```
#with open('bi-lstm_summary_adgrad.txt','w') as f:
```



```

with open('bi-lstm_summary_adam.txt','w') as f:
    with redirect_stdout(f):
        blstm_model.summary()
# TRAINING AND EVALUATING
def evaluating_report(labels,predictions):
    matt_coeff=matthews_corrcoef(labels,predictions)

    confusion_=confusion_matrix(labels,predictions)

    fp=confusion_.sum(axis=0)-np.diag(confusion_)
    fn=confusion_.sum(axis=1)-np.diag(confusion_)
    tp=np.diag(confusion_)
    tn=confusion_.sum()-(fp+fn+tp)

    precision=tp/(tp+fn)
    recall=tp/(tp+fn)
    f1=(2*(precision*recall))/(precision+ recall)
    return{
        "matthews_correlation_coefficient" : matt_coeff,
        "true positive": tp,
        "true negative":tn,
        "false positive":fp,
        "false negative":fn,
        "precision": precision,
        "recall": recall,
        "F1": f1,
        "accuracy": (tp+tn)/(tp+tn+fp+fn)
    }

```

```

    }

def computing_metrics(labels,predictions):
    assert len(predictions)==len(labels)
    return evaluating_report(labels,predictions)

def graph_plotting(history,string):
    plt.plot(history.history[string])
    plt.plot(history.history['val_' +string],")
    plt.xlabel('Epochs')
    plt.ylabel(string)
    plt.legend([string, 'val_'+string])
    plt.show()

history=blstm_model.fit(train_data_Glove,train_data_y,validation_data=(test_data_Glove,test_data_y),epochs=3,batch_size=256,verbose=1)

graph_plotting(history, 'accuracy')
graph_plotting(history, 'loss')

#pickle.dump(blstm_model,open('blstm_model_adam.pkl','wb'))

predicted=blstm_model.predict_classes(test_data_Glove)

report=metrics.classification_report(test_data_y, predicted,output_dict=True)

print(report)

report_dataframe=pd.DataFrame(report).transpose()

report_dataframe.to_csv('BLSTM_metrics_adam.csv')

print("\n")

logger=logging.getLogger('logger')

results=computing_metrics(test_data_y, predicted)

blstm_model=OneVsRestClassifier(blstm_model)

predicted=label_binarize(predicted,classes=[0,1,2])

num_classes=3

#ROC curve

f_pr=dict()

```

```

t_pr=dict()
ROC_AUC=dict()
test_data_y=label_binarize(test_data_y,classes=[0,1,2])
for i in range(num_classes):
    f_pr[i],t_pr[i], threshold=roc_curve(test_data_y[:,i],predicted[:,i])
    ROC_AUC[i]=auc(f_pr[i],t_pr[i])
for i in range(num_classes):
    plt.figure()
    plt.plot(f_pr[i],t_pr[i],label='ROC curve (area=%0.2f)' % ROC_AUC[i])
    plt.plot([0, 1], [0, 1], 'k--')
    plt.xlim([0.0, 1.0])
    plt.ylim([0.0, 1.05])
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
    plt.title('Receiver operating characteristic for class'+str(i))
    plt.legend(loc="lower right")
    plt.show()
plt.show()
for k in (results.keys()):
    logger.info('%s =%s',k,str(results[k]))
from sklearn.metrics import classification_report,accuracy_score
print(classification_report(test_data_y, predicted,digits=6,target_names=['0','1','2']))
print(f'ACCURACY of the BI-LSTM Model:{ accuracy_score(test_data_y,predicted)} ')

```

Bert Transformer Model:

```

import torch
if torch.cuda.is_available():
    device = torch.device("cuda")
    print(f'There are {torch.cuda.device_count()} GPU(s) available.')

```

```

print('Device name:', torch.cuda.get_device_name(0))

else:
    print('No GPU available, using the CPU instead.')
    device = torch.device("cpu")
import pandas as pd
import torch
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from keras.preprocessing.text import Tokenizer
import pickle
from keras.preprocessing.sequence import
pad_sequences torch.cuda.empty_cache()
mydata=pd.read_csv('merged_data.csv',skiprows=[0,1,2])
my_test_data = pd.read_csv('FINAL_DAY.csv')
val_data = pd.read_csv('validation Data.csv')
mydataframe=pd.DataFrame(mydata)
mytestdataframe = pd.DataFrame(my_test_data)
del mydataframe['Unnamed: 0']
del mydataframe['Unnamed: 0.1']
del mydataframe['Unnamed:0.1.1']
del mytestdataframe['Unnamed:0']
del mytestdataframe['Unnamed: 0.1']
del mytestdataframe['id']
train_data = mydataframe['text']
train_data_y=mydataframe['label']
train_data_set=pd.DataFrame(train_data)
train_data_set['label']=list(train_data_y)

```

```

test_data_set = pd.DataFrame(val_data)
test_data = test_data_set['text']
test_data_y=test_data_set['label']
from transformers import BertTokenizerFast
mytokenizer=BertTokenizerFast.from_pretrained('bert-base-uncased',do_lower=True)
def bert_data_preprocessing(values):
    input_ID = []
    attention_mask = []
    for data in values:
        encode_sent_data      =      mytokenizer.encode_plus(text=data      ,truncation=True,
add_special_tokens=True,
max_length=MAX_LEN,pad_to_max_length=True,return_attention_mask=True)
        input_ID.append(encode_sent_data.get('input_ids'))
        attention_mask.append(encode_sent_data.get('attention_mask'))
        attention_mask = torch.tensor(attention_mask)
        input_ID = torch.tensor(input_ID)
    return input_ID, attention_mask
seq_len = [len(i.split()) for i in train_data]
pd.Series(seq_len).hist(bins=30)
all_twitter_data = np.concatenate([train_data_set.text.values,test_data_set.text.values])
test_twitter_data = np.concatenate([mytestdataframe.text.values])
my_encoded_tweets=  [mytokenizer.encode(tweet,add_special_tokens=True)  for  tweet  in
all_twitter_data]
test_encoded_tweets  =  [mytokenizer.encode(tweet,add_special_tokens=True)  for  tweet  in
test_twitter_data]
MAX_LEN=40
token_ids = list(bert_data_preprocessing([mydataframe.text.values[0]])[0].squeeze().numpy())
print(X[0])
print(token_ids)
print("WE ARE TOKENIZING OUR DATA NOW")

```

```

train_input_id, train_attention_mask = bert_data_preprocessing(train_data)
test_input_id , test_attention_mask = bert_data_preprocessing(test_data)
testing_input_id,testing_attention_mask = bert_data_preprocessing(mytestdataframe['text'])
# Data Loader
from torch.utils.data import TensorDataset , DataLoader, RandomSampler,
SequentialSampler
training_labels = torch.tensor(list(train_data_y))
testing_labels = torch.tensor(list(test_data_y))
test_labels = torch.tensor(list(mytestdataframe['label']))
batch_size=430
print(train_input_id.shape)
print(train_attention_mask.shape)
print(training_labels.shape)
tensors_train = [train_input_id, train_attention_mask]
training_data = TensorDataset(train_input_id, train_attention_mask,training_labels)
training_sampler = RandomSampler(training_data)
training_dataloader = DataLoader(training_data , sampler = training_sampler,
batch_size=batch_size)
testing_data=TensorDataset(test_input_id, test_attention_mask,testing_labels)
testing_sampler = SequentialSampler(testing_data)
testing_dataloader=DataLoader(testing_data,sampler=testing_sampler,batch_size=batch_size)
test_data = TensorDataset(testing_input_id,testing_attention_mask,test_labels)
test_sampler = SequentialSampler(test_data)
test_dataloader = DataLoader(test_data , sampler = test_sampler, batch_size=batch_size)
# classifier
import torch
import torch.nn as tor_nn
from transformers import BertForSequenceClassification,BertModel
class ClassifierBert(tor_nn.Module):
    def __init__(self, freeze_bert=False):

```

```

super (ClassifierBert,self)._init()
bert_hidden, classifier_hidden , output_labels = 768 , 64 , 3
self.bert=BertModel.from_pretrained('bert-base-uncased')
iffreeze_bert:
    for parameter in self.bert.parameters():
        parameter.requires_grad=False
self.dropout = tor_nn.Dropout(0.6)
self.relu = tor_nn.ReLU()
self.fc1 = tor_nn.Linear(768,25)
self.fc4 = tor_nn.Linear(25,3)
self.softmax = tor_nn.LogSoftmax(dim=1)
def forward(self, input_ids, attention_mask):
    _,output = self.bert(input_ids=input_ids,attention_mask=attention_mask,return_dict=False)
    logits = self.fc1(output)
    logits=self.relu(logits)
    logits = self.dropout(logits)
    logits = self.fc4(logits)
    logits = self.softmax(logits)
    return logits
# optimizers and learning rate scheduler
from transformers import AdamW, get_linear_schedule_with_warmup
def model_initialization(epochs=10):
    classifier_bert = ClassifierBert(freeze_bert = False)
    classifier_bert=classifier_bert.to(device)
    optimizer = AdamW(classifier_bert.parameters(),
                        lr= 5e-6,
                        eps = 1e-8
                        )
    steps = len(training_dataloader) * epochs

```

```

scheduler_bert = get_linear_schedule_with_warmup(optimizer , num_warmup_steps=0,
        num_training_steps=steps
    )

return classifier_bert,optimizer,scheduler_bert

import random
import time
from torch.nn import functional as F
import tensorflow as tf
from sklearn.utils.class_weight import compute_class_weight
class_weights = compute_class_weight('balanced',np.unique(train_data_y),train_data_y)
weights_ = torch.tensor(class_weights, dtype = torch.float).to(device)
loss_function = torch.nn.CrossEntropyLoss(weight=weights_)

def set_seed(seed_val=42):
    random.seed(seed_val)
    np.random.seed(seed_val)
    torch.manual_seed(seed_val)
    torch.cuda.manual_seed_all(seed_val)

def model_training(model , training_dataloader , testing_dataloader=None , epochs=10 ,
evaluation=False):
    print('We are starting to train the model. .. \n')
    for epoch_i in range(epochs):
        print(f'{{'Epoch':^7}} | {{'Batch':^7}} | {{'Train Loss':^12}} | {{'Val Loss':^10}} | {{'Elapsed':^9}}")
        print('-'*60)
        t_epoch , t_batch = time.time() , time.time()
        loss_total , loss_batch , count_batch =0,0,0
        model.train()
        for step,batch in enumerate(training_dataloader):
            count_batch+=1
            bert_input_ids , bert_attention_mask , bert_label = tuple(t.to(device) for t in batch)

```



```

model.zero_grad()
logits=model(bert_input_ids, bert_attention_mask)
loss = loss_function(logits,bert_label)
loss_batch+=loss.item()
loss_total += loss.item()
loss.backward()
torch.nn.utils.clip_grad_norm_(model.parameters(),1.0)
optimizer.step()
scheduler_bert.step()
if (step%20 == 0 and step!=0) or (step==len(training_dataloader)-1):
    time_elapsed = time.time() - t_batch
    print(f'{epoch_i + 1:^7} | {step:^7} | {loss_batch / count_batch:^12.6f} | {'-':^10} |
{time_elapsed:^9.2f}')
    loss_batch , count_batch = 0,0
    t_batch = time.time()
    print('-'*70)
    average_training_loss = loss_total/len(training_dataloader)
    if evaluation == True:
        if (step==len(training_dataloader)-1):
            loss_training,loss_accuracy = model_evaluating(model,testing_dataloader)
            time_elapsed = time.time() - t_epoch
            print(f'{epoch_i + 1:^7} | {'-':^7} | {average_training_loss:^12.6f} | {loss_training:^10.6f}
| {time_elapsed:^9.2f}')
            print("-"*70)
        print("\n")
    print("Training complete!")

```

```

def model_evaluating(model,testing_dataloader):

```

```

model.eval()
loss_accuracy = []
loss_training = []
for batch in testing_dataloader:
    bert_input_ids , bert_attention_mask, bert_label = tuple(t.to(device) for t in batch)
    with torch.no_grad():
        logits = model(bert_input_ids,bert_attention_mask)
        loss = loss_function(logits,bert_label)
        loss_training.append(loss.item())
        prediction = torch.argmax(logits , dim=1).flatten()
        accuracy = (prediction==bert_label).cpu().numpy().argmax().mean()*100
        loss_accuracy.append(accuracy)
loss_training = np.mean(loss_training)
loss_accuracy = np.mean(loss_accuracy)
return loss_training ,
loss_accuracyset_seed(42)
classifier_bert, optimizer, scheduler_bert = model_initialization(epochs=100)
model_training(classifier_bert,                                     training_dataloader,
testing_dataloader,epochs=100,evaluation=True)
def evaluate(model,testing_dataloader):
    model.eval()
    loss_validation=0
    predictions,true_value = [],[]
    for batch in testing_dataloader:
        bert_input_ids , bert_attention_mask, bert_label = tuple(t.to(device) for t in batch)
        with torch.no_grad():

            logits = model(bert_input_ids,bert_attention_mask)
            #logits =output.logits

```

```

#
loss=loss_function(logits , bert_label)
loss_validation+=loss.item()
logits = logits.detach().cpu().numpy()
label_ids = bert_label.cpu().numpy()
predictions.append(logits)
true_value.append(label_ids)

average_validation_loss = loss_validation/len(testing_dataloader)

predictions = np.concatenate(predictions ,axis=0)
true_value = np.concatenate(true_value,axis=0)
return predictions,true_value

from sklearn.metrics import f1_score,
classification_report,accuracy_score,log_loss,precision_score,recall_score

possible_labels = mydataframe.label.unique()
label_dict = { }
for index, possible_label in enumerate(possible_labels):
    label_dict[possible_label] = index
label_dict

def f1_score_evaluation(pred,labels):
    pred_flat = np.argmax(pred,axis=1).flatten()
    label_flat = labels.flatten()
    print(f1_score(label_flat , pred_flat , average = 'weighted'))
    print(accuracy_score(label_flat , pred_flat))
def accuracy_score_evaluation(pred,labels):
    pred_flat = np.argmax(pred,axis=1).flatten()
    label_flat = labels.flatten()
    return accuracy_score(label_flat ,pred_flat)
def classification_report_evaluation(pred,labels):
    pred_flat =np.argmax(pred,axis=1).flatten()

```

```

label_flat = labels.flatten()
print(classification_report(label_flat , pred_flat))
def accuracy_of_classes(pred,labels):
    label_dict_inverse = {v:k for k,v in label_dict.items()}
def overall_evaluation(pred,labels):
    pred_flat = np.argmax(pred,axis=1).flatten()
    label_flat = labels.flatten()
    print(f"ACCURACY:{accuracy_score(label_flat , pred_flat)}')
    print(f"F1 SCORE: {f1_score(label_flat , pred_flat, average='weighted')}')
    print(f"PRECISION:{precision_score(label_flat , pred_flat, average='weighted')}')
    print(f"RECALL:{recall_score(label_flat , pred_flat, average='weighted')}')
    print(classification_report(label_flat , pred_flat,digits=6))
    bert_metrics=classification_report(label_flat , pred_flat,digits=6,output_dict=True)
    bert_metrics_dataframe = pd.DataFrame(bert_metrics).transpose()
    print(bert_metrics)
predictions1,true_value1 = evaluate(classifier_bert,testing_data_loader)
overall_evaluation(predictions1,true_value1)

```

XLNet Model:

```

import pandas as pd
import torch
import numpy as np
from keras.preprocessing.sequence import
pad_sequences
torch.cuda.empty_cache()
#import dataframe and drop unnecessary columns
mydata=pd.read_csv('merged_data.csv',skiprows=[0,1,2])
myvaldata=pd.read_csv('validation Data.csv')
mydataframe=pd.DataFrame(mydata)

```

```

myvaldf = pd.DataFrame(myvaldata)
del mydataframe['Unnamed: 0']
del mydataframe['Unnamed: 0.1']
del mydataframe['Unnamed: 0.1.1']
del myvaldf['Unnamed: 0']
del myvaldf['Unnamed: 0.1']
del myvaldf['Unnamed: 0.1.1']
X=mydataframe['text']
Y=mydataframe['label']
text_lists = mydataframe['text'].values
val_text_list = myvaldf['text'].values
sentences = [sentence + " [SEP] [CLS]" for sentence in text_lists]
val_sentence = [sentence + " [SEP] [CLS]" for sentence in val_text_list]
labels = mydataframe['label'].values
val_labels = myvaldf['label']
from transformers import XLNetTokenizerFast
!pip install sentencepiece
myTokenizer = XLNetTokenizerFast.from_pretrained('xlnet-base-cased',do_lower_case=True)
tokenized_tweets = [myTokenizer.tokenize(data) for data in sentences]
val_tokenized_tweets = [myTokenizer.tokenize(data) for data in val_sentence]
print(tokenized_tweets[0])
MAX_LEN=35
input_ids = [myTokenizer.convert_tokens_to_ids(x) for x in tokenized_tweets]
val_input_ids = [myTokenizer.convert_tokens_to_ids(x) for x in val_tokenized_tweets]
input_ids = pad_sequences(input_ids,maxlen=25,dtype="long",truncating="post",padding="post")
val_input_ids = pad_sequences(val_input_ids,maxlen=25,dtype="long",truncating="post",padding="post")
val_attention_masks,attention_masks=[],[]
for data in input_ids:

```

```

    att = [float(i>0) for i in data]
attention_masks.append(att)
for data in val_input_ids:
    att = [float(i>0) for i in data]
val_attention_masks.append(att)
train_input = torch.tensor(input_ids)
validation_input = torch.tensor(val_input_ids)
train_labels = torch.tensor(labels)
validation_labels = torch.tensor(val_labels)
train_att_mask = torch.tensor(attention_masks)
val_att_mask = torch.tensor(val_attention_masks)
batch_size=450

from torch.utils.data import TensorDataset , DataLoader, RandomSampler,
SequentialSampler
training_data = TensorDataset(train_input, train_att_mask,train_labels)
training_sampler = RandomSampler(training_data)

training_dataloader = DataLoader(training_data,
sampler=training_sampler,batch_size=batch_size)

validating_data = TensorDataset(validation_input, val_att_mask,validation_labels)
validating_sampler = RandomSampler(validating_data)

validation_dataloader = DataLoader(validating_data,
sampler=validating_sampler,batch_size=batch_size)

from transformers import XLNetModel,
XLNetForSequenceClassification,XLNetForTokenClassification

xlNetModel = XLNetForSequenceClassification.from_pretrained('xlnet-base-
cased',num_labels=3)

xlNetModel.cuda()

parameter_optimizer = list(xlNetModel.named_parameters())
no_decay = ['bias' , 'gamma' , 'beta']
grouped_parameters_optimizer = [

    {'params':[p for n, p in parameter_optimizer if not any(nd in n for nd in
no_decay)]},

```

```

        'weight_decay_rate': 0.01
    },
    {'params':[p for n, p in parameter_optimizer if any(nd in n for nd in
no_decay)],
        'weight_decay_rate': 0.0
    }
]

from transformers import AdamW

optimizer = AdamW(grouped_parameters_optimizer,lr=5e-5)

def model_accuracy(predictions, labels):
    predictions_flatten = np.argmax(predictions, axis=1).flatten()
    labels_flatten = labels.flatten()
    return np.sum((predictions_flatten==labels_flatten)/len(labels_flatten))

from tqdm import tqdm, trange

from transformers import get_linear_schedule_with_warmup

def modelTraining(model , training_dataloader, validating_dataloader=None, epochs=5 ,
evaluation=False, optimizer=optimizer):
    training_loss = []
    print("-----MODELTRAINING ----- \n")
    scheduler =
    get_linear_schedule_with_warmup(optimizer,num_warmup_steps=0,num_training_steps =
len(training_dataloader) * epochs)
    for _ in trange(epochs , desc="Epoch"):
        model.train()
        training_l=0
        num_train_exp, num_train_steps=0,0
        for step, batch in enumerate(training_dataloader):
            batch = tuple(t.to(device) for t in batch)
            xlnet_input_ids , xlnet_input_mask , xlnet_labels = batch
            optimizer.zero_grad()

```

```

    output = model(xlnet_input_ids, token_type_ids=None, attention_mask =
xlnet_input_mask, labels = xlnet_labels)
    loss = output[0]
    logits = output[1]
    training_loss.append(loss.item())
    loss.backward()
    optimizer.step()
    scheduler.step()
    training_l+=loss.item()
    num_train_exp+=xlnet_input_ids.size(0)
    num_train_steps+=1
    print("TRAINING LOSS: { }\n".format(training_l/num_train_steps))
    if (evaluation==True):
        evaluating_model(model, validating_dataloader)
def evaluating_model(model,validating_dataloader):
    model.eval()
    validation_loss, validation_accuracy = 0,0
    num_val_steps, num_val_exp = 0,0
    for batch in validating_dataloader:
        batch = tuple(t.to(device) for t in batch)
        xlnet_input_ids , xlnet_input_mask , xlnet_labels = batch
        with torch.no_grad():
            output = model(xlnet_input_ids, token_type_ids=None, attention_mask = xlnet_input_mask)
            logits = output[0]
        logits = logits.detach().cpu().numpy()
        label_ids = xlnet_labels.to('cpu').numpy()
        temp_val_accuracy = model_accuracy(logits,label_ids)
        validation_accuracy+=
        temp_val_accuracynum_val_steps+=1

```



```

print("Validation Accuracy: {}".format(validation_accuracy/num_val_steps))

def predict_xlnet(model, validating_dataloader):
    model.eval()
    validation_loss=0
    predictions , true_values = [],[]
    for batch in validating_dataloader:
        batch = tuple(t.to(device) for t in batch)
        xlnet_input_ids , xlnet_input_mask , xlnet_labels = batch
        with torch.no_grad():
            output = model(xlnet_input_ids, token_type_ids=None,attention_mask = xlnet_input_mask)
            logits = output[0]
        logits = logits.detach().cpu().numpy()
        label_ids = xlnet_labels.to('cpu').numpy()
        predictions.append(logits)
        true_values.append(label_ids)
    predictions = np.concatenate(predictions ,axis=0)
    true_values = np.concatenate(true_values,axis=0)
    return predictions,true_values

from sklearn.metrics import f1_score,
classification_report,accuracy_score,log_loss,precision_score,recall_score

def overall_evaluation(pred,labels):
    pred_flat = np.argmax(pred,axis=1).flatten()
    label_flat = labels.flatten()
    print(f"ACCURACY:{accuracy_score(label_flat , pred_flat)}")
    print(f"F1 SCORE: {f1_score(label_flat , pred_flat, average='weighted')}")
    print(f"PRECISION:{precision_score(label_flat , pred_flat, average='weighted')}")
    print(f"RECALL:{recall_score(label_flat , pred_flat, average='weighted')}")
    print(classification_report(label_flat , pred_flat,digits=6))
    xlnet_metrics=classification_report(label_flat , pred_flat,digits=6,output_dict=True)

```

```

xlnet_metrics_dataframe = pd.DataFrame(xlnet_metrics).transpose()
print(xlnet_metrics)
predictions,true_values = predict_xlnet(xlNetModel,validation_dataloader)
overall_evaluation(predictions,true_values)

```

FINAL ANALYSIS:

```

import torch
import torch.nn as tor_nn
from transformers import *
class ClassifierBert(tor_nn.Module):
    def __init__(self, freeze_bert=False):
        super (ClassifierBert,self).__init__()
        bert_hidden, classifier_hidden , output_labels = 768 , 64 , 3
        model_name = "bert_classifier.pt"
        path = F"/content/drive/My Drive/{model_name}"
        state_bert = torch.load(path)
        self.bert = BertModel.from_pretrained('bert-base-uncased',state_dict=state_bert)
        if freeze_bert:
            for parameter in self.bert.parameters():
                parameter.requires_grad=False
        self.dropout = tor_nn.Dropout(0.6)
        self.relu = tor_nn.ReLU()
        self.fc1 = tor_nn.Linear(768,25)
        self.fc4 = tor_nn.Linear(25,3)
        self.softmax = tor_nn.LogSoftmax(dim=-1)
    def forward(self, input_ids, attention_mask):
        _,output = self.bert(input_ids=input_ids,attention_mask=attention_mask,return_dict=False)
        logits = self.fc1(output)
        logits=self.relu(logits)

```

```

logits = self.dropout(logits)
logits = self.fc4(logits)
logits = self.softmax(logits)
return logits

def model_initialization(epochs=10):
    classifier_bert = ClassifierBert(freeze_bert = False)
    classifier_bert=classifier_bert.to(device)
    return classifier_bert

import pandas as pd
test_data = pd.read_csv('post_vaccination_twitter_data.csv',names=['created_at','id','text'])
test_df = pd.DataFrame(test_data)
del test_df['id']
del test_df['created_at']
data = list(test_df['text'])
from transformers import
BertTokenizermodel_name =
"bert_classifier.pt"
path = F"/content/drive/My Drive/{model_name}"
state_bert = torch.load(path)
mytokenizer=BertTokenizer.from_pretrained('bert-base-
uncased',do_lower=True,state_dict=state_bert)
test_input_id = []
test_attention_mask = []
test_df.dropna()
texts = test_df["text"].to_list()
for value in texts:
    encode_sent_data = mytokenizer.encode_plus(text=str(value),truncation=True,
add_special_tokens=True,
max_length=35,pad_to_max_length=True,return_attention_mask=True)
    test_input_id.append(encode_sent_data.get('input_ids'))

```

```
test_attention_mask.append(encode_sent_data.get('attention_mask'))
```

```

test_attention_mask = torch.tensor(test_attention_mask)
test_input_id = torch.tensor(test_input_id)
from torch.utils.data import TensorDataset , DataLoader, RandomSampler,
SequentialSamplerbatch_size=1
test_data = TensorDataset(test_input_id , test_attention_mask)
test_sampler = SequentialSampler(test_data)
test_dataloader = DataLoader(test_data , sampler = test_sampler, batch_size=batch_size)
import numpy as np
def evaluate(model,testing_dataloader):
    model.eval()
    predictions,true_value = [],[]
    for batch in testing_dataloader:
        bert_input_ids , bert_attention_mask = tuple(t.to(device) for t in batch)
        with torch.no_grad():
            logits = model(bert_input_ids,bert_attention_mask)
            logits = logits.detach().cpu().numpy()
            predictions.append(logits)
    predictions = np.concatenate(predictions ,axis=0)
    return predictions
predictions = evaluate(bert_classifier,test_dataloader)
predictions = torch.tensor(predictions)
pred=torch.exp(predictions).numpy()
label=[]
for data in pred:
    label.append(data.argmax())
#print(label)
test_df['label'] = list(label)
test_df.to_csv('predicted_post_covid.csv')

```

LOCKDOWN VS POST-LOCKDOWN AND POST-VACCINATION COMAPRISON:

```
import pandas as pd

pre_vacc_file = pd.read_csv('merged_data.csv',skiprows=[0,1,2])
post_vacc_file = pd.read_csv('predicted_post_covid.csv',skiprows=[1])
pre_vacc_df = pd.DataFrame(pre_vacc_file)
post_vacc_df = pd.DataFrame(post_vacc_file)

# De-code 0,1,2 to neutral, depressed, non-depressed
pre_vacc_df['label'] = pre_vacc_df['label'].replace([0,1,2],['neutral','depressed','non-depressed'])
post_vacc_df['label'] = post_vacc_df['label'].replace([0,1,2],['neutral','depressed','non-depressed'])

pre_vacc_df.head()
post_vacc_df.head()

pre_vacc_df['label'].value_counts().plot.pie(label =
",figsize=(5,5),colors=['red','purple','blue'],title='DISTRIBUTION OF TWEETS DURING COVID')

post_vacc_df['label'].value_counts().plot.pie(label =
",figsize=(5,5),colors=['blue','red','purple'],title='DISTRIBUTION OF TWEETS POST-VACCINATION')

import itertools
import collections
import numpy as np

pre_vacc_texts=[]
for word in pre_vacc_df['text']:
    pre_vacc_texts.append(word)

pre_vacc_words = [text.split() for text in pre_vacc_texts]
pre_vacc_words[:5]

pre_vacc_all_words = list(itertools.chain(*pre_vacc_words))
pre_vacc_word_counts = collections.Counter(pre_vacc_all_words)
pre_vacc_word_counts.most_common(15)

import nltk
```

```

from nltk.corpus import
stopwordsnltk.download('stopwor
ds') post_vacc_texts = []
for word in post_vacc_df['text']:
    post_vacc_texts.append(word)
post_vacc_words = [text.lower().split() for text in post_vacc_texts]
post_vacc_words[:5]
post_vacc_all_words = list(itertools.chain(*post_vacc_words))
common_words = ['i','get','trump', "b'fauci",'hopes','getting','one','covid-
19','i'\xe2\x80\x99m',"b'i",'tell','people','see','vaccine.','vaccine','i'm','everyone','got','&','th
e','really','many','don'\xe2\x80\x99t']
post_vacc_all_word = [word for word in post_vacc_all_words if word not in common_words]
post_vacc_all_word = [word for word in post_vacc_all_word if word not in
stopwords.words('english')]
post_vacc_word_counts = collections.Counter(post_vacc_all_word)
post_vacc_word_counts.most_common(15)
pre_vacc_freq_df =
pd.DataFrame(pre_vacc_word_counts.most_common(15),columns=['tweet_word','word_count'])
pre_vacc_freq_df.sort_values(by='word_count').plot.barh(x='tweet_word',y='word_count',title='
COMMON WORDS FOUND DURING PANDEMIC LOCKDOWN',color='pink')
post_vacc_freq_df =
pd.DataFrame(post_vacc_word_counts.most_common(15),columns=['tweet_word','word_count']
)
post_vacc_freq_df.sort_values(by='word_count').plot.barh(x='tweet_word',y='word_count',title='
COMMON WORDS FOUND POST-LOCKDOWN AND POST-
VACCINATION',color='skyblue')

```


PLAGIARISM REPORT

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY				
(Deemed to be University u/s 3 of UGC Act, 1956)				
Office of Controller of Examinations				
REPORT FOR PLAGIARISM CHECK ON THE DISSERTATION / PROJECT REPORTS FOR UG/ PG PROGRAMMES				
1	Name of the candidate (IN BLOCK LETTERS)	ADITYA AGARWAL		
2	Address of the candidate	DEPARTMENT OF INFORMATION TECHNOLOGY, SRMIST, KTR CAMPUS		
3	Registration number	RA1711008010343		
4	Date of Birth	06 TH AUGUST 1999		
5	Department	Information Technology		
6	Faculty	Engineering and Technology		
7	Title of the Dissertation / Project	TWEET SENTIMENTAL ANALYSIS OF COVID TOWARDS MENTAL DEPRESSION USING DEEP LEARNING		
8	Whether the above dissertation is done by	Individual / Group a) If group, number of students: 2 b) Name and Register Numbers of other candidates: NASEELA PARVEZ(RA1711008010096)		
9	Name and address of the Supervisor/ Guide	DR. S.SURESH, PROFESSOR Email ID: sureshs3@srmist.edu.in phone: 8939991130		
10	Name and address of the CO-Supervisor/ Co-guide (if any)	-		
11	Software used	Turnitin		
12	Date of Verification	11 th may 2021		
13	Plagiarism Details: (to attach the final report from the software)			
Chapter	Title of the Chapter	Percentage of similarity index (including self citations)	Percentage of similarity index (excluding self citations)	Percentage of plagiarism excluding Quotes, Bibliography, etc
	ABSTRACT	-	-	0%

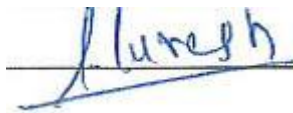
1	INTRODUCTION	-	-	2%
2	LITERATURE REVIEW	-	-	1%
3	SYSTEM ANALYSIS	-	-	6%
4	SYSTEM DESIGN	-	-	2%
5	CODING AND TESTING	-	-	1%
6	RESULTS AND ANALYSIS			5%
7	CONCLUSION AND FUTURE WORK	-	-	2%



Signature of the Candidate

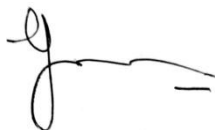


Name and Signature of the Staff who uses the plagiarism software



Name and Signature of Guide :

Name and Signature of Co - Guide :



Dr. G. VADIVU
Head
Department of Information Technology
Faculty of Engineering & Technology
SRM Institute of Science and Technology
SRM Nagar, Kattankulathur - 603 203.
Kancheepuram Dist., Tamil Nadu, India.

Name and signature of the Head of Department

Appendices			
We declare that the above information has been verified and found true to the best of our knowledge.			

Abstract

ORIGINALITY
REPORT

0%
SIMILARITY INDEX

0%
INTERNET
SOURCES

0%
PUBLICATION
S

0%
STUDENT PAPERS

PRIMARY SOURCES

Exclude quotes On
Exclude bibliography On

Exclude matches < 10 words

Chapter 1

ORIGINALITY REPORT

2%
SIMILARITY INDEX

0%
INTERNET
SOURCES

0%
PUBLICATIONS

2%
STUDENT PAPERS

PRIMARY SOURCES

1 Submitted to Leeds Trinity and All Saints **2%**
Student Paper

Exclude quotes On
Exclude bibliography On

Exclude matches < 10 words

Chapter 2

ORIGINALITY REPORT

1 %

SIMILARITY INDEX

X

0%

INTERNET
SOURCES

1%

PUBLICATION
S

0%

STUDENT PAPERS

PRIMARY SOURCES

1

HarvinderJeetKaur,RajivKumar."Sentiment
analysisfromsocialmediaincrisissituations",
International Conference on Computing,
Communication & Automation, 2015

Publication

1 %

Exclude quotes
Exclude bibliography

On
On

Exclude matches

< 10 words

chapter 3

ORIGINALITY REPORT

6%

SIMILARITY INDEX

X

4%

INTERNET
SOURCES

4%

PUBLICATION
S

4%

STUDENT PAPERS

PRIMARY SOURCES

1

en.wikipedia.org

Internet Source

<1%

2

arxiv.org

Internet Source

<1%

3

Submitted to University of West London

Student Paper

<1%

4

medium.com

Internet Source

<1%

5

"Intelligent Natural Language Processing: Trends and Applications", Springer Science and Business Media LLC, 2018

Publication

<1%

6

Submitted to Heriot-Watt University

Student Paper

<1%

7

Submitted to Indian School of Mines

Student Paper

<1%

8

users.dsic.upv.es

Internet Source

<1%

9	YuliangLi,AaronFeng,JinfengLi,Saran Mumick, Alon Halevy, Vivian Li, Wang-Chiew Tan. "Subjective databases", Proceedings of the VLDB Endowment, 2019 Publication	<1%
10	"ECAI 2020", IOS Press, 2020 Publication	<1%
11	Submitted to University of Hong Kong Student Paper	<1%
12	Submitted to University of Glasgow Student Paper	<1%
13	Submitted to University of Westminster Student Paper	<1%
14	Michael Zenou. "", IEEE Photonics Technology Letters, 11/2007 Publication	<1%
15	export.arxiv.org Internet Source	<1%
16	www.mdpi.com Internet Source	<1%

Excludequotes On
Exclude bibliography On

Excludematches < 10words

Chapter 4

ORIGINALITY REPORT

2%
SIMILARITY INDEX

2%
INTERNET
SOURCES

0%
PUBLICATION
S

0%
STUDENT PAPERS

PRIMARY SOURCES

1 www.essaysauce.com
Internet Source

2%

Exclude quotes On
Exclude bibliography On

Exclude matches < 10 words

Chapter 5

ORIGINALITY REPORT

1 %
SIMILARITY INDEX

1 %
INTERNET
SOURCES

0 %
PUBLICATION
S

0 %
STUDENT PAPERS

PRIMARY SOURCES

1	auroracs.lk	1 %
	InternetSource	

Exclude quotes	On
Exclude bibliography	On

Exclude matches	< 10 words
-----------------	------------

Chapter 6

ORIGINALITY REPORT

5%

SIMILARITY INDEX

X

4%

INTERNET
SOURCES

4%

PUBLICATION
S

3%

STUDENT PAPERS

PRIMARY SOURCES

1

Statham, M.M.. "Intraoperative PTH: Effect of sample timing and vitamin D status", Otolaryngology - Head and Neck Surgery, 2007 06

Publication

<1%

2

medium.com

Internet Source

<1%

3

bmcmedgenomics.biomedcentral.com

Internet Source

<1%

4

Md. Mahbubur Rahman, Rifat Sadik, Al Amin Biswas. "Bangla Document Classification using Character Level Deep Learning", 2020 4th International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT), 2020

Publication

<1%

5

www.hbs.edu

Internet Source

<1%

6

docplayer.net

Internet Source

<1%

7	"Data Analytics and Management", Springer Science and Business Media LLC, 2021 Publication	<1%
8	seerc.org InternetSource	<1%
9	Prakash, Surya, and PhalguniGupta. "Fusion of Ear with Other Traits", Augmented Vision and Reality, 2015. Publication	<1%
10	link.springer.com Internet Source	<1%
11	www.actuariesindia.org Internet Source	<1%

Exclude quotes On
Exclude bibliography On

Exclude matches < 10 words

Chapter 7

ORIGINALITY REPORT

2%
SIMILARITY INDEX

2%
INTERNET
SOURCES

0%
PUBLICATIONS

0%
STUDENT PAPERS

PRIMARY SOURCES

1 erepository.uonbi.ac.ke **2%**
Internet Source

Exclude quotes On
Exclude bibliography On

Exclude matches < 10 words

Btech Report

ORIGINALITY REPORT

4%

SIMILARITY
INDEX

3%

INTERNET
SOURCES

3%

PUBLICATIO
NS

2%

STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to Indian School of Mines Student Paper	<1 %
2	"ECAI 2020", IOS Press, 2020 Publication	<1 %
3	arxiv.org Internet Source	<1 %
4	Submitted to Leeds Trinity and AllSaints Student Paper	<1 %
5	"Data Analytics and Management", Springer Science and Business Media LLC,2021 Publication	<1 %
6	Statham, M.M.. "Intraoperative PTH: Effect of sample timing and vitamin D status", Otolaryngology - Head and Neck Surgery, 200706 Publication	<1 %
7	Submitted to University of West London Student Paper	<1 %

9	bmcmedgenomics.biomedcentral.com Internet Source	<1 %
10	citeseerx.ist.psu.edu Internet Source	<1 %
11	"Intelligent Natural Language Processing: Trends and Applications", Springer Science and Business Media LLC, 2018 Publication	<1 %
12	auroracs.lk Internet Source	<1 %
13	Submitted to Heriot-Watt University Student Paper	<1 %
14	users.dsic.upv.es Internet Source	<1 %
15	"Machine Learning and Information Processing", Springer Science and Business Media LLC, 2021 Publication	<1 %
16	Faiz Ul Islam, Guangjie Liu, JiangtaoZhai, Weiwei Liu. "VoIP Traffic Detection in Tunnelled and Anonymous Networks Using Deep Learning", IEEE Access, 2021 Publication	<1 %
17	Md. Mahbubur Rahman, Rifat Sadik, Al Amin Biswas. "Bangla Document	

Classification using Character Level Deep Learning",2020

Student Paper

<1 %

4th International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT), 2020

Publication

18

www.hbs.edu

Internet Source

<1 %

19

Yuliang Li, Aaron Feng, Jinfeng Li, Saran Mumick, Alon Halevy, Vivian Li, Wang-Chiew Tan. "Subjective databases", Proceedings of the VLDB Endowment, 2019

Publication

<1 %

20

docplayer.net

Internet Source

<1 %

21

Submitted to University of HongKong

Student Paper

<1 %

22

seerc.org

InternetSource

<1 %

23

Prakash, Surya, and Phalguni Gupta. "Fusion of Ear with Other Traits", Augmented Vision and Reality, 2015.

Publication

<1 %

24

Submitted to University of Glasgow

Student Paper

<1 %

25

Submitted to University of Westminster

<1 %

26	Michael Zenou. "", IEEE Photonics Technology Letters, 11/2007 Publication	<1 %
27	erepository.uonbi.ac.ke Internet Source	<1 %
28	export.arxiv.org Internet Source	<1 %
29	link.springer.com Internet Source	<1 %
30	www.essaysauce.com Internet Source	<1 %
31	www.actuariesindia.org Internet Source	<1 %

Excludequotes On
Excludebibliography

Excludematches

< 10words