Educate Elevate Enlighten

JSS Mahavidyapeetha
JSS Science And Technology University
(Established Under JSS Science and Technology University Act No. 43 of 2013)
(Formerly Known as SJCE)

JSS SCIENCE AND TECHNOLOGY UNIVERSITY MYSURU

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

DEEP LEARNING BASED DETECTION OF DEPRESSION

Project report submitted in partial fulfillment of curriculum prescribed for the award of the degree of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

Submitted By:

Sl.No	Name	USN	Section	Roll No
1	Akshay Suryanarayan Hegde	01JST17CS014	A	08
2	Ganesh S	01JST17CS054	A	23
3	Aniket Kharad	01JST17CS018	A	09
4	Manzoor Ahmed	01JST17CS084	C	23

Under the guidance of

Prof. Ashritha R Murthy

(Assistant Professor)

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

2020-2021

Educate Elevate Enlighten

JSS Mahavidyapeetha

JSS Science And Technology University

(Established Under JSS Science and Technology University Act No. 43 of 2013)

(Formerly Known as SJCE)

JSS SCIENCE AND TECHNOLOGY UNIVERSITY MYSURU

CERTIFICATE

This is to certify that the project titled "***Deep Learning Based Detection of Depression*** " is presented by Akshay Suryanaryan Hegde, Ganesh S and Aniket Kharad, Manzoor Ahmed for partial fulfillment of the award of the degree of Bachelor of Engineering in Department of Computer Science and Engineering, JSS Science & Technological University, Mysuru during the year 2020-21. It is certified that all corrections/ suggestions indicated during report submission have been incorporated in the report. The project report has been approved as it satisfies the academic requirements in respect of project prescribed for the Bachelor of Engineering degree.

Signature of guide

____

(Prof. Ashritha R Murthy)

Panel Members	**Signature**
1. Prof. Ashritha R Murthy	_______________
2. _______________	_______________
3. _______________	_______________
4. _______________	_______________

Date: **Signature of HOD**

Declaration

We hereby declare that the project work entitled ”**Deep Learning Based Detection of Depression**” , is a record of an original work done by us under the guidance of Prof. Ashritha R Murthy, Assistant Professor, Dept. of CSE, JSS Science & Technological University, Mysuru and this project is submitted in partial fulfillment of the requirements for the award of the degree of Bachelor of Engineering in Computer Science and Engineering. The results embodied in this thesis have not been submitted to any other University or Institute for the award of any degree or diploma.

- Akshay Suryanaryan Hegde
- Ganesh S
- Aniket Kharad
- Manzoor Ahmed

Abstract

With busy life, increasing daily activities and the dawn of the pandemic like COVID-19 people tend to forget about their mental health, with these busy lives people find themselves with anxiety, stress, depression related problems. Depression is a mental illness which is the mixed feelings of sadness, feeling low, stressed, mood swings. According to WHO approximately 264 million people across all ages suffer from depression. Patients of these mental illnesses generally turn towards social media sites like Facebook, Twitter, Reddit to share their feelings with the community and try to seek emotional support. This is the motivation to capture the user posts and identify the person's psychological problems as early as possible. There are also chances people don’t express themselves on social media sites. With our solution discussed in the paper it is difficult to reach out to them with medical guidance.

Acknowledgement

It gives us immense pleasure to write an acknowledgement to this project, a contribution of all the people who helped to realize it. We extend our deep regards to Dr.S.B.Kivade, Honorable Principal of JSS Science Technology University, for providing an excellent environment for our education and his encouragement throughout our stay in college. We would like to convey our heartfelt thanks to our HOD, Dr.M.P.Pushpalatha, for giving us the opportunity to embark upon this topic. We would like to thank our project guide, Prof. Ashritha R Murthy for their invaluable guidance and enthusiastic assistance and for providing us support and constructive suggestions for the betterment of the project, without which this project would not have been possible. We appreciate the timely help and kind cooperation of our lecturers, other staff members of the department and our seniors, with whom we have come up all the way during our project work without whose support this project would not have been successful. Finally, we would like to thank our friends for providing numerous insightful suggestions. We also convey our sincere thanks to all those who have contributed to this learning opportunity at every step of this project.

Contents

Abstract 2

Acknowledgement 3

1. Introduction 6
    - 1.1 Aim/ statement of the problem 6
    - 1.2 Objectives of the project work 6
    - 1.3 Introduction to the problem domain 6
    - 1.4 Applications 7
        - 1.4.1 Monitor Family members mental health 7
        - 1.4.2 Analyse the data of Depressed People 8
        - 1.4.3 Clinical treatment design 8
        - 1.4.4 Design precautions to avoid depression 8
    - 1.5 Existing solution methods 8
    - 1.6 Proposed solution methods 8
    - 1.7 Time schedule for completion of the project work 9

2. Literature survey 10

3. System requirements and analysis 13
    - 3.1 Functional Requirements 13
    - 3.2 Non Functional Requirements 14

4. Tools and technology 15

5. System design 21
    - 5.1 Collection of the Data 21
    - 5.2 Data preprocessing 22
    - 5.3 Network Architecture 23

6. System implementation 25
    - 6.1 Integrating Google Drive with Google Colab 25
    - 6.2 Fetching the Data from Google Drive 26
    - 6.3 Preprocessing the Data 26
    - 6.4 Tokenizing the Data 28
    - 6.5 Building the Embedding Matrix 28
    - 6.6 Developing Machine Learning Models 29
    - 6.7 Developing Hybrid Deep Learning Models 30
        - 6.7.1 Long Short Term Memory (LSTM) 31

6.7.2 Gated Recurrent Units (GRU) 32

7. System testing and result analysis 34
7.1 Result Analysis 34
7.2 Testing 37
7.2.1 Unit Testing 37
7.2.2 System Testing 38

8. Conclusion and future work 39

Appendix A : Project team details 40

Appendix B : COs,POs and PSOs mapping for the project work 41
10.1 Course Outcomes 41
10.2 Program Outcomes 41
10.3 Program Specific Outcomes 41
10.4 Justification for the mapping 42

Appendix C : Publication Details 44

References 45

1. Introduction

1.1 Aim/ statement of the problem

A widespread psychiatric illness is depression. It has a direct and indirect effect on economic development. Not only for those affected, but also for their families and their social and work related environments, depression also has significant consequences. It can be the psychological basis for symptoms of panic and anxiety. Increasingly, panic disorder has been focused on health care and the media, impacting young people aged 20-40. Social media is increasingly used by different levels of age groups. Patients of psychiatric disorders also turn to online social media and web forums for specific conditions and emotional support information. Although social media can be used to transform the life of an individual as a really beneficial tool, it can create certain conflicts that can have a negative effect. The practices turn out to be extremely difficult with the growing number of users and their content. This motivates for detecting depression from the user posts. The main idea is to detect such psychological problems from user's posts as early as possible. This leads us to develop Deep Learning based Detection of Depression.

1.2 Objectives of the project work

• The first step is to pre-process the data brought from text mining into a form that is predictable and analyzable for depression detection.

• It is imperative for extracting features from the structured textual data for natural language processing.

• With a large pool of features we have to generate models using different machine learning and deep learning techniques.

• To analyze and optimize the models for assessment for depressive symptoms of an individual.

1.3 Introduction to the problem domain

There are multiple ways to detect depression by using text data, audio data, video data, but text data proves to be of more significance as an explosion of users in the social media sites

connect various people and they tend to post pictures, comments on these sites. In this paper we are considering text data in order to detect depression in individuals.
Text classification, also known as text categorization, is a classical problem in natural language processing (NLP), which aims to assign labels or tags to textual units such as sentences, queries, paragraphs, and documents. It has a wide range of applications including question answering, spam detection, sentiment analysis, news categorization, user intent classification, content moderation, and so on. Text data can come from Different sources, including web data, emails, chats, social media, tickets, insurance claims, user reviews, and questions and answers from customer services, to name a few. Text is an extremely rich source of information. But extracting insights from text can be challenging and time-consuming, due to its unstructured nature. Text classification can be performed either through manual annotation or by automatic labeling. With The growing scale of text data in industrial applications, automatic text classification is becoming increasingly important.
Approaches to automatic text classification can be grouped into categories:

- Rule-based methods
- Machine Learning(data-driven) based method

In this project, we aim at detailed research and analysis on detecting depression through text, from social media by applying machine learning and deep learning algorithms. We start with collecting data from various social media sources by mining texts from tools like TWINT. Then we perform data preprocessing by cleaning and converting raw text into required format which can be further fed into the machine learning which uses various machine learning algorithms and deep learning models as well. But the dataset being very huge, we propose deep learning algorithms on the processed data, which seems to be appropriate and serves the purpose of applying the model to a huge corpus. With tuned hyperparameters we architect deep layered models. Finally, we analyze all the algorithms through various evaluation metrics, make a comparative analysis, evaluate results to choose the best algorithm to detect the depression.

1.4 Applications

1.4.1 Monitor Family members mental health

By using the project which we design, people can check mental health of family members, whether their family members or friends are having any depression symptoms or have depression, if project detects depression symptoms in any one of their family or friend, then they can help the person with depression and can try to cure him by taking him to rehab centers or hospitals or can provide help to get out of depression.

1.4.2 Analyse the data of Depressed People

Experts can feed their data to the project and can get the tweets or text which show depression and find the people with depression and then from that data they can get region wise depressed people data or reason wise depressed people data and can use it for further experiments to find solutions to solve the problem of depression.

1.4.3 Clinical treatment design

Doctors or medical experts can use the project to feed data of particular patients and find the reasons for their depression and can give the proper treatment to the depressed patients and cure them. This will save the time of doctors to find the reason for the patient's depression and they can give treatment at the early stages only before depression becomes a serious issue for the patient.

1.4.4 Design precautions to avoid depression

Experts of government or Medical research scientists or Corporate companies can use our application and find out the reasons for the depression of the people or employees and can design precautions to stop people from get depressed, like giving vacations to employees or conducting entertaining competitions, or educating people how to stay mentally healthy and how to lead a happy life and etc.

1.5 Existing solution methods

The existing solution is based on taking the predefined dataset from kaggle or other websites, applying feature extraction and implementing some Machine Learning and Deep Learning old models.

1.6 Proposed solution methods

The proposed solution is based on the following

- Dataset is retrieved by using Automation on Twitter using TWINT
- Data is cleaned and features are extracted using NLP
- Machine Learning Models are applied like Random Forest, Naïve Bayes
- Deep Learning Models are applied using CNN, CNN + LSTM, CNN + GRU
- Result is analysed with various metrics of accuracy

1.7 Time schedule for completion of the project work

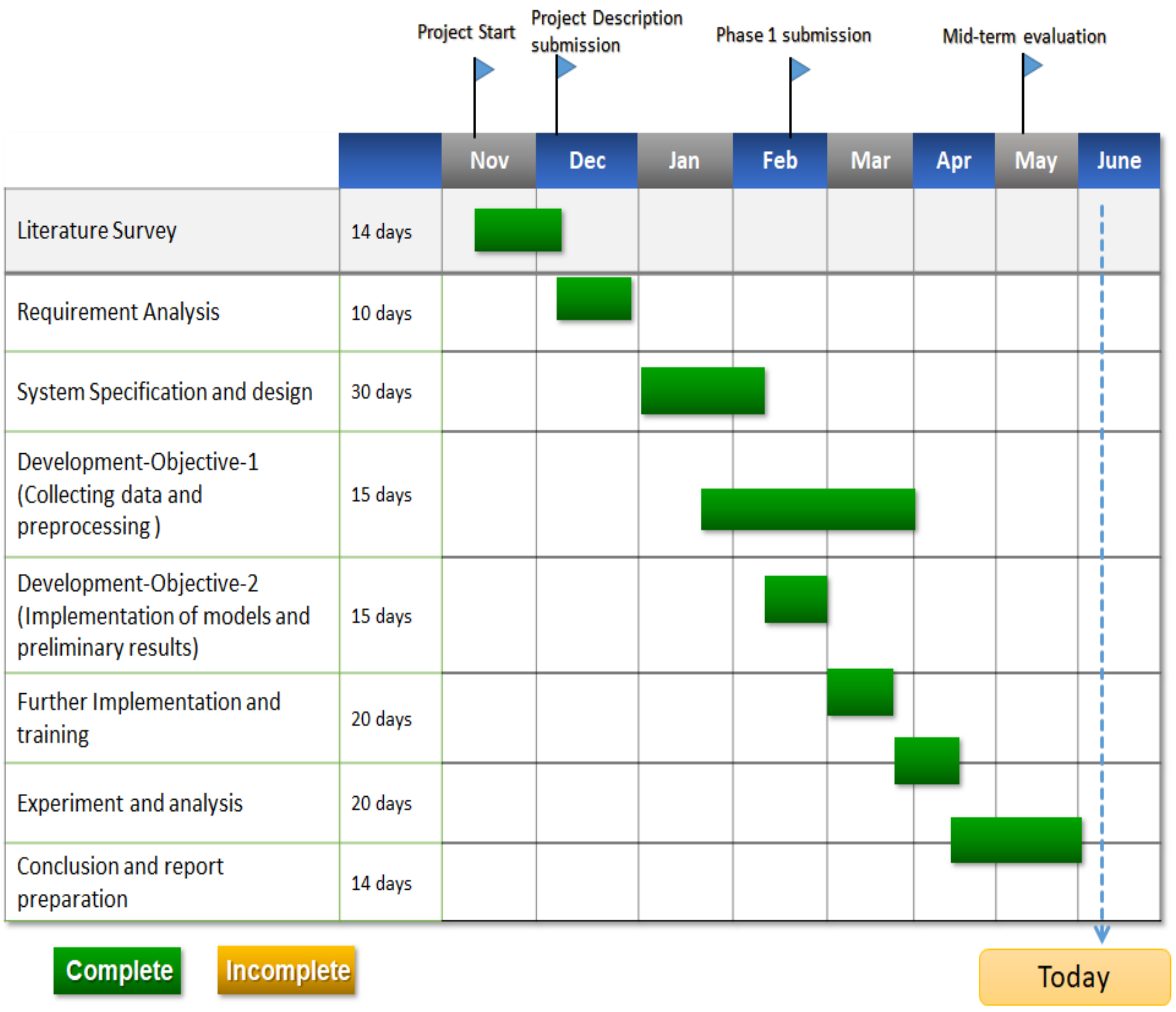


Figure 1.7.1 Gantt Chart

2. Literature survey

According to DEY, SHARMISTHA, et. al., [14] more than 264 million people across the globe suffer from mental health conditions and depression, between 76% to 85% of lower and middle income groups don't receive any treatment for depression. The author also mentions that since the dawn of the beginning of the internet people have begun to dump their thoughts on the Internet which may prove to be a very powerful tool for the diagnosis of a person's mental health problems.

[14],[15],[3] have given a survey on various algorithms applied on depressions data which includes Machine Learning, Deep Learning, Data Mining etc. also mentioned about various trends in the usage of Random Forest, Hidden Markov Model, Naïve Bayes models for the better analysis of the model.

Mandar Deshpande, Vignesh Rao [15] have proposed a concrete idea on data extraction and data preprocessing using Natural Language Processing. The model proposed incorporates SVM, Multinomial Naïve Bayes as the primary algorithms. Author has taken into consideration F1-Score, Precision and Recall as the accuracy measures.

Md. Rafqul Islam et. al., [3] proposed a model which uses various Machine Learning Algorithms like KNN, SVM, Decision Tree and variations of it. The model has made use of Ensemble learning techniques to increase the performance and Accuracy of the model. There was also a special mention about the tool called LIWC which is used to extract relevant data from Social Media Sites. In the feature extraction step, there is a thorough analysis of characteristics of the data.

Hao Guoa et. al., [16] proposed a model on the basis of "Resting state functional brain networks" which has been widely studied in brain disease, Resting state functional brain networks were constructed for 38 major depressive disorders. The model has an average accuracy of 79.27% and 78.22% for SVM and Neural Network with RBF kernel respectively, with 28 features.

DEY, SHARMISTHA, et. al., [14] has proposed a cumulative survey of various models and approaches which they have taken and summarized in the form of a table and it is shown in the below table which gives us clear insights on how different models give different accuracies.

Author	Year	Approach	Platform Used	Sample Size	Performance and Future Scope
Eichstaedt C. J.[1]	2017	Logistics Regression with 10-fold cross validation	Facebook	683 patients, 114 depressed	
Aldarwish M.M., Hafiz F. A[4]	2017	SVM and Naïve Bayes Classifier).	Twitter, Facebook	2073 post indicating depression and 2073 posts indicating non-Depression	The authors have calculated accuracy, precision and recall.
Reece G. A. and et. al.,[6]	2017	Random forest, Hidden Markov Model	Twitter	204 persons (105 having depression, 99 healthy), 74,990 daily observations	85% accuracy they have achieved
Islam R, Kabir A, Wang H and Ulhaq[2]	2019	Decision Tree, KNN, Support Vector Machine	Facebook	7145 comments, comments that shows depression: 4149, Non depression indicative comments: 2996	Decision tree Recall 98% Precision 58%
Jonathon C and et. al,[3]	2018	probability calculation, using 10- fold cross validation	Facebook post	683, 114 of whom is detected with depression	AUC is 0.69, give same performance as manual survey

Table 2.1 - Literature Review

Jana M. Havigerova et.al methodology emphasis more on quantitative linguistic characteristics which are suited based on pronominalisation index and readiness to action index in men and sentence complexity and punctuation for women. Making use of this peculiar feature of Text the author has proposed various models.

Thirteen linguistic variables (6 single morpho-syntactic characteristics, 7 indexes combining more morphosyntactic characteristics) were included into the predictive models. Eight predictive models (for 4 different texts and 2 genders) were created and compared with each other.Author has considered gender depression for the purposes of analysing the models and specifies there are different symptoms based on the gender.

Tejus R. S. et.al [12] proposed possible CNN + LSTM model and other machine learning models to find stressful tweets from different kinds of datasets. As the paper was under research during early stages of pandemic, it includes comparative study between the stress level and local demographic level disaster against before and during initial days of pandemic. Where it is found out that stress level indicating stress levels had risen by 30 times whereas disaster related by 10 times.

In the paper proposed by Shervin Minaee et.al, [13] they have surveyed over 150 DL models, which were developed in the past 6 years and have significantly improved state of the art on various TC tasks. They also provide an overview of more than 40 popular TC datasets, and present a quantitative analysis of the performance of these models on several public benchmarks.

3. System requirements and analysis

3.1 Functional Requirements

Python

The implementation is done in Python using Anaconda. Python is a high level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English words frequently whereas other languages use punctuation, and it has fewer syntactic constructions than other languages. Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative, functional and procedural, and has a large and comprehensive standard library.

Google Colaboratory for running the models

Colaboratory (also known as Colab) is a free Jupyter notebook environment that runs in the cloud and stores its notebooks on Google Drive. Colab was originally an internal Google project; an attempt was made to open source all the code and work more directly upstream, leading to the development of the ”Open in Colab” Google Chrome extension, but this eventually ended, and Colab development continued internally. Julia language can also work on Colab (with e.g. Python and GPUs; Google’s tensor processing units also work with Julia on Colab)

Functional requirements define the internal workings of the software: that is, the technical details, data manipulation and processing and other specific functionality that show how the use cases are to be satisfied. They are supported by non-functional requirements, which impose constraints on the design or implementation.

1. **Software Configuration**
   a. This application will be developed by using python for creating and training of models.
   b. Operating System: Mac Os,Windows 8 9 10,Linux
   c. Language: Python
   d. Environments: Google colab,Anaconda Jupyter notebook

2. **Hardware Configuration**
   The training of this project was done on the Google Colab platform and utilized a remote system which is a part of the Google Cloud Platform as well as on a local machine for initial stages of development. The hardware requirements for this project are mainly the system on which the code was written and used to access the remote system using a browser. The hardware requirements are as follows for remote development with local machine specifications :
   1. Processor 64 bit, 4 core, 2.40 GHZ
   2. RAM: 8 GB
   3. Input Device: Standard keyboard and mouse
   4. Output Device: Monitors with decent resolution
   5. Internet: A stable connection to the internet for Google Colab

   The local system can be used for training and testing purpose and must at least meet the below mentioned specifications:
   1. Processor: Quad core Intel i7 Skylake or higher(Dual core is manageable)
   2. RAM: Minimum 8 GB of RAM is required.
   3. STORAGE: 500 GB of Hard disk(SDD is preferred)
   4. GPU: Premium Graphics card like NVIDIA 9x or 10x series.

Our model uses deep learning,NLP, feature extractions which are highly CPU intensive tasks. So minimum specifications of the Laptop is essential and our solution is compatible across all the Operating systems installed with the Functional Requirements specified.
But building a complex Deep Learning Model with a lot of hidden layers may prove to be a challenge for the computation.So to accommodate these changes or complex features we have to extend the RAM capacity and also GPU for smooth performance and also a CPU with high computational power, currently these high end systems seems to be infeasible but we can incorporate these once we get our hands on the required resources.
But overall for building a model with good accuracy the current systems and Hardware and Software seems to be sufficient which leads to the technical feasibility of the Project.

3.2 Non Functional Requirements

Non-functional requirements are requirements which specify criteria that can be used to judge the operation of a system, rather than specific behaviors. This should be contrasted with functional requirements that specify specific behavior or functions.
Typical non-functional requirements are reliability, scalability, and cost. Non-functional requirements are often called the ilities of a system. Other terms for non-functional requirements are”constraints”,”quality attributes”and”quality of service requirements”.

1. Reliability: If any exceptions occur during the execution of the software it should be caught and thereby prevent the system from crashing.
2. Scalability: The system should be developed in such a way that new modules and functionalities can be added, thereby facilitating system evolution.
3. Cost : The cost should be low because of the free availability of software packages and frameworks.
4. Security: The system should be developed such that the data available should be secured from the external harms.

4. Tools and technology

Tools and technologies utilized for this project are vscode, jupyter lab, google colab, for creating these models, with pair with local disk or google drive for cloud storage, the backend comprises of modules such as nltk, gensim, keras for processing and matplotlib for visualization. The complete project is coded in python.

Tensorflow

TensorFlow is an open source software library for numerical computation using data flow graphs. Nodes in the graph represent mathematical operations, while the graph edges represent the multidimensional data arrays (tensors) communicated between them. The flexible architecture allows you to deploy computation to one or more CPUs or GPUs in a desktop, server, or mobile device with a single API. It is used for machine learning applications such as neural networks. It is used for both research and production at Google, often replacing its closed-source predecessor, DistBelief.

TensorFlow computations are expressed as stateful dataflow graphs.The name Tensor- Flow derives from the operations that such neural networks perform on multidimensional data arrays. These arrays are referred to as ”tensors”.

TensorFlow separates the definition of computations from their execution even further by having them happen in separate places: a graph defines the operations, but the operations only happen within a session. Graphs and sessions are created independently. To do efficient numerical computing in Python, we typically use libraries like NumPy that do expensive operations such as matrix multiplication outside Python, using highly efficient code implemented in another language. Unfortunately, there can still be a lot of overhead from switching back to Python every operation. This overhead is especially bad if you want to run computations on GPUs or in a distributed manner, where there can be a high cost to transferring data. TensorFlow also does its heavy lifting outside Python, but it takes things a step further to avoid this overhead. Instead of running a single expensive operation independently from Python, TensorFlow lets us describe a graph of interacting operations that run entirely outside Python.

Numerical Python (NumPy)

NumPy is the fundamental package for scientific computing in Python. It is a Python library that provides a multidimensional array object, various derived objects (such as masked arrays and matrices), and an assortment of routines for fast operations on arrays, including mathematical, logical, shape manipulation, sorting, selecting, I/O, discrete Fourier transforms, basic linear algebra, basic statistical operations, random simulation and much more. At the core of the NumPy package, is the ndarray object. This encapsulates n-dimensional arrays of homogeneous data types, with many operations being performed in compiled code for performance. There are several important differences between NumPy arrays and the standard Python sequences:

1. NumPy arrays have a fixed size at creation, unlike Python lists (which can grow dynamically). Changing the size of an ndarray will create a new array and delete the original.
2. The elements in a NumPy array are all required to be of the same data type, and thus will be the same size in memory. The exception: one can have arrays of (Python, including NumPy) objects, thereby allowing for arrays of different sized elements
3. NumPy arrays facilitate advanced mathematical and other types of operations on large numbers of data. Typically, such operations are executed more efficiently and with less code than is possible using Python's built-in sequences.
4. A growing plethora of scientific and mathematical Python-based packages are using NumPy arrays; though these typically support Python Sequence input, they convert such input to NumPy arrays prior to processing, and they often output NumPy array

In other words, in order to efficiently use much (perhaps even most) of today's scientific/mathematical Python-based software, just knowing how to use Python's built-in sequence types is insufficient - one also needs to know how to use NumPy arrays.

Visual studio code

Features include support for debugging, syntax highlighting, intelligent code completion, snippets, code refactoring, and embedded Git. Users can change the theme, keyboard shortcuts, preferences, and install extensions that add additional functionality. The source code is free and open-source, released under the permissive MIT License. The compiled binaries are freeware for any use. In the Stack Overflow 2019 Developer Survey, Visual

Studio Code was ranked the most popular developer environment tool, with 50.7 percent of 87,317 respondents claiming to use it.

Matplotlib

Matplotlib is a plotting library for the Python programming language and its numerical mathematics extension NumPy. It provides an object-oriented API for embedding plots into applications using general-purpose GUI toolkits like Tkinter, wxPython, Qt, or GTK+. There is also a procedural ”pylab” interface based on a state machine (like OpenGL), designed to closely resemble that of MATLAB, though its use is discouraged. It used to create 2D graphs and plots by using python scripts. It has a module named pyplot which makes things easy for plotting by providing features to control line styles, font properties, formatting axes etc. It supports a very wide variety of graphs and plots namely - histogram, bar charts, power spectra, error charts etc. It is used along with NumPy to provide an environment that is an effective open source alternative for MatLab. It can also be used with graphics toolkits like PyQt and wxPython. For simple plotting the pyplot module provides a MATLAB-like interface, particularly when combined with IPython. For the power user, you have full control of line styles, font properties, axes properties, etc, via an object oriented interface or via a set of functions familiar to MATLAB users.

Keras

Keras is an open-source neural-network library written in Python. It is capable of running on top of TensorFlow, Microsoft Cognitive Toolkit, R, Theano, or PlaidML. Designed to enable fast experimentation with deep neural networks, it focuses on being user-friendly, modular, and extensible. It was developed as part of the research effort of project ONEIROS (Open-ended Neuro-Electronic Intelligent Robot Operating System), and its primary author and maintainer is Fran¸cois Chollet, a Google engineer. Chollet also is the author of the XCeption deep neural network model.

In 2017, Google’s TensorFlow team decided to support Keras in TensorFlow’s core library. Chollet explained that Keras was conceived to be an interface rather than a standalone machine learning framework. It offers a higher-level, more intuitive set of abstractions that make it easy to develop deep learning models regardless of the computational backend used. Microsoft added a CNTK backend to Keras as well, available as of CNTK v2.0. Keras contains numerous implementations of commonly used neural-network building blocks such as layers, objectives, activation functions, optimizers, and a host of tools to make working with image and text data easier to simplify the coding necessary for writing deep neural network code. The code is hosted on GitHub, and community support forums include the GitHub issues page, and a Slack channel. In addition to standard neural networks, Keras has support for convolutional and recurrent neural networks. It supports other common utility layers like dropout, batch normalization, and pooling. Keras allows users to productize deep models on smartphones (iOS and Android), on the web, or on the Java Virtual Machine. It also allows use of distributed training of deep-learning models on clusters of Graphics processing units (GPU) and tensor processing units (TPU) principally in conjunction with CUDA.

Word2Vec

This tool provides an efficient implementation of the continuous bag-of-words and skip-gram architectures for computing vector representations of words. These representations can be subsequently used in many natural language processing applications and for further research.

Word2vec is a two-layer neural net that processes text by “vectorizing” words. Its input is a text corpus and its output is a set of vectors: feature vectors that represent words in that corpus. While Word2vec is not a deep neural network, it turns text into a numerical form that deep neural networks can understand. Word2vec’s applications extend beyond parsing sentences in the wild. It can be applied just as well to genes, code, likes, playlists, social media graphs and other verbal or symbolic series in which patterns may be discerned. Because words are simply discrete states like the other data mentioned above, and we are simply looking for the transitional probabilities between those states: the likelihood that they will co-occur. So gene2vec, like2vec and follower2vec are all possible. With that in mind, the tutorial below will help you understand how to create neural embeddings for any group of discrete and co-occurring states. The purpose and usefulness of Word2vec is to group the vectors of similar words together in vector space. That is, it detects similarities mathematically. Word2vec creates vectors that are distributed numerical representations of word features, features such as the context of individual words. It does so without human intervention.

Given enough data, usage and contexts, Word2vec can make highly accurate guesses about a word’s meaning based on past appearances. Those guesses can be used to establish a word’s association with other words (e.g. “man” is to “boy” what “woman” is to “girl”), or cluster documents and classify them by topic. Those clusters can form the basis of search, sentiment analysis and recommendations in such diverse fields as scientific research, legal discovery, e-commerce and customer relationship management. The output of the Word2vec neural net is a vocabulary in which each item has a vector attached to it, which can be fed into a deep-learning net or simply queried to detect relationships between words. Measuring cosine similarity, no similarity is expressed as a 90 degree angle, while total similarity of 1 is a 0 degree angle, complete overlap; i.e. Sweden equals Sweden, while Norway has a cosine distance of 0.760124 from Sweden, the highest of any other country. Word embeddings contain the potential of being very useful, even fundamental to many NLP tasks, not only traditional text, but also genes, programming languages(seriously, has anyone tried that?), and other types of languages.

Word Embeddings is an active research area trying to figure out better word representations than the existing ones. But, with time they have grown large in number and more complex. This article was aimed at simplifying some of the workings of these embedding models without carrying the mathematical overhead. If you think that I was able to clear some of your confusion, comment below. Any changes or suggestions would be welcomed.

GloVe

GloVe is an unsupervised learning algorithm for obtaining vector representations for words. Training is performed on aggregated global word-word co-occurrence statistics from a corpus, and the resulting representations showcase interesting linear substructures of the word vector space. GloVe is essentially a log-bilinear model with a weighted least-squares objective. The main intuition underlying the model is the simple observation that ratios of word-word co-occurrence probabilities have the potential for encoding some form of meaning. For example, consider the co-occurrence probabilities for target words ice and steam with various probe words from the vocabulary.

As one might expect, ice co-occurs more frequently with solid than it does with gas, whereas steam co-occurs more frequently with gas than it does with solid. Both words co-occur with their shared property water frequently, and both co-occur with the unrelated word fashion infrequently. Only in the ratio of probabilities does noise from non-discriminative words like water and fashion cancel out, so that large values (much greater than 1) correlate well with properties specific to ice, and small values (much less than 1) correlate well with properties specific to steam. In this way, the ratio of probabilities encodes some crude form of meaning associated with the abstract concept of thermodynamic phase. The training objective of GloVe is to learn word vectors such that their dot product equals the logarithm of the words' probability of co-occurrence. Owing to the fact that the logarithm of a ratio equals the difference of logarithms, this objective associates (the logarithm of) ratios of co-occurrence probabilities with vector differences in the word vector space. Because these ratios can encode some form of meaning, this information gets encoded as vector differences as well. For this reason, the resulting word vectors perform very well on word analogy tasks, such as those examined in the word2vector package.

Recent methods for learning vector space representations of words have succeeded in capturing fine-grained semantic and syntactic regularities using vector arithmetic, but the origin of these regularities has remained opaque. We analyze and make explicit model properties needed for such regularities to emerge in word vectors. The result is a new global log bilinear regression model that combines the advantages of the two major model families in the literature: global matrix factorization and local context window methods. Our model efficiently leverages statistical information by training only on the nonzero elements in a word-word co occurrence matrix, rather than on the entire sparse matrix or on individual context windowsinalargecorpus. Themodelproducesavectorspacewithmeaningfulsubstructure, as evidenced by its performance of 75% on a recent word analogy task. It also outperforms related models on similarity tasks and named entity recognition.

Recently, considerable attention has been focused on the question of whether distributional word representations are best learned from count-based methods or from prediction-based methods. Currently, prediction-based models garner substantial support; for example, Baroni et al. (2014) argue that these models perform better across a range of tasks. In this work we argue that the two classes of methods are not dramatically different at a fundamental level since they both probe the underlying co-occurrence statistics of the corpus, but the efficiency with which the count-based methods capture global statistics can be advantageous. We construct a model that utilizes this main benefit of count data while simultaneously capturing the meaningful linear substructures prevalent in recent log-bilinear prediction-based methods like word2vec. The result, GloVe, is a new global log-bilinear regression model for the unsupervised learning of word representations that outperforms other models on word analogy, word similarity, and named entity recognition tasks.

5. System design

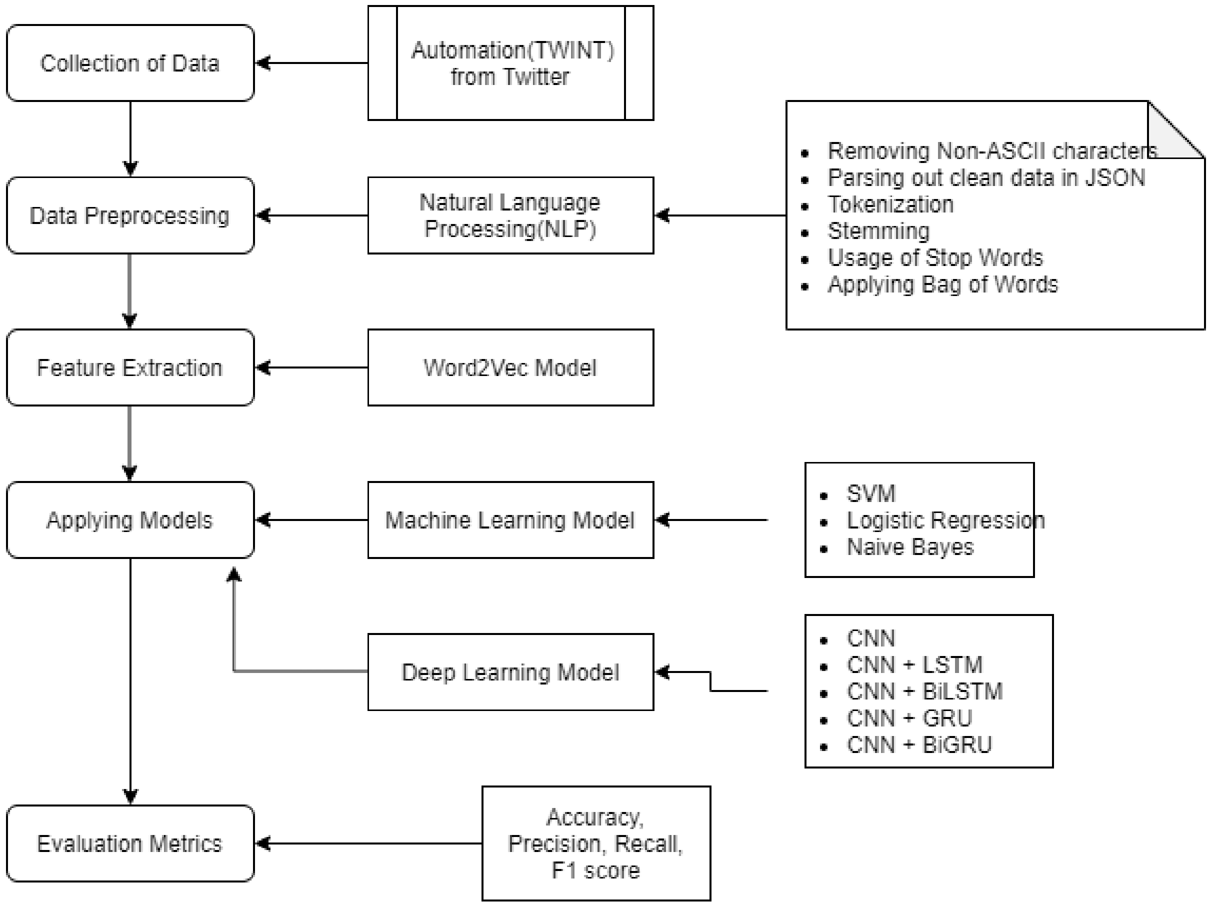


Figure 5.1 - System design of the application

As shown in the Fig 5.1 the project has various processes involved before making the full model and analysing the results.It undergoes various preprocessing before applying the model, So here is the details of the same.

5.1 Collection of the Data

The data was mainly collected from Twitter using TWINT automation tool where we specified the date from which we want the tweets and specify a particular keyword based on which the entire twitter was queried and to look for such tweets and the same was extracted and stored in various directories based on the severity of the words chosen.
Directory1 - No Depression - Random words were specified
Directory2 - Mild Depression - Keywords like stress, anxiety, low, trauma were used to query Twitter and the outcome was stored in the Directory named mild symptoms.

Directory3 - Critical Depression - Keywords like Depression, PTSD, Double Depression, etc were used to query Twitter and the results of these were stored in a Directory named Critical Depression.

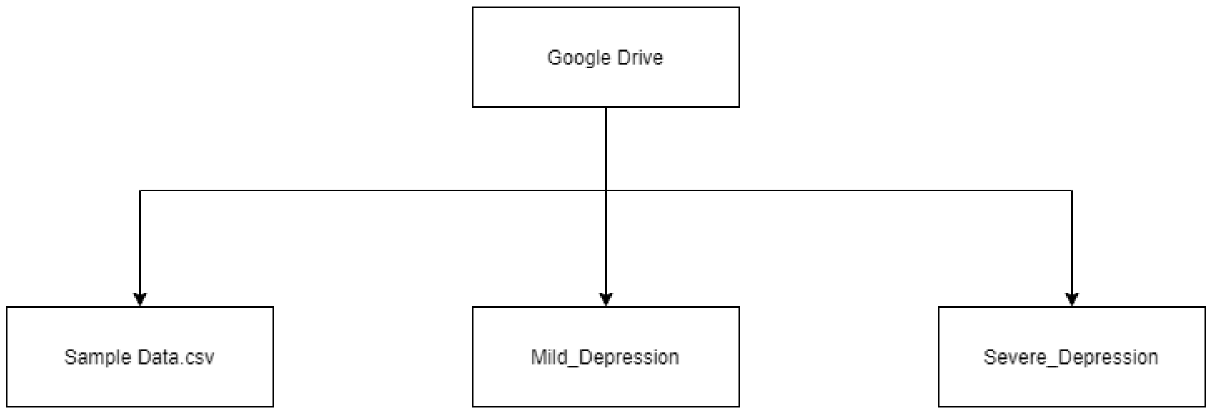


Figure 5.1.1 - Storing Dataset

5.2 Data preprocessing

A common text preprocessing technique is to remove the punctuations from the text data. This is again a text standardization process that will help to treat 'hurray' and 'hurray!' in the same way. Some of the punctuations include '!', ';' ,'$' etc.

Stopwords are commonly occuring words in a language like 'the', 'a' and so on. They can be removed from the text most of the time, as they don't provide valuable information for downstream analysis. In cases like Part of Speech tagging, we should not remove them as they provide very valuable information about the POS.

Stemming is the process of reducing inflected (or sometimes derived) words to their word stem, base or root form,For example, if there are two words in the corpus walks and walking, then stemming will stem the suffix to make them walk.
With more and more usage of social media platforms, there is an explosion in the usage of emojis in our day to day life as well. Probably we might need to remove these emojis for some of our textual analysis.

Next preprocessing step is to remove any URLs present in the data. For example, if we are doing a twitter analysis, then there is a good chance that the tweet will have some URL in it. Probably we might need to remove them for our further analysis.

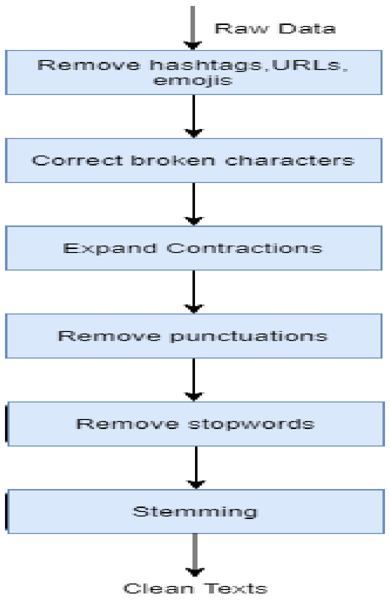


Figure 5.2.1 - Data Preprocessing

5.3 Network Architecture

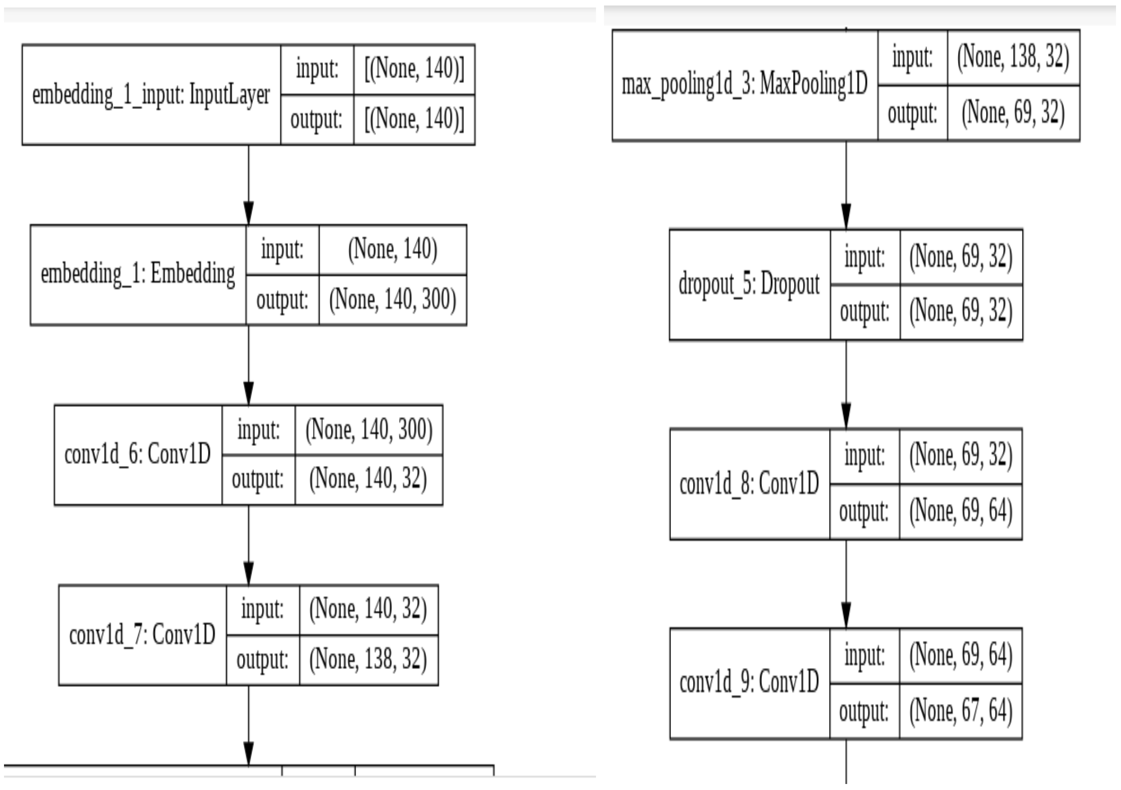


Figure 5.3.1 CNN Network Architecture

The Figures 5.3.1 and 5.3.2 shows the Network Architecture of the Deep Learning Model which includes various Convolutional Layers, Hidden Layers, Dropout, Maxpooling Layers.

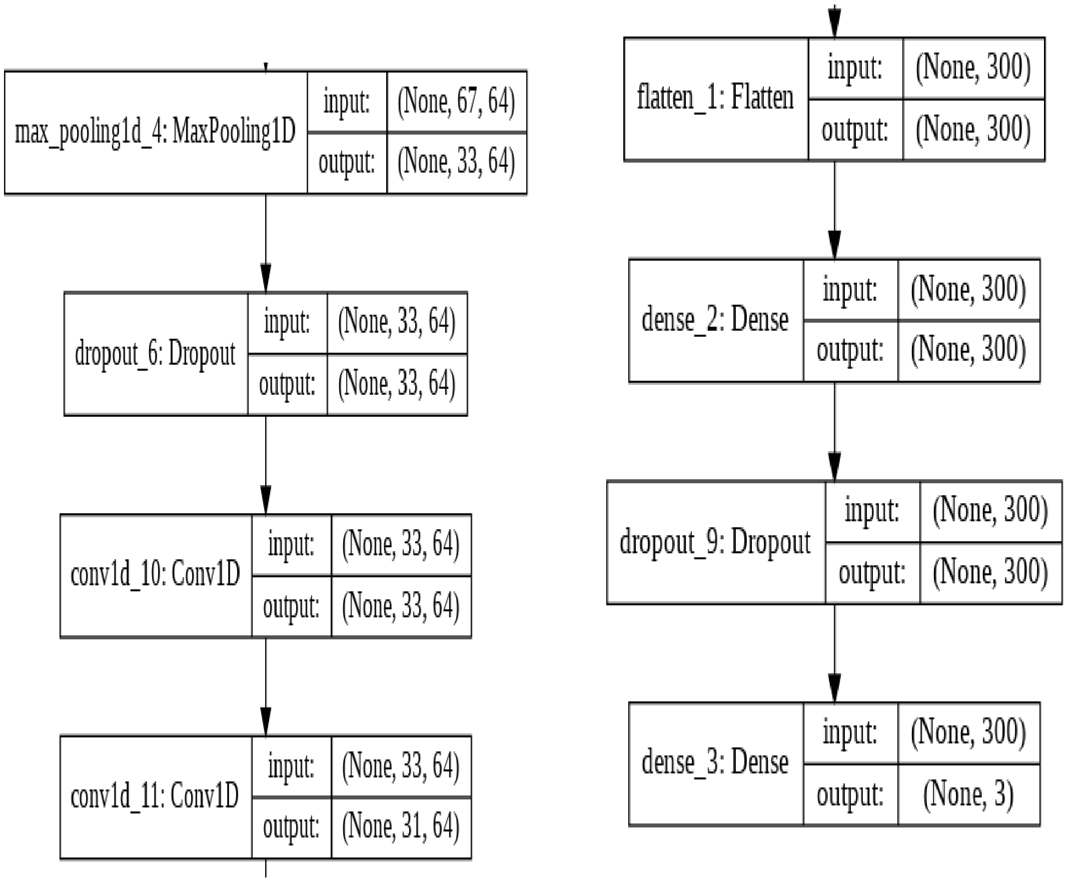


Figure 5.3.2 CNN Network Architecture

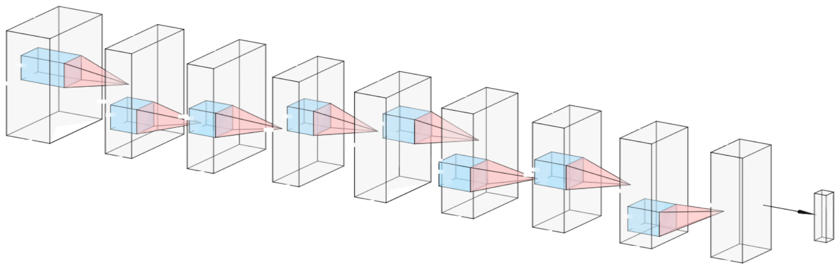


Figure 5.3.3 CNN Network Architecture with filters

6. System implementation

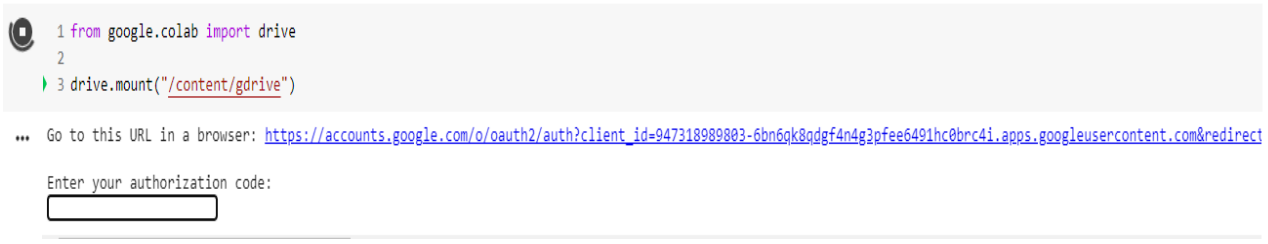
Considering the system requirements for this project we have a huge demand for the RAM and GPU because of implementing the Deep Learning Models which are CPU intensive and take a lot of computation, mainly made using Google Colab.
Colab is a free Jupyter notebook environment that runs entirely in the cloud. Most importantly, it does not require a setup and the notebooks that you create can be simultaneously edited by your team members - just the way you edit documents in Google Docs. Colab supports many popular machine learning libraries which can be easily loaded in your notebook.

6.1 Integrating Google Drive with Google Colab

The huge corpus is stored in Google Drive which could be fetched from the Google Colab environment as the files in the local machine can be fetched. So to connect Google Drive with Google Colab we can run this command.This command mounts the root directory within the Google colab runtime environment.

```
from google.colab import drive

drive.mount("/content/gdrive")
```

Executing the above command prompts for a user to specify the Authorization code to access the dataset by allowing permissions for Google Colab to access Google Drive.



```
from google.colab import drive

drive.mount("/content/gdrive")
```

```
Go to this URL in a browser: https://accounts.google.com/o/oauth2/auth?client_id=947318989803-6bn6qk8qdgf4n4g3pfee6491hc0brc4i.apps.googleusercontent.com&redirect

Enter your authorization code:
```

Fig 6.1.1 Authorization of the user

Once connected with the google drive we can access all the files and store the Network models and weights in the specified folders.

6.2 Fetching the Data from Google Drive

Since we have multiclass classification and multiple datasets are stored in various directories based on the severity of the Depressive class, below shown is the example of fetching the dataset for the mild class and concatenating the dataset into a single frame.
Based on the severity of depression it is classified as mild and severe and no depression.In the mild depressed tweets we have keywords like stress, anxiety, panic, trauma and in the severe depression class we have tweets which have the keyword depression, PTSD, double depression etc.

```
filepaths = [f for f in listdir("/content/gdrive/MyDrive/dataset_mild") if
f.endswith('.csv')]
filepaths
li = []

for filename in filepaths:
    print(filename)
    df = pd.read_csv('/content/gdrive/My Drive/dataset_mild/' +
str(filename),header=None,sep='\t',nrows = 1000)
    li.append(df)

frame_mild = pd.concat(li, axis=0, ignore_index=True)
```

The dataset contains various attributes apart from tweets or comments. It also contains timestamp, username, retweets and other tags extracted from TWINT. The data needs to be analysed as it contains many NaN values and broken characters which have to be fixed.

6.3 Preprocessing the Data

Text preprocessing is the method of cleaning the raw data to get data which is suitable for feeding it to the model and analysing it for other purposes.The data may contain broken characters,non ASCII characters, hashtags, emails, links, emojis etc.
The different types of preprocessing steps are

- Lower casing.
- Removal of Punctuations.
- Removal of Stopwords.
- Removal of Frequent words.

- Removal of Rare words.
- Stemming.
- Lemmatization.
- Removal of emojis.

```
def clean_tweets(tweets):
    cleaned_tweets = []
    for tweet in tweets:
        tweet = str(tweet)
        if re.match("(\w+:\/\/\S+)", tweet) == None and len(tweet) > 10:

            tweet = 
''.join(re.sub("(@[A-Za-z0-9]+)|(\#[A-Za-z0-9]+)|(<Emoji:.*>)|(pic\.twitter\
.com\/.*)", " ", tweet).split())

            tweet = ftfy.fix_text(tweet)

            tweet = expandContractions(tweet)

            tweet = ' '.join(re.sub("([^0-9A-Za-z \t])", " ",tweet).split())

            stop_words = set(stopwords.words('english'))
            word_tokens = nltk.word_tokenize(tweet)
            filtered_sentence = [w for w in word_tokens if not w in 
stop_words]
            tweet = ' '.join(filtered_sentence)

            tweet = PorterStemmer().stem(tweet)

            cleaned_tweets.append(tweet)

    return cleaned_tweets
```

All the above cleaning methods are applied in this function which takes a list of comments and cleans each one of them and returns the cleaned tweets.

6.4 Tokenizing the Data

The data is in the textual format or in its linguistic form which is not understandable by the computer so it has to be converted to a format which the computer understands so we convert it into numbers using Tokenizer.
Tokenization is a way of separating a piece of text into smaller units called tokens. Here, tokens can be either words, characters, or subwords. ... As each token is a word, it becomes an example of Word tokenization. Similarly, tokens can be either characters or subwords.
Basically the words are broken into tokens and total unique tokens are identified and we pad the matrix with 0 so that we have it of a length of 140 columns in the matrix.

```
tokenizer = Tokenizer(num_words=MAX_NB_WORDS)
tokenizer.fit_on_texts(X_m + X_s + X_r)

sequences_m = tokenizer.texts_to_sequences(X_m)
sequences_s = tokenizer.texts_to_sequences(X_s)
sequences_r = tokenizer.texts_to_sequences(X_r)

data_m = pad_sequences(sequences_m, maxlen=MAX_SEQUENCE_LENGTH)
data_s = pad_sequences(sequences_s, maxlen=MAX_SEQUENCE_LENGTH)
data_r = pad_sequences(sequences_r, maxlen=MAX_SEQUENCE_LENGTH)
```

6.5 Building the Embedding Matrix

While extraction of the features word2vec model is used so that each word in the comment can be given appropriate meaning in the form of representing in a vector of length 300, which is a shallow neural network created by Google. Making use of this we can create an embedding matrix which can be used to create an Embedding Layer in the Neural Network designed.

The idea behind word2vec is to represent words by a vector of real numbers of dimension d. Therefore the second matrix is the representation of those words. The i-th line of this matrix is the vector representation of the i-th word.It is an approach for representing words and documents. Word Embedding or Word Vector is a numeric vector input that represents a word in a lower-dimensional space. It allows words with similar meaning to have a similar representation. They can also approximate meaning. A word vector with 50 values can represent 50 unique features.

```
nb_words = min(MAX_NB_WORDS, len(word_index))

embedding_matrix = np.zeros((nb_words, EMBEDDING_DIM))

for (word, idx) in word_index.items():
    if word in word2vec.vocab and idx < MAX_NB_WORDS:
        embedding_matrix[idx] = word2vec.word_vec(word)
```

Goal of Word Embeddings

- To reduce dimensionality
- To use a word to predict the words around it
- Inter word semantics must be captured

6.6 Developing Machine Learning Models

Once the preprocessing is done and appropriate features are extracted from the text now we can train the model using various Machine Learning models.Since this is multiclass classification Naive Bayes, Random Forest, Decision Trees seems to work very well with the Text data.
Naive Bayes is a machine learning model that is used for large volumes of data, even if you are working with data that has millions of data records the recommended approach is Naive Bayes. It gives very good results when it comes to NLP tasks such as sentimental analysis.Feature vectors represent the frequencies with which certain events have been generated by a multinomial distribution. This is the event model typically used for document classification.

```
from sklearn.naive_bayes import GaussianNB,MultinomialNB
classifier = MultinomialNB()
classifier.fit(data_train, labels_train)

y_pred = classifier.predict(data_test)
confusion_matrix(y_pred,labels_test)
accuracy_score(labels_test,y_pred)
```

6.7 Developing Hybrid Deep Learning Models

Deep Learning models are applied on a huge corpus because it takes a lot of processing and yields only good results with large dataset.We have a huge amount of data gathered from Kaggle and Twitter which is suitable for applying Deep Learning Models.Basically the paper explores about Convolution Neural Network(CNN) cascading with Recurrent Neural Networks(RNN).

The models implemented are as follows

1. CNN
2. CNN + LSTM
3. CNN + Bidirectional LSTM
4. CNN + GRU
5. CNN + Bidirectional GRU

```
model = Sequential()
# Embedded layer
model.add(Embedding(len(embedding_matrix), EMBEDDING_DIM,
weights=[embedding_matrix], input_length=MAX_SEQUENCE_LENGTH,
trainable=False))

# The first two layers with 32 filters of window size 3x3
model.add(Conv1D(32, kernel_size=3, padding='same', activation='relu'))
model.add(Conv1D(32, kernel_size=3, activation='relu'))
model.add(MaxPooling1D(pool_size=2))
model.add(Dropout(0.25))

model.add(Conv1D(64, kernel_size=3, padding='same', activation='relu'))
model.add(Conv1D(64, kernel_size=3, activation='relu'))
model.add(MaxPooling1D(pool_size=2))
model.add(Dropout(0.25))

model.add(Conv1D(64, kernel_size=3, padding='same', activation='relu'))
model.add(Conv1D(64, kernel_size=3, activation='relu'))
model.add(MaxPooling1D(pool_size=2))
model.add(Dropout(0.25))

model.add(Flatten())
model.add(Dense(300, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(3, activation='softmax'))


model.compile(loss='sparse_categorical_crossentropy', optimizer='nadam',
metrics=['acc'])
print(model.summary())

early_stop = EarlyStopping(monitor='val_loss', patience=10)

hist = model.fit(data_train, labels_train, \
        validation_data=(data_val, labels_val), \
        epochs=10, batch_size=1024, shuffle=True, \
        callbacks=[early_stop])
```

6.7.1 Long Short Term Memory (LSTM)

The LSTM recurrent unit tries to “remember” all the past knowledge that the network has seen so far and to “forget” irrelevant data. This is done by introducing different activation function layers called “gates” for different purposes.

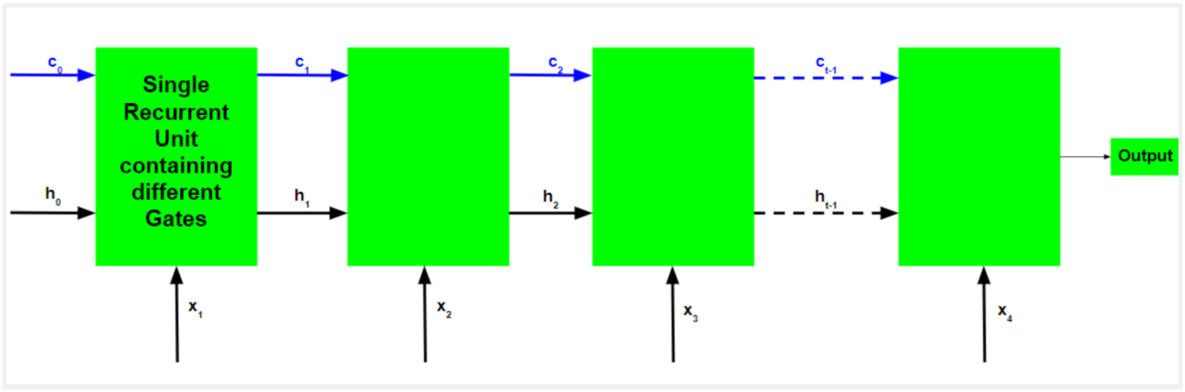


Fig 6.7.1.1 LSTM Network

Input Gate : Sigmoid Function

Output Gate : Sigmoid Function

Forget Gate : Sigmoid Function

Input Modulation Gate : Hyperbolic Tangent Function

Calculate the current internal cell state by first calculating the element-wise multiplication vector of the input gate and the input modulation gate, then calculate the element-wise multiplication vector of the forget gate and the previous internal cell state and then adding the two vectors.

$$Ct = i \circledcirc g + f \circledcirc Ct - 1$$

Calculate the current hidden state by first taking the element-wise hyperbolic tangent of the current internal cell state vector and then performing element wise multiplication with the output gate.

$$ht = O \circledcirc tan(Ct)$$

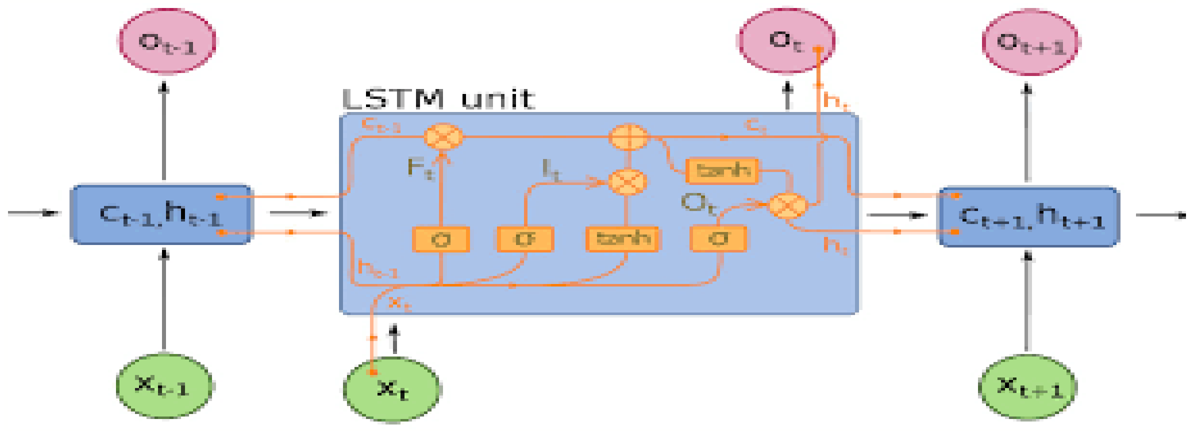


Figure 6.7.1.2 LSTM Cell

6.7.2 Gated Recurrent Units (GRU)

The basic work-flow of a Gated Recurrent Unit Network is similar to that of a basic Recurrent Neural Network when illustrated, the main difference between the two is in the internal working within each recurrent unit as Gated Recurrent Unit networks consist of gates which modulate the current input and the previous hidden state.

The process of calculating the Current Memory Gate is a little different. First, the Hadmard product of the Reset Gate and the previous hidden state vector is calculated. Then this vector is parameterized and then added to the parameterized current input vector.

$$ht = tanh(W \circledcirc xt + W \circledcirc (r \circledcirc ht - 1))$$

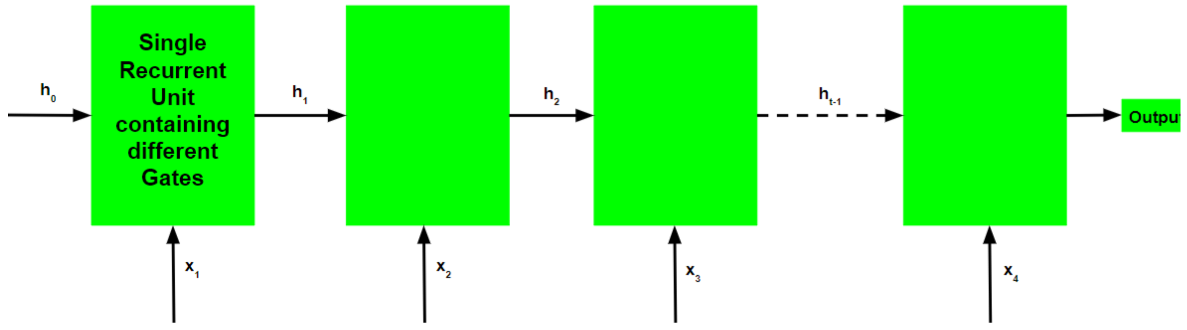


Fig 6.7.2.1 GRU Network

To calculate the current hidden state, first a vector of ones and the same dimensions as that of the input is defined. This vector will be called ones and mathematically be denoted by 1. First calculate the hadmard product of the update gate and the previous hidden state vector. Then generate a new vector by subtracting the update gate from ones and then calculate the hadmard product of the newly generated vector with the current memory gate. Finally add the two vectors to get the current hidden state vector.

$$ht = zt \circledcirc ht - 1 + (1 - zt) \circledcirc ht$$

7. System testing and result analysis

7.1 Result Analysis

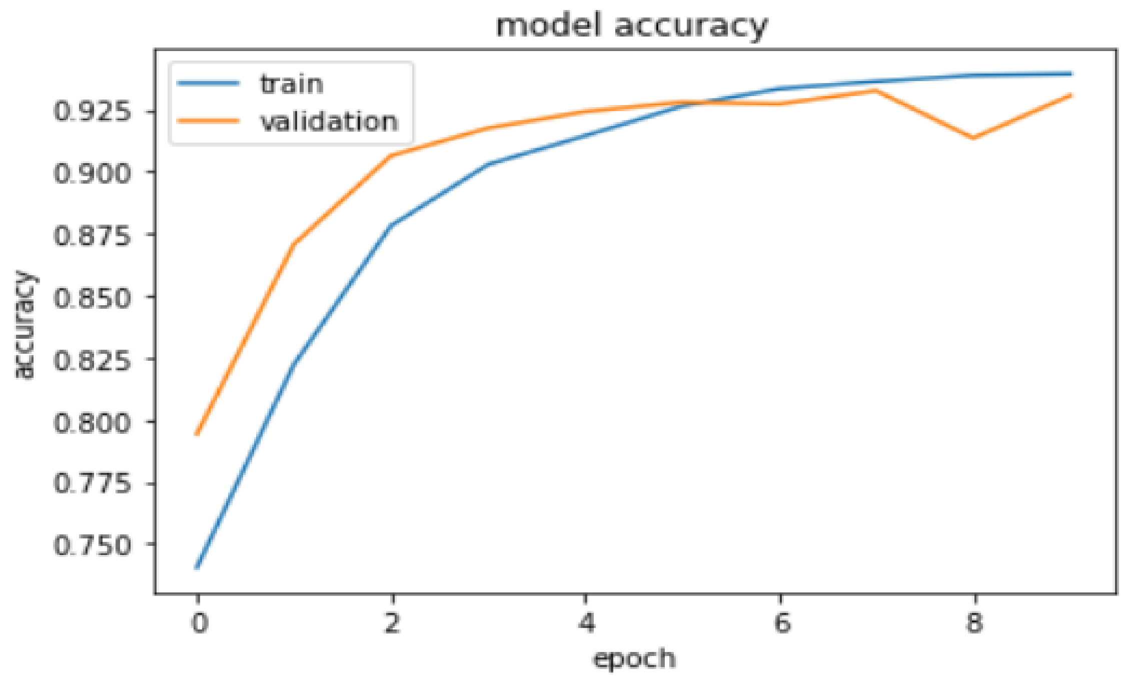


Fig 7.1 Epoch vs Accuracy

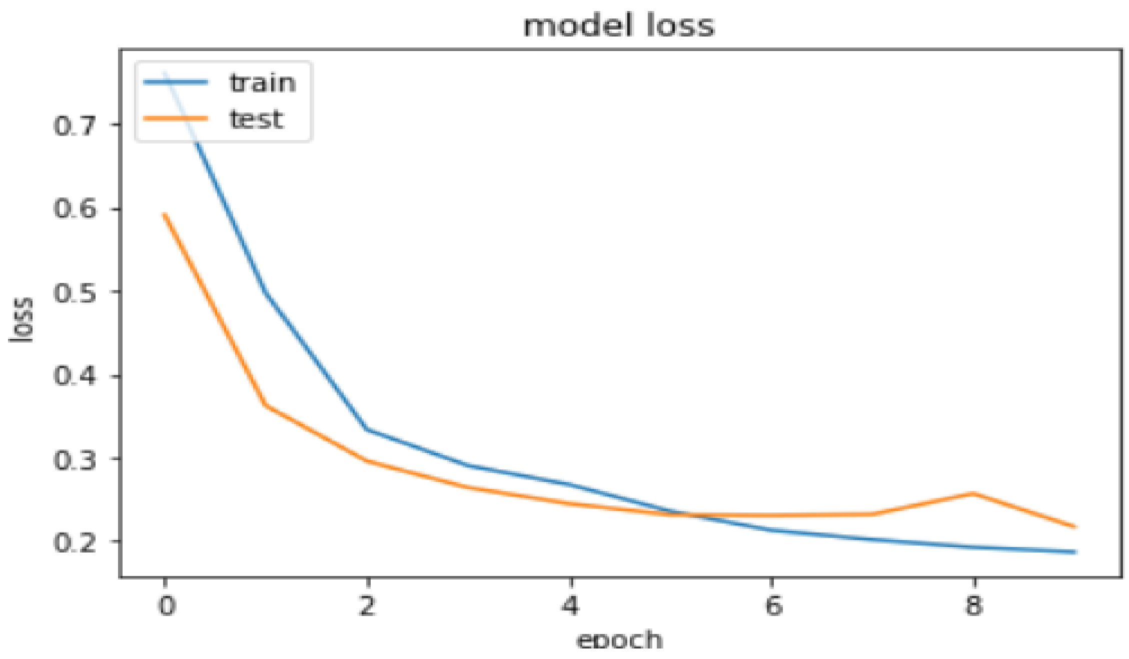
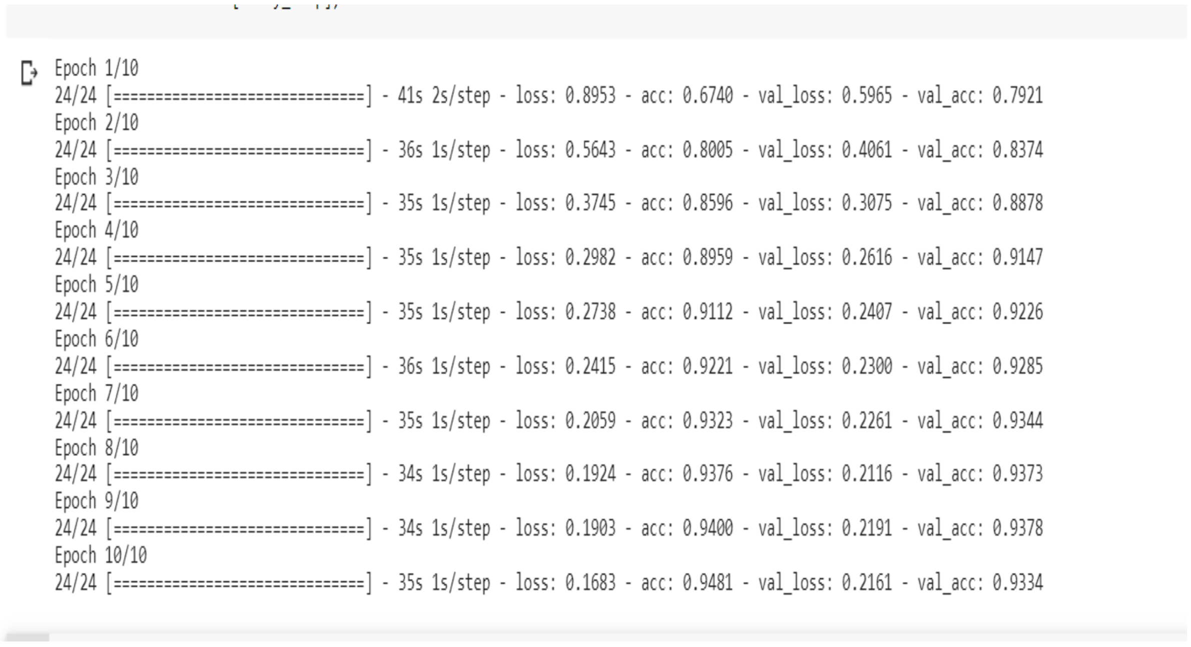


Fig 7.1 Epoch vs Loss

Every Machine Learning and Deep Learning Model works based on a cost function defined,the goal is to increase the performance of the model by decreasing the error which the model is making by falsely predicting or classifying the output, So as the epochs progress in the model it tries to decrease the error or loss of the cost function defined as a result of which the Network learns better and the weights and bias are updated accordingly.

As shown in Fig 7.1 the accuracy increases to a saturation point as the model progresses through the epochs and Fig 7.2 displays the decrease in loss for both training and testing data. The above figures are the results displayed for the variation of CNN models throughout its lifetime.



```
Epoch 1/10
24/24 [==============================] - 41s 2s/step - loss: 0.8953 - acc: 0.6740 - val_loss: 0.5965 - val_acc: 0.7921
Epoch 2/10
24/24 [==============================] - 36s 1s/step - loss: 0.5643 - acc: 0.8005 - val_loss: 0.4061 - val_acc: 0.8374
Epoch 3/10
24/24 [==============================] - 35s 1s/step - loss: 0.3745 - acc: 0.8596 - val_loss: 0.3075 - val_acc: 0.8878
Epoch 4/10
24/24 [==============================] - 35s 1s/step - loss: 0.2982 - acc: 0.8959 - val_loss: 0.2616 - val_acc: 0.9147
Epoch 5/10
24/24 [==============================] - 35s 1s/step - loss: 0.2738 - acc: 0.9112 - val_loss: 0.2407 - val_acc: 0.9226
Epoch 6/10
24/24 [==============================] - 36s 1s/step - loss: 0.2415 - acc: 0.9221 - val_loss: 0.2300 - val_acc: 0.9285
Epoch 7/10
24/24 [==============================] - 35s 1s/step - loss: 0.2059 - acc: 0.9323 - val_loss: 0.2261 - val_acc: 0.9344
Epoch 8/10
24/24 [==============================] - 34s 1s/step - loss: 0.1924 - acc: 0.9376 - val_loss: 0.2116 - val_acc: 0.9373
Epoch 9/10
24/24 [==============================] - 34s 1s/step - loss: 0.1903 - acc: 0.9400 - val_loss: 0.2191 - val_acc: 0.9378
Epoch 10/10
24/24 [==============================] - 35s 1s/step - loss: 0.1683 - acc: 0.9481 - val_loss: 0.2161 - val_acc: 0.9334
```

Fig 7.3 Output of CNN model

The above Fig 7.3 displays how the loss, accuracy, validation loss, validation accuracy varies as the model learns and adjusts its weights and biases. It reaches a saturation point where the model is unable to decrease the loss because of the characteristic of the data or the features extracted.Sometimes overfitting of the model shoots up the accuracy but we have to also consider the validation accuracy which is a data hidden from the model and it is only shown once the model is trained, if validation accuracy is on par with the actual accuracy then model is working fine. There are various other features which contribute to the overfitting or underfitting of the model like skewed data or the learning rate might be some of the reasons.

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd4ff497050>
```

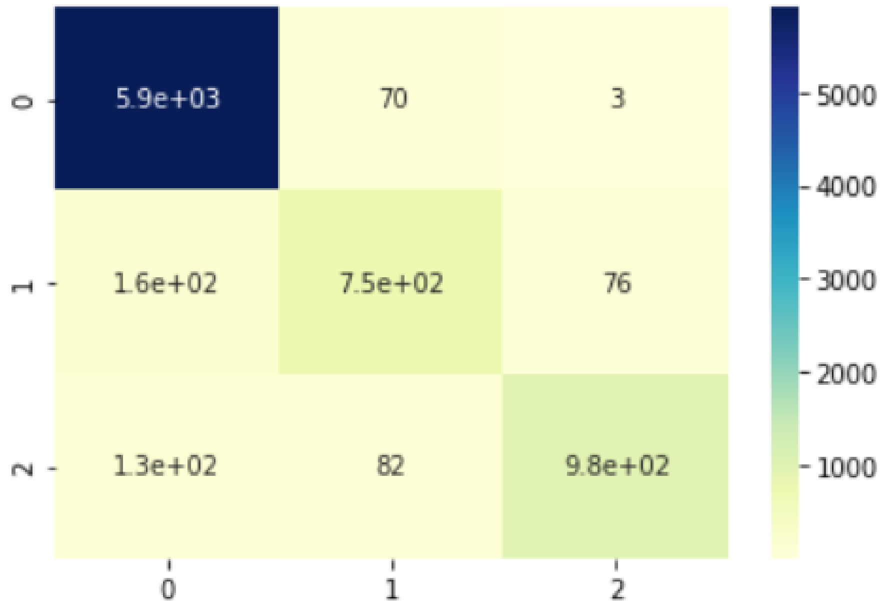
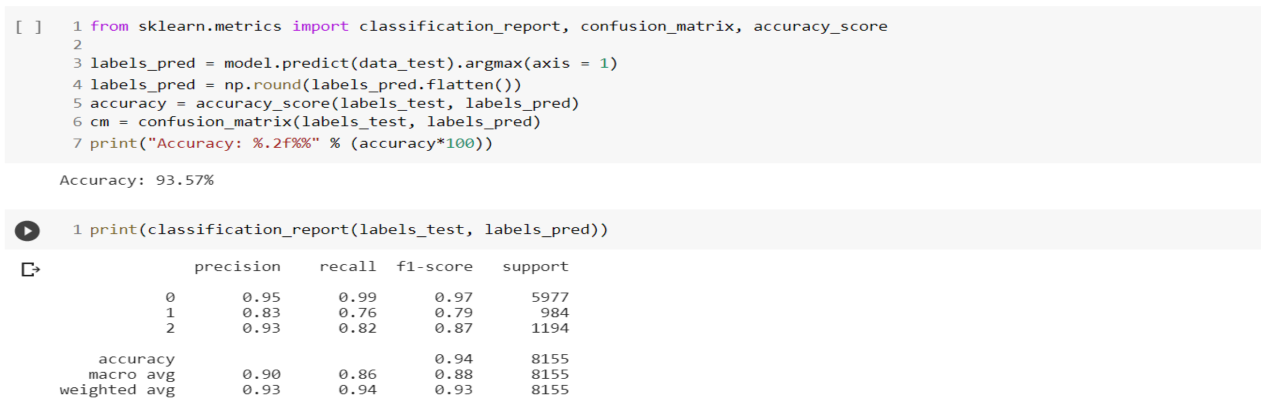


Fig 7.3 Confusion Matrix based heatmap for Random Forest

A Confusion matrix is an N x N matrix used for evaluating the performance of a classification model, where N is the number of target classes. The matrix compares the actual target values with those predicted by the machine learning model. It gives us an edge to look for the values which are falsely classified and improve the model so as to decrease the loss. The above is a confusion matrix for the Random Forest Classifier, the diagonal values show the values which are correctly classified and the rest are incorrectly classified.
Other accuracy Metrics include Precision, Recall, F1-score these are considered to be better metrics for evaluation as these would expose the skewness of the data and give low accuracies if the model is overfitting or underfitting.



```
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score

labels_pred = model.predict(data_test).argmax(axis = 1)
labels_pred = np.round(labels_pred.flatten())
accuracy = accuracy_score(labels_test, labels_pred)
cm = confusion_matrix(labels_test, labels_pred)
print("Accuracy: %.2f%%" % (accuracy*100))
```

```
Accuracy: 93.57%
```

```
print(classification_report(labels_test, labels_pred))
```

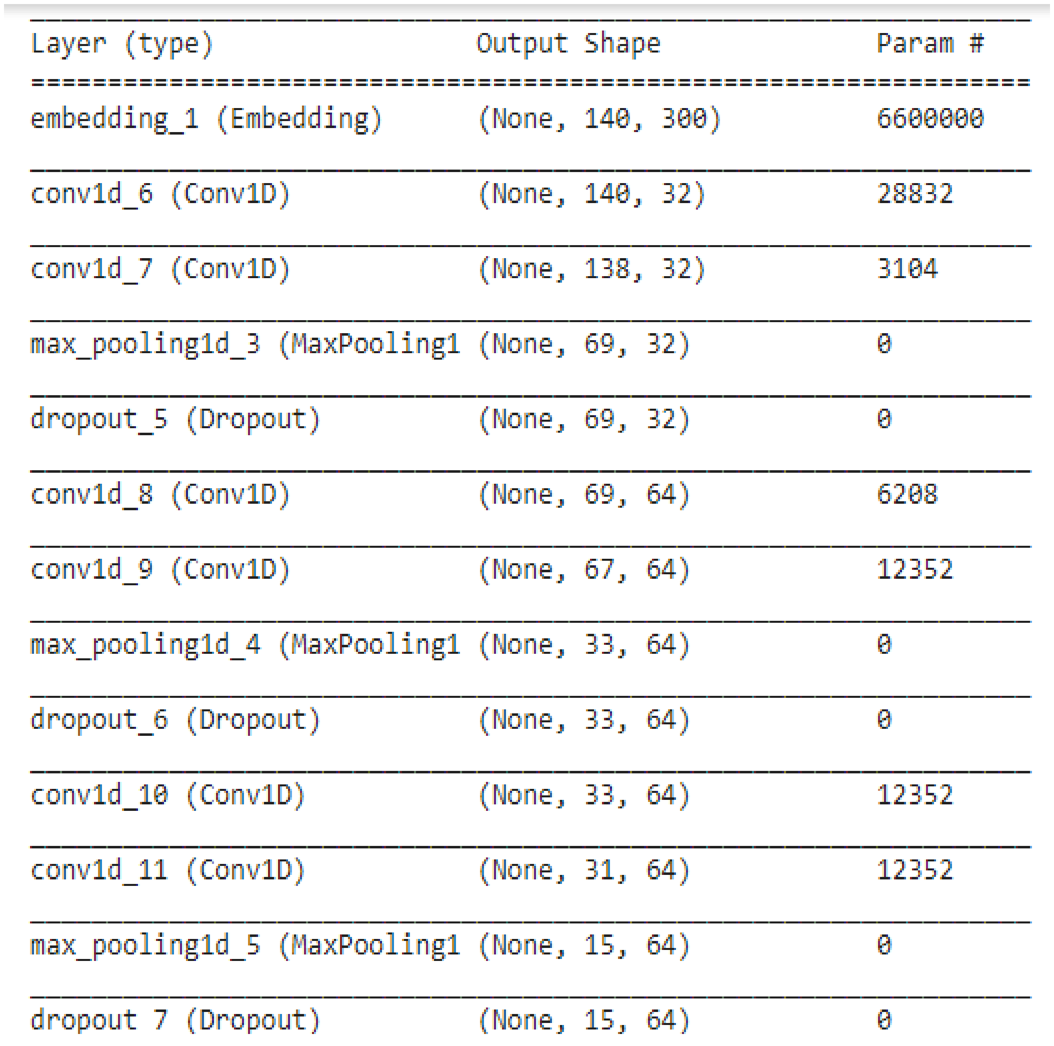
```
              precision    recall  f1-score   support

           0       0.95      0.99      0.97      5977
           1       0.83      0.76      0.79       984
           2       0.93      0.82      0.87      1194

    accuracy                           0.94      8155
   macro avg       0.90      0.86      0.88      8155
weighted avg       0.93      0.94      0.93      8155
```

Fig 7.5 Accuracy, Recall, Precision and F1-score

7.2 Testing

The system needs to be checked for its intended behavior and direction of progress at each development stage to avoid duplication of efforts, time and cost overruns, and to assure completion of the system within stipulated time.To assure completion of the system within stipulated time.There are various types of testing based on the use cases designed.

7.2.1 Unit Testing



```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_1 (Embedding)      (None, 140, 300)          6600000
_________________________________________________________________
conv1d_6 (Conv1D)            (None, 140, 32)           28832
_________________________________________________________________
conv1d_7 (Conv1D)            (None, 138, 32)           3104
_________________________________________________________________
max_pooling1d_3 (MaxPooling1 (None, 69, 32)            0
_________________________________________________________________
dropout_5 (Dropout)          (None, 69, 32)            0
_________________________________________________________________
conv1d_8 (Conv1D)            (None, 69, 64)            6208
_________________________________________________________________
conv1d_9 (Conv1D)            (None, 67, 64)            12352
_________________________________________________________________
max_pooling1d_4 (MaxPooling1 (None, 33, 64)            0
_________________________________________________________________
dropout_6 (Dropout)          (None, 33, 64)            0
_________________________________________________________________
conv1d_10 (Conv1D)           (None, 33, 64)            12352
_________________________________________________________________
conv1d_11 (Conv1D)           (None, 31, 64)            12352
_________________________________________________________________
max_pooling1d_5 (MaxPooling1 (None, 15, 64)            0
_________________________________________________________________
dropout 7 (Dropout)          (None, 15, 64)            0
```

Figure 7.2.1.1 Checking Compatibility of Shapes

Applying Deep Learning models, we have to look for the compatibility of the tensors defined as the Internal operations being multiplication, dot product of the Matrix if found not compatible then the required operations fail. The Figure 7.2.1.1 shows the resultant tensor of output of each layer which guarantees the compatibility.

7.2.2 System Testing

System testing is testing conducted on a complete integrated system to evaluate the system's compliance with its specified requirements. System testing takes, as its input, all of the integrated components that have passed integration testing.

With Deep learning model into consideration and applying NLP word2vec model which consumes huge amount of memory and high level of computation. With basic hardware requirements specified for the project when multiple Deep learning models applied on top of NLP framework and huge dataset it accumulates large amount of memory in the RAM, system testing has to make sure ample amount of memory, resources, GPU is defined so that when multiple frameworks are into operation the system doesn’t crash.

8. Conclusion and future work

Due to Covid-19, the number of people using social media has increased and we can see all the reasons for depression. By using this data we can get more accurate results than previous models, and with the use of several Machine Learning Techniques such as Naïve Bayes,SVM,KNN we can improve the results.The hybrid Deep Learning model has also proven to be of considerable significance in terms of accuracy.The scope of the product is to detect depression using various Text mining techniques and also it can be used for medical diagnosis consulting the experts in the field of medical sciences. The scope of this product can also be extended in getting to know the state of mind of an individual.

The models applied here are Machine Learning and Hybrid Deep learning models but more advanced and CPU intensive algorithms can be deployed, before that we can extract the features from the text using more sophisticated methods available like GloVe, TF-IDF and others and look out for the best methods to extract more meaningful insights from the text.

With the explosion of data it is available everywhere and people seem to express their part of the story in LinkedIn, Reddit, Facebook and many more.Look out for the API's available to get the user comments from these social media websites to work on the same and a quality research can be conducted with variety of data in hand. With the growing technology we can classify Sentiments and facts before hand before giving input to the system.

The research of the project can not only be extended to various languages used in the social media sites, but also we can try to identify the use of languages like Hindi, Kannada, Telugu written in English and can extract the information. We can consider the emojis and hashtags as a major attribute or feature in the upcoming projects.

Appendix A : Project team details

Akshay Suryanarayan Hegde
01JST17CS014

Ganesh S
01JST17CS054

Aniket Kharad
01JST17CS018

Manzoor Ahmed
01JST17CS084

Appendix B : COs,POs and PSOs mapping for the project work

10.1 Course Outcomes

CO1: Formulate the problem definition, conduct literature review and apply requirements analysis.
CO2: Develop and implement algorithms for solving the problem formulated.
CO3: Comprehend, present and defend the results of exhaustive testing and explain the major findings.

10.2 Program Outcomes

PO1: Apply knowledge of computing, mathematics, science, and founda- tional engineering concepts to solve the computer engineering prob- lems.
PO2: Identify, formulate and analyze complex engineering problems
PO3: Plan, implement and evaluate a computer-based system to meet de- sired societal needs such as economic, environmental, political, health- care and safety within realistic constraints.
PO4: Incorporate research methods to design and conduct experiments to investigate real time problems, to analyze, interpret and provide feasible conclusions.
PO5: Propose innovative ideas and solutions using modern tools.
PO6: Apply computing knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to pro- fessional engineering practice.
PO7: Analyze the local and global impact of computing on individuals and organizations for sustainable development.
PO8: Adopt ethical principles and uphold the responsibilities and norms of computer engineering practice.
PO9: Work effectively as an individual and as a member or leader in diverse teams and in multidisciplinary domains.
PO10: : Effectively communicate and comprehend.
PO11: Demonstrate and apply engineering knowledge and management prin- ciples to manage projects in multidisciplinary environments.
PO12: Recognize contemporary issues and adapt to technological changes for lifelong learning.

10.3 Program Specific Outcomes

PSO1: Problem Solving Skills: Ability to apply standard practices and mathematical methodologies to solve computational tasks, model real world problems in the areas of

database systems, system software, web technologies and Networking solutions with an appropriate knowledge of Data structures and Algorithms.
PSO2: Knowledge of Computer Systems: An understanding of the structure and working of the computer systems with performance study of various computing architectures.
PSO3: Successful Career and Entrepreneurship: The ability to get acquain- tance with the state of the art software technologies leading to en- trepreneurship and higher studies.
PSO3: Computing and Research Ability: Ability to use knowledge in var- ious domains to identify research gaps and to provide solutions to new ideas leading to innovations.

10.4 Justification for the mapping

The first CO is related to problem definition, literature survey and requirement analysis. Planning the project such that it meets the needs of society, by considering all the constraints is very relevant for this. Investigating real time problems and incorporating our findings in literature surveys plays an important role as well. Understanding the constraints of the environment in which the system will be used plays a crucial role in deciding the require- ments and using the latest technology to make our implementation better is of high relevance. The second CO, design and implementation highly depends on the way we apply already acquired knowledge about computing, mathematics, etc., the innovation we bring in to our implementation, make our implementation adaptable to technological changes that might happen in future and the way

We look at the problem and apply our knowledge. The ability to analyze global and local impact of the system, ability to uphold the ethical principles of engineering practices, the way in which we communicate and comprehend the concepts, the way in which the project is handled in multidisciplinary environments hold great relevance in defending our work and explaining major findings during our project.
Note:
1. Scale 1 Low relevance
2. Scale 2 Medium relevance
3. Scale 3 High relevance

The following subjects helped us in developing our project:

To formulate the problem definition, conduct literature review and apply requirements analysis that we gained from Software Engineering and System Software.

To Develop and implement algorithms for solving the problem formu- lated , we gained the knowledge from subjects such as Neural Networks, Object Oriented Programming, Machine Learning.

To Comprehend, present and defend the results of exhaustive testing and explain the major findings, we gained the knowledge from Engineering Mathematics, Environments like Google colab, Githhub.

Appendix C : Publication Details

This project was published as a paper titled DEEP LEARNING BASED DETECTION OF DEPRESSION in the Volume 11 Issue - VI 2021 of Pramana Research Journal(PRJ).
DOI:16.10089.PRJ.2021.V11I6.20.9308

References

[1] Eichstaedt C. J. Facebook language predicts depression in medical records, Psychological And Cognitive Science, vol. 115, No. 44, pp: 11203– 11208, October 2018.

[2] Islam R, Kabir A, Wang H and Ulhaq A, Depression detection from social network data using machine learning Techniques, Islam et al. Health Inf Sci Syst, vol. 6, No. 8,pp:1-12,2019.

[3] Jonathon C and et. al, Facebook language predicts depression in medical records, PNAS ,vol.115 ,No.44,pp: 11203-11208;October,2018.

[4] Aldarwish M.M., Hafiz F. A. "Predicting Depression Levels Using Social Media",King SaudUniversity for Health Science National Guard, 2017 IEEE 13th International Symposium on Autonomous Decentralized Systems.

[5] Sadeque F, Xu D. and Bethard S."Measuring the Latency of Depression Detection in Social Media",WSDM '18: Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining, pp.: 495-503, October 2018.

[6] Reece G. A. and et. al., Forecasting the onset and course of mental illness with Twitter data, Scientific Reports | 7: 13006.

[7] Tao X., Zhou X., Zhang J and Yong J. Sentiment Analysis for Depression Detection on Social Networks, J. Li et al. (Eds.): ADMA 2016, LNAI 10086, pp. 807–810, 2016.

[8] https://en.wikipedia.org/wiki/Random_forest

[9] Dey Chowdhury M, Counts S. and Horvitz E, Social Media as a Measurement Tool of Depression in Populations.

[10] https://www.kaggle.com/sudalairajkumar/getting-started-with-text-preprocessing

[11] https://en.wikipedia.org

[12] Samarth P, Tejus R S, Nihal Pradhan for Comparison of Depression Detection Algorithms And Analysing Differences In Stress Level During and Before The COVID Pandemic

[13] Shervin Minaee, Nal Kalchbrenner, Erik Cambria, Narjes Nikzad, Meysam Chenaghlu, Jianfeng Gao, for Deep Learning Based Text Classification: A Comprehensive Review

[14] DEY, SHARMISTHA, et al. "DEPRESSION DETECTION USING INTELLIGENT ALGORITHMS FROM SOCIAL MEDIA CONTEXT-STATE OF THE ART, TRENDS AND FUTURE ROADMAP."

[15] Deshpande, Mandar & Rao, Vignesh. (2017). Depression detection using emotion artificial intelligence. 858-862. 10.1109/ISS1.2017.8389299.

[16] Guo, Hao, et al. "Resting-state functional connectivity abnormalities in first-onset unmedicated depression." Neural regeneration research 9.2 (2014): 153.

[17] Guntuku. C. S. and et. al," Detecting depression and mental illness on social media: an integrative review", Current Opinion in Behavioral Sciences Vol. 2017, No. 18pp:43–49, 2017.

[18] Tadesse M. M, Lin H, Xu B., Yang L (2019). Detection of Depression-Related Posts in Reddit Social Media Forum, IEEE Access, Vol. 7, pp.:44883-44893, 2019.

[19] De Choudhury M, Counts S, Horvitz EJ, Hoff A: "Characterizing and predicting postpartum depression from shared Facebook data". In Proceedings of the 17th ACM Conference on Computer Supported Cooperative Work & Social Computing. ACM; 014:626-638.

[20] Kate Daine, Keith Hawton, Vinod Singaravelu, Anne Stewart, Sue Simkin, and Paul Montgomery.The power of the web: A systematic review of studies of the influence of the internet on self-harm and suicide in young people. Plos One, 8(10):e77555, 2013.

[21] Lin H & et. al., User-level psychological stress detection from social media using deep neural network. In ACM International Conference on Multimedia, pp.: 507–516, 2014.

[22] Lin H., Jia J.,Nie L. ,Shen L. , and Chua T. S.. "What does social media say about your stress?", In International Joint Conference on Artificial Intelligence,3775–3781, 2016.

[23] Lin H. & et. al. Detecting stress based on social interactions in social networks., IEEE Transactions on Knowledge & Data Engineering, vol. 29, No. 9,pp:1820–1833, 2017.

[24] Park M., David W. & Cha M. “Perception differences between the depressed and non-depressed users in twitter”. In Proceedings of the International Conference on Weblogs and Social Media, pages 476–485, 2013.

[25] X Liu and et. Al, Proactive Suicide Prevention Online (PSPO): Machine Identification and Crisis Management for Chinese Social Media Users With Suicidal Thoughts and Behaviors, J Med Internet Res., vol.21, No..5: e11705. 2019.

[26] Murthy RS. National Mental Health Survey of India 2015-2016. Indian J Psychiatry. 2017 Jan-Mar;vol. 59, No.1, pp.: 21-26.,March, 2017.

[27] S. Biswas, I. Sarkar, P. Das, R. Bose, S. Roy , Examining the Effects of Pandemics on Stock Market Trends through Sentiment Analysis, , Journal of Xidian University 14 (6), 1163-1176.

[28] K. R. Dey, D. Sarddar, I. Sarkar, R. Bose, S. Roy, A Literature Survey on Sentiment Analysis Techniques involving Social Media and Online Platforms, International Journal of Scientific & Technology Research, 9 (5), 166-173.

[29] K. Chanda, P. Bhattacharjee, S. Roy, S. Biswas, Intelligent Data Prognosis of Recurrent of Depression in Medical Diagnosis, International Conference on Reliability, Infocom Technologies And Optimization (ICRITO 2020), pp.: 1–5.

[30] D. Sarddar, K.R. Dey, R. Bose, S. Roy, Topic Modeling as a Tool to Gauge Political Sentiments from Twitter Feeds, International Journal of Natural Computing Research (IJNCR) 9 (2), 1 – 22.