ANALYZING TEXTUAL DATA FOR EARLY DETECTION OF SUICIDE AND DEPRESSION

A PROJECT REPORT

Submitted *by*

SANKET SAYAL (19BCS1153)

DHEERAJ KARWASRA (19BCS1177)

SAHIL PATIL (19BCS2267)

in partial fulfilment for the award of the degree of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE & ENGINEERING

Chandigarh University

JUNE 2023

BONAFIDE CERTIFICATE

Certified that this project report "Analyzing textual data for early detection of suicide and depression" is the Bonafide work of "Dheeraj Karwasra (19BCS1177), Sanket Sayal (19BCS1153) and Sahil Patil (19BCS2267)" who carried out the project work under my/our supervision.

SIGNATURE
Dr. Navpreet Kaur Walia
Head of the Department
Computer Science & Engineering

SIGNATURE
Er. Khushwant Virdi
Assistant Professor
Supervisor
Computer Science & Engineering

Submitted for the project viva-voice examination held on

INTERNAL EXAMINER

Er. Khushwant Virdi

EXTERNAL EXAMINER

ACKNOWLEDGEMENTS

We would like to express our deep gratitude to our project guide Er. Khushwant Virdi, Department of Computer Science and Engineering, for her guidance with unsurpassed knowledge and immense encouragement. We are grateful to our Head of Department for providing us with the required facilities for the completion of the project work.

We express our thanks to Project Coordinator, for her continuous support and encouragement. We thank all teaching faculty of Department of computer science and engineering, whose suggestions during reviews helped us in accomplishment of our project.

We would like to thank our parents, friends, and classmates for their encouragement throughout our project period. At last, but not the least, we thank everyone for supporting us directly or indirectly in completing this project successfully.

Sanket Sayal (19BCS1153)

Dheeraj Karwasra (19BCS1177)

Sahil Patil (19BCS2267)

ABSTRACT

Depression is a mental illness which can impair the performance of a person and can lead to suicide. There are Traditional methods used by medical experts to diagnose a person for depression or suicidal tendencies. Deep Learning was used to understand how to predict that a person is diagnosed with depression and their accuracy. In this work, we have used a dataset from twitter which is a collection of tweets because of its prompt idea exchange, flexibility in expressing emotions, and wide range of content. people's mental states, according to a quick perusal of the articles. Deep Learning and Machine Learning methods are proposed in this research

सार

डिप्रेशन एक मानसिक बीमारी है जो किसी व्यक्ति के प्रदर्शन को खराब कर सकती है और आत्महत्या का कारण बन सकती है। किसी व्यक्ति के अवसाद या आत्महत्या की प्रवृत्ति का निदान करने के लिए चिकित्सा विशेषज्ञों द्वारा पारंपरिक तरीकों का उपयोग किया जाता है। डीप लर्निंग का उपयोग यह समझने के लिए किया गया था कि कैसे भविष्यवाणी की जाए कि किसी व्यक्ति को अवसाद और उनकी सटीकता का निदान किया गया है। इस काम में, हमने ट्विटर से एक डेटासेट का उपयोग किया है जो ट्वीट्स का एक संग्रह है क्योंकि इसके त्वरित विचारों का आदान-प्रदान, भावनाओं को व्यक्त करने में लचीलापन और सामग्री की विस्तृत श्रृंखला है। लोगों की मानसिक स्थिति, लेखों के त्वरित अवलोकन के अनुसार। इस शोध में डीप लर्निंग और मशीन लर्निंग के तरीके प्रस्तावित हैं

TABLE OF CONTENT

LIST OF FIGURES i

ABBREVIATIONS ii

CHAPTER 1 INTRODUCTION 1

1.1 Relevant Contemporary Issues 1
1.2 Identification of Problem 3
1.3 Identification of Tasks 3
1.4 Timeline 4
1.5 Organization of the Report 5

CHAPTER 2 LITERATURE SURVEY 6

2.1 Timeline of the Reported Problem 6
2.1.2 Deep Learning Features with NLP 6
2.1.3 Datasets and Evaluation 7
2.2 Proposed Solution 7
2.2.1 Support Vector Machine 7
2.2.2 TF-IDF Vectorizer 8
2.3 Bibliometric Analysis 8
2.3.1 DEEP LEARNING BASED METHODS 8
2.3.2 Limitations 9
2.4 Review Summary 10
2.5 Problem Definition 11
2.6 Goals and Objectives 12

CHAPTER 3 DESIGN FLOW 13

3.1 Evaluation & Selection of Specifications/Features 13
3.2 Design Constraints 14
3.3 Analysis and Feature Finalization 15
3.4 Design Flow 16
3.4.1 LSTM Model 16

3.5 Design Selection 17
3.5.1 TF-IDF 18
3.5.2 Long Short Term Memory 18
3.5.3 Kaggle Dataset 19
3.5.4 Model Advantages 20
3.6 Methodology 21
3.6.1 Model Overview 21
3.6.2 Recurrent Neural Network 21
3.6.4 Long Short Term Memory 22
CHAPTER 4 RESULTS ANALYSIS AND VALIDATION 24
4.1 Implementation 24
4.1.1 Pre-Requisites 24
4.1.2 Project File Structure 24
4.2 Building Project 25
4.2.1 Data Cleaning 25
4.2.2 Data preprocessing 26
4.3 Loading Dataset for Training the Model 27
4.3.1 Tokenizing the Vocabulary 28
4.3.2 Defining the Model 28
4.4 Training and Testing the model 29
CHAPTER 5 CONCLUSION AND FUTURE WORK 30
5.1 Conclusion 30
5.2 Future Scope 31
REFERENCES 32
APPENDIX 1 34
ANNEXURE-1 43

LIST OF FIGURES

Figure 1.1 People experience different combinations of symptoms 2
Figure 1.2 Project Timeline 4
Figure 3.1 RNN 22
Figure 3.2 LSTM Inner Model 23
Figure 4.1 File containing the data 25
Figure 4.2 Data loaded into the model 26
Figure 4.3 Data after pre-processing 27
Figure 4.4 Target of the model 28
Figure 4.5 Graph of the result of training data 29

ABBREVIATIONS

ML	Machine Learning
AI	Artificial Intelligence
NN	Neural Networks
LSTM	Long Short Term Memory
RNN	Recurrent Neural Network

CHAPTER 1
INTRODUCTION

1.1 Relevant Contemporary Issues

There are several contemporary issues in suicide and depression detection research that are currently being explored by researchers and mental health professionals.

Use of technology: With the advancements in technology, researchers are exploring the use of various tools such as artificial intelligence (AI), machine learning, and natural language processing (NLP) to detect signs of depression and suicidal ideation. For instance, researchers are developing chatbots and mobile applications that can detect changes in a person's mood and alert mental health professionals or loved ones in case of any potential risks.

Stigma and bias: There is still a lot of stigma surrounding mental health, particularly around issues related to suicide and depression. This can make it challenging for individuals to seek help, and for researchers to gather accurate data. Researchers are actively working to address this issue by creating more inclusive and culturally sensitive approaches to suicide and depression detection.

Early intervention: Early detection of depression and suicidal ideation is crucial in preventing these issues from escalating. Researchers are working on developing new screening tools that can detect these issues at an early stage and provide timely intervention.

Personalized treatment: Depression and suicidal ideation affect individuals differently, and there is no one-size-fits-all approach to treatment. Researchers are exploring the use of personalized treatment approaches that take into account an individual's unique experiences, needs, and preferences.

Co-occurring disorders: Depression and suicide risk often co-occur with other mental health disorders such as anxiety, substance abuse, and PTSD. Researchers are exploring how to effectively detect and treat these co-occurring disorders to improve overall mental health outcomes.

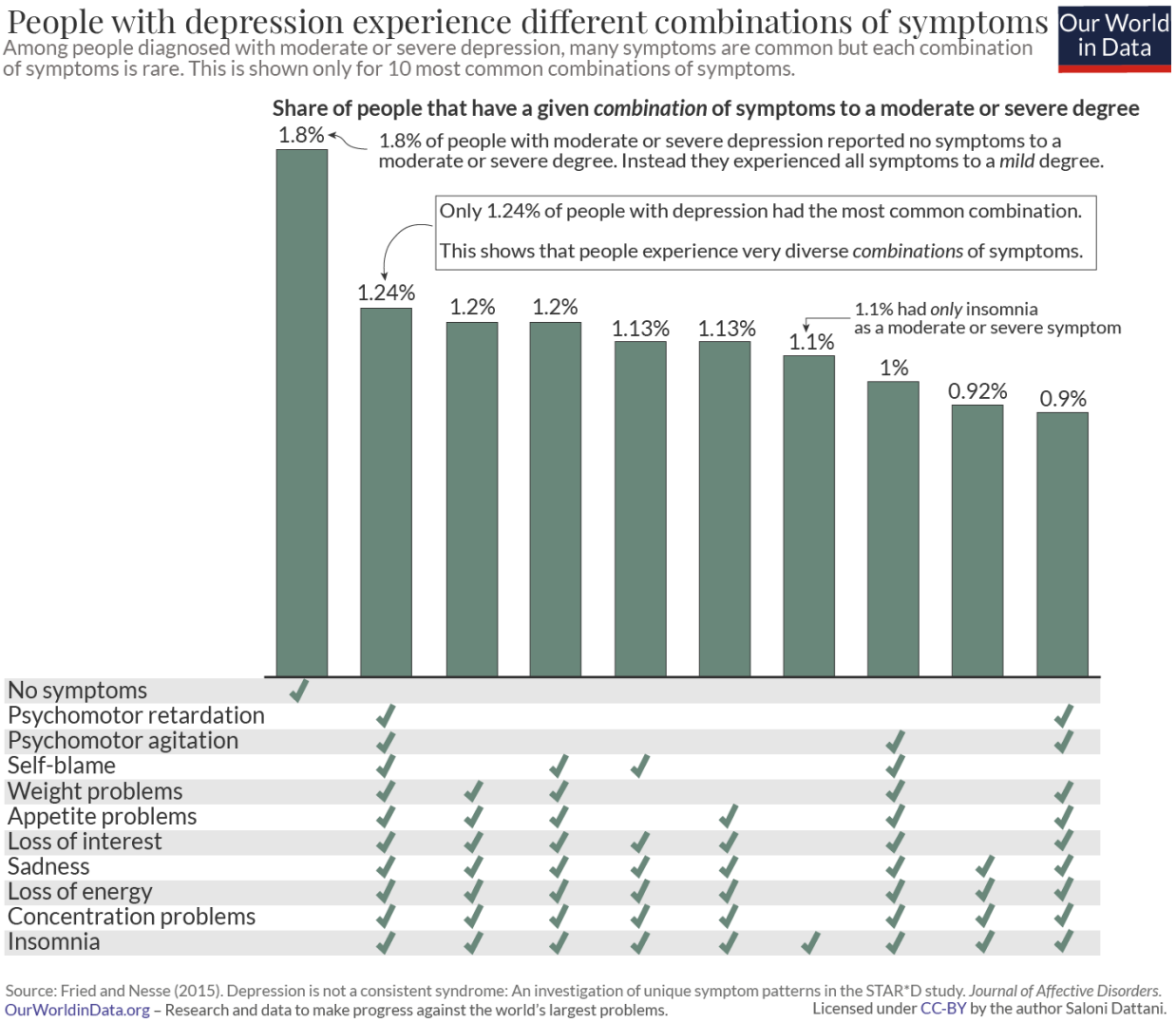

Figure 1.1 People experience different combinations of symptoms

The whole model of suicide and depression detection has a potential to have large impact. According to the World Health Organization (WHO), suicide is a major public health issue, with over 700,000 people dying from suicide each year globally. Depression is also a leading cause of disability worldwide, affecting over 264 million people. WHO emphasizes the need for a comprehensive and collaborative approach to suicide and depression detection research, which takes into account the complex social, cultural, and environmental factors that contribute to these issues.

1.2 Identification of Problem

Though numerous researches and deep learning models have been already put into play to detect suicide and depression, One of the key challenges in suicide and depression detection using machine learning (ML) and deep learning (DL) approaches is the identification of relevant features or patterns in the data that can accurately predict depression and suicidal ideation. There are several factors that can contribute to depression and suicidal ideation, including social, environmental, and genetic factors. ML and DL approaches can be used to analyze large datasets and identify patterns and features that are associated with depression and suicidal ideation. However, identifying the most relevant features can be challenging, as there may be many variables to consider, and the relationships between these variables may be complex and nonlinear. Another problem in using ML and DL approaches for suicide and depression detection is the need for high-quality, diverse datasets. In order to train accurate models, it is important to have access to large, diverse datasets that capture the complexity and diversity of depression and suicidal ideation. However, collecting such data can be challenging due to ethical and privacy concerns, as well as the stigma surrounding mental health issues. There is a risk of bias in ML and DL algorithms if the data used to train the model is not representative of the population. For example, if the dataset used to train the model only includes individuals from a certain demographic or cultural background, the model may not accurately generalize to other populations.

1.3 Identification of Tasks

A lot of tasks need to be accomplish and relevant skills required to successfully build this project:

1. Dataset fetching from an online website
2. Data Pre-processing or cleaning
3. Building a model – implement using a specific programming language.
4. Testing

Skills required to complete this project are stated:

1. System design – knowledge of UML diagram, system design concepts
2. Data collection – through online websites like kaggle

3. Data Pre-processing or cleaning – knowledge of python, or any contemporary modern day programming language
4. Building a model – knowledge about a specific programming language preferably Python, expertise in various machine learning techniques.
5. Testing – Knowledge about Unit testing, Black box testing or any special test package in Python.
6. Report Generation – Good command over English language.

1.4 Timeline

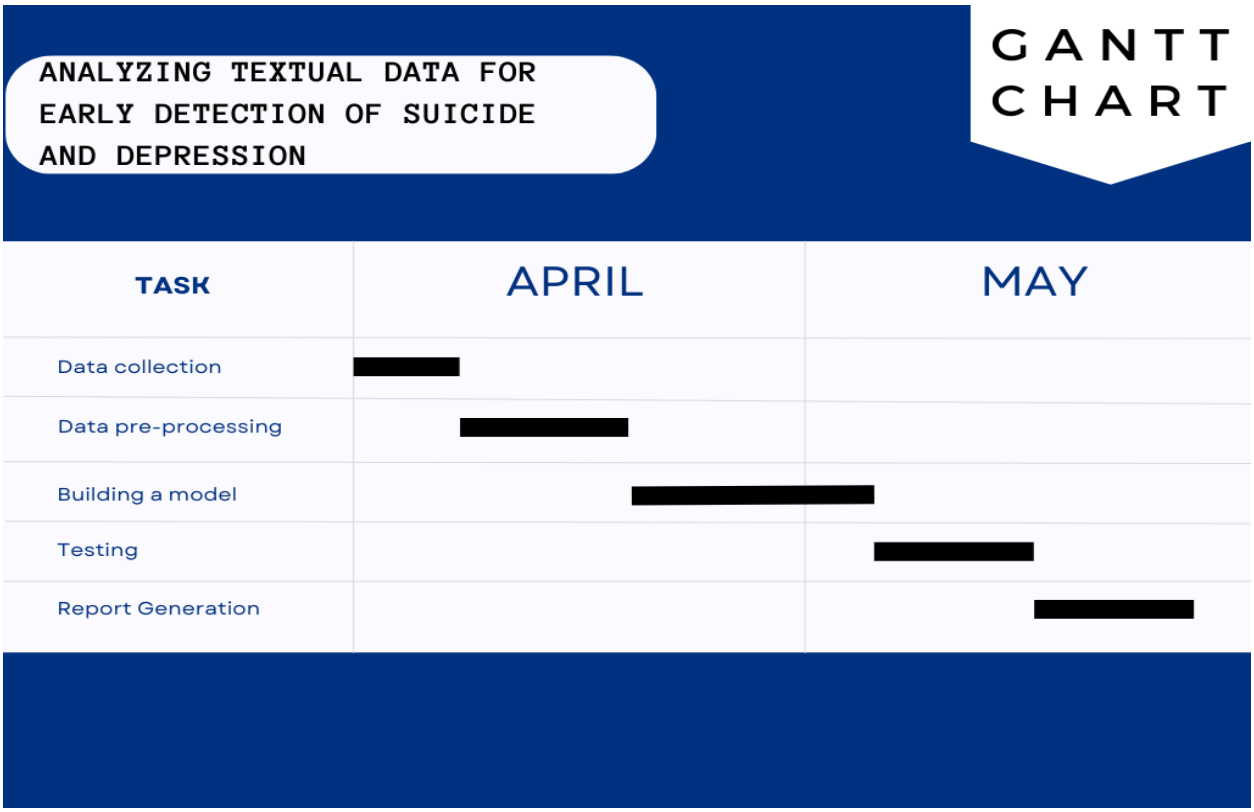

Figure 1.2 Project Timeline

1.5 Organization of the Report

The following chapters will give an elaborate description of the project.

- Chapter 2, of this report describes the various literature surveyed, hence give an idea of the recent trends and development in the field. This chapter also propose solutions to solve the problems identified in chapter 1, goals and objectives this project wish to achieve.
- Chapter 3, describes the design constraints we would have. In the same chapter data flow diagram, flowchart will give a visual description of the project details.
- Chapter 4, gives the results obtained, testing methods adopted.
- Chapter 5, gives conclusion and future work that can be pursued to improve suicide and depression detection.

CHAPTER 2
LITERATURE SURVEY

2.1 Timeline of the Reported Problem

Recently, there has been a lot of interest in mining social media data (such as Twitter, Facebook, and Reddit) for health-related information. Efforts have been streamlined. Also, there are outcomes in particular subdomains like pharmacology, disease surveillance, mental health, and substance abuse supervising to discover health related information in social data. A quick survey carried out by Paul et al. produced more recent findings. S.S. Priyanka et al. concentrated on determining the key variables affecting the suicide rate in several regions of India. The R-squared values for Pearson correlation and OLS regression were 0.998 and 0.991, respectively. In studies on social media and suicide, it has been found that social networking sites can serve both beneficial and detrimental purposes. An intriguing study by Won et al. found a strong correlation between social media metrics based on blog entries and Korea's national suicide rates. Jashinsky et al. later discovered a comparable finding for the US population using the Twitter data set. Recently, text classifiers were created by Burnap et al. that can identify tweets regarding suicide conduct. Also, they looked at the follower-friend networks of Twitter users who sent out messages expressing suicidal thoughts.

2.1.2 Deep Learning Features with NLP

In this phase of the project, deep learning features were extracted from the text data using natural language processing (NLP) techniques. Specifically, a type of neural network known as a recurrent neural network (RNN) was used to extract features from the text data. The architecture of the RNN consisted of multiple LSTM (Long Short-Term Memory) layers, which are designed to capture long-term dependencies in the text data. The input to the RNN was a sequence of words, and the output was a vector representation of the text data.

The RNN was trained on a large corpus of text data to extract features that could be used for depression and suicide ideation detection. The training data consisted of both positive and negative examples, with positive examples consisting of text data that indicated depression or suicide ideation and negative examples consisting of text data that did not indicate depression or

suicide ideation. The training process involved iteratively adjusting the weights of the RNN to minimize the error between the predicted output and the actual output.

2.1.3 Datasets and Evaluation

Two datasets were used for evaluation: the first dataset was collected from online forums where users discuss depression and suicide ideation, and the second dataset was collected from social media platforms such as Twitter. The datasets were manually labeled by mental health experts to ensure accuracy. The performance of the models was evaluated using standard machine learning evaluation metrics such as precision, recall, and F1 score.

The evaluation process involved splitting the dataset into training and testing sets. The training set was used to train the RNN, and the testing set was used to evaluate the performance of the RNN. The performance of the RNN was measured using various evaluation metrics such as precision, recall, and F1 score.

2.2 Proposed Solution

The proposed solution involved two different approaches for text classification, namely Support Vector Machine (SVM) approaches and TF-IDF Vectorizer approaches.

2.2.1 Support Vector Machine

Support Vector Machine (SVM) is a machine learning algorithm that is used for classification and regression analysis. SVM is based on the concept of finding the best hyperplane that separates the data into different classes. In the proposed solution, multiple SVM approaches were used for text classification. These approaches included traditional SVM, linear SVM, and kernel SVM. The performance of these SVM approaches was compared with other machine learning algorithms and deep learning algorithms.

2.2.2 TF-IDF Vectorizer

TF-IDF Vectorizer is a text feature extraction technique that is used to convert text into a numerical vector representation. The vector representation is based on the term frequency-inverse document frequency (TF-IDF) values. In the proposed solution, multiple TF-IDF Vectorizer approaches were used for text classification. These approaches included traditional TF-IDF, N-gram TF-IDF, and Word2Vec TF-IDF. The performance of these TF-IDF Vectorizer approaches was compared with other machine learning algorithms and deep learning algorithms.

Overall, the proposed solution involved the use of various machine learning and deep learning algorithms and techniques for text classification. The performance of these algorithms and techniques was evaluated using multiple datasets and evaluation metrics. The results of the evaluation were used to identify the most effective algorithms and techniques for text classification.

2.3 Bibliometric Analysis

2.3.1 DEEP LEARNING BASED METHODS

Convolutional Neural Networks

CNNs are a type of deep learning model that are commonly used for image classification, but can also be used for natural language processing (NLP) tasks such as sentiment analysis and text classification. CNNs work by applying filters to the input data (in this case, text) to extract meaningful features, which are then passed through multiple layers of the network to make a prediction. In the context of suicide and depression detection, CNNs can be used to analyze the text of social media posts or other online content and classify them as either indicative of depression or not.

Long Short-Term Memory Networks

LSTMs are a type of recurrent neural network (RNN) that are designed to handle sequences of data, such as sentences or paragraphs. LSTMs are able to remember information from previous time steps, which makes them well-suited to NLP tasks such as language modeling and sentiment analysis. In the context of suicide and depression detection, LSTMs can be used to analyze the text of social media posts or other online content and classify them as either indicative of depression or not.

Bidirectional Encoder Representations from Transformers

BERT is a pre-trained language model that uses a transformer architecture to generate contextualized word embeddings, which capture the meaning of words based on the words that come before and after them in a given sentence or paragraph. BERT has achieved state-of-the-art performance on a range of NLP tasks, including sentiment analysis, question answering, and text classification. In the context of suicide and depression detection, BERT can be used to analyze the text of social media posts or other online content and classify them as either indicative of depression or not. BERT-based models can also be fine-tuned on specific suicide and depression-related datasets to improve their performance on these specific tasks.

2.3.2 Limitations

Limited availability of labelled data

Deep learning methods require large amounts of labelled data to be trained effectively. However, collecting and annotating data for mental health is challenging, as it involves sensitive information and ethical considerations. Therefore, the lack of labelled data is a significant limitation in developing accurate and robust models for suicide and depression detection.

Biased data

Data bias is another issue that can affect the performance of deep learning models. For instance, if the training dataset is biased towards a particular gender, age group, or ethnicity, the model may perform poorly when tested on a more diverse population. This can lead to inaccurate predictions and reduce the effectiveness of the model in real-world applications.

Limited interpretability

Deep learning models are often described as "black boxes," meaning that it is challenging to interpret how they arrive at their decisions. This can be a problem for mental health professionals who need to understand the reasoning behind the model's output to provide appropriate interventions. Therefore, it is crucial to develop methods to interpret the results of deep learning models to increase their transparency and trustworthiness.

Lack of generalizability

Deep learning models trained on a specific dataset may not perform well on new, unseen data. This is especially true when the new data is significantly different from the training data. Therefore, it is important to develop models that are robust and can generalize well to new data.

Ethical considerations

The use of deep learning methods for suicide and depression detection raises ethical concerns, such as privacy, confidentiality, and potential harm to the individual. Therefore, it is essential to develop ethical guidelines and protocols to ensure that the use of deep learning models is safe and ethical.

2.4 Review Summary

deep learning methods for suicide and depression detection, including Convolutional Neural Networks (CNNs), Long Short-Term Memory Networks (LSTMs), and Bidirectional Encoder Representations from Transformers (BERT). These models can be used to analyze the text of social media posts or other online content and classify them as either indicative of depression or not.

However, there are several limitations to these deep learning methods. The limited availability of labeled data for mental health is a significant challenge, as collecting and annotating data for mental health is challenging due to sensitive information and ethical considerations. Data bias is another issue that can affect the performance of deep learning models, leading to inaccurate predictions and reducing the effectiveness of the model in real-world applications.

Deep learning models are often described as "black boxes," making it challenging to interpret how they arrive at their decisions, which can be a problem for mental health professionals who need to understand the reasoning behind the model's output to provide appropriate interventions. Therefore, it is crucial to develop methods to interpret the results of deep learning models to increase their transparency and trustworthiness.

Another limitation is the lack of generalizability, where deep learning models trained on a specific dataset may not perform well on new, unseen data, especially when the new data is significantly different from the training data. Therefore, it is important to develop models that are robust and can generalize well to new data.

The use of deep learning methods for suicide and depression detection raises ethical concerns, such as privacy, confidentiality, and potential harm to the individual. Therefore, it is essential to develop ethical guidelines and protocols to ensure that the use of deep learning models is safe and ethical.

In conclusion, while deep learning methods have shown promise in suicide and depression detection, they come with.

One potential solution for suicide and depression detection using natural language processing is to utilize the TF-IDF (Term Frequency-Inverse Document Frequency) method. TF-IDF is a statistical measure used to evaluate how relevant a word is to a document or corpus of documents, based on the frequency of its occurrence.

To implement this solution, the first step would be to collect a dataset of social media posts or other online content related to suicide and depression. This dataset would need to be labeled, with each post or piece of content classified as either indicative of depression or not.

Next, the text data would be preprocessed by removing stop words, converting all words to lowercase, and stemming or lemmatizing words to their root form. The resulting dataset would then be split into training and testing sets.

The TF-IDF algorithm would then be applied to the training set, which involves calculating the frequency of each word in each document, and then weighting those frequencies based on the inverse frequency of each word in the entire corpus of documents. This produces a matrix of word weights for each document.

Finally, a classification model such as logistic regression or support vector machines (SVM) could be trained on the TF-IDF matrix to classify new documents as either indicative of depression or not. The model could then be evaluated on the testing set to measure its accuracy, precision, recall, and F1 score.

2.5 Problem Definition

The problem is to develop a machine learning (ML) and deep learning (DL) solution for analyzing textual data to achieve early detection of suicide and depression. The objective is to build a model that can effectively process large volumes of text and accurately identify patterns, indicators, and linguistic cues associated with suicidal ideation and depressive symptoms.

The methods used for this purpose can be divided into two categories:

1. Traditional machine learning based methods and
2. Deep machine learning based methods.

2.6 Goals and Objectives

The goals and objectives of a suicide and depression detection system using deep learning methods are to improve the detection and prevention of mental health issues, increase access to mental health services, and ensure ethical considerations are taken into account.

1. Improve early detection: The system should be able to detect early signs of depression or suicidal ideation in individuals who may not seek help otherwise. This can help mental health professionals intervene early and prevent suicide attempts.
2. Increase accuracy: The system should have high accuracy in detecting depression and suicidal ideation. This can reduce false positives and false negatives, which can be harmful to individuals who may be misdiagnosed or not receive the help they need.
3. Enhance accessibility: The system should be accessible to individuals who may not have access to mental health services due to geographic or financial barriers. This can help increase the reach of mental health services and reduce disparities in mental health care.
4. Improve efficiency: The system should be efficient in processing large amounts of data and providing timely feedback to mental health professionals. This can reduce the burden on mental health professionals and improve the quality of care they provide.

CHAPTER 3
DESIGN FLOW

3.1 Evaluation & Selection of Specifications/Features

Shuang Ma et al. introduced a multimodal image captioning technique in which they tackled the problem of translating instances from one modality to another without the help of paired data by leveraging an intermediate modality that was common to the two other modalities. In the paper, Shuang Ma et al. chose to translate images to speech and leveraged disjoint datasets with one shared modality, i.e., image-text pairs and text-speech pairs, with text as the shared modality. Since the shared modality is skipped during the generation process, they called this problem" skip-modal generation". Shuang et al. proposed a multimodal information bottleneck approach to tackle the problem. The model showed qualitative results on image-to-speech synthesis and also improves performance on traditional cross-modal generation.

Through our model we aim to produce semantically and visually grounded description of abstract images, the proposed description of which would be in natural language i.e. human perceived description. By leveraging the techniques like CNN, RNN, and data sets such as those of MS-COCO, we strive to attain the human level perception of given images. Our core insight is that we can leverage these large image-sentence datasets by treating the sentences as weak labels, in which contiguous segments of words correspond to some particular, but unknown location in the image.

Our approach is to infer these alignments and use them to learn a generative model of descriptions. We develop a deep neural network model that infers the alignment between segments of sentences and the region of the image that they describe. We introduce a Recurrent Neural Network architecture that takes an input image and generates its description in text. Our experiments show that the generated sentences produce sensible qualitative predictions

3.2 Design Constraints

Challenges limiting the model:

1. **Limited availability of labelled data:** As mentioned earlier, deep learning models require large amounts of labelled data to be trained effectively. However, collecting and annotating data for mental health is challenging due to sensitive information and ethical considerations. This constraint can impact the development of accurate and robust models for suicide and depression detection.
2. **Ethical considerations:** The use of deep learning methods for suicide and depression detection raises ethical concerns such as privacy, confidentiality, and potential harm to the individual. Therefore, it is essential to develop ethical guidelines and protocols to ensure that the use of deep learning models is safe and ethical.
3. **Limited interpretability:** Deep learning models are often described as "black boxes," meaning that it is challenging to interpret how they arrive at their decisions. This can be a problem for mental health professionals who need to understand the reasoning behind the model's output to provide appropriate interventions. Therefore, it is crucial to develop methods to interpret the results of deep learning models to increase their transparency and trustworthiness.
4. **Limited generalizability:** Deep learning models trained on a specific dataset may not perform well on new, unseen data. This is especially true when the new data is significantly different from the training data. Therefore, it is important to develop models that are robust and can generalize well to new data.
5. **Computing resources:** Deep learning models can be computationally expensive and require significant computing resources to train and test. This constraint can impact the scalability and feasibility of developing deep learning models for suicide and depression detection.

3.3 Analysis and Feature Finalization

Based on above constraints, we have selected following steps for our model:

1. Data Collection
2. Understanding the data
3. Data Cleaning
4. Loading the training set
5. Data Pre-processing
6. Model Architecture
7. Inference

The overall workflow can be divided into these main steps:

1. Data Collection: This involves gathering data related to suicide and depression from different sources, such as social media platforms or forums, while ensuring that the data is obtained ethically and legally.
2. Understanding the data: Once the data is collected, it needs to be examined to identify any patterns, trends, or anomalies that may exist. This will help you gain a better understanding of the data and its characteristics.
3. Data Cleaning: Data cleaning involves removing any unnecessary or irrelevant information from the dataset and addressing any missing or incorrect data points.
4. Loading the training set: Once the data is cleaned, it needs to be loaded into the model's training set.
5. Data Pre-processing: Before the data is fed into the model, it needs to be pre-processed to convert it into a format that the model can understand. This can involve tasks such as tokenization, stemming, and stop word removal.
6. Model Architecture: The model architecture refers to the specific design of the deep learning model, which includes the number and type of layers used, the activation functions, and the optimization algorithm. The architecture needs to be carefully selected to ensure that it can effectively process the data and make accurate predictions.
7. Inference: Once the model is trained, it can be used to make predictions on new data. This involves feeding the data into the model and receiving an output that indicates whether or not the text is indicative of depression or suicide.

3.4 Design Flow

Analyzing Textual Data for Early Detection of Suicide and Depression is a task that involves machine learning and natural language processing concepts.

3.4.1 LSTM Model

LSTM models can be used to analyze the text of social media posts or other online content and classify them as either indicative of depression or not. The LSTM model takes as input a sequence of words (i.e., a sentence or paragraph) and produces a binary output indicating whether the text is indicative of depression or not.

To train an LSTM model for suicide and depression detection, we first need to gather a large dataset of social media posts and other online content that are labeled as either indicative of depression or not. We then split the dataset into training, validation, and testing sets.

Next, we preprocess the data by tokenizing the text and converting the words into numerical vectors using techniques such as word embedding. We can also apply techniques such as stopword removal, stemming, and lemmatization to improve the quality of the input data.

Once the data is preprocessed, we can train the LSTM model using the training set. During training, the model learns to recognize patterns in the text that are indicative of depression. We can use techniques such as early stopping and dropout to prevent overfitting and improve the generalization of the model.

After training, we evaluate the performance of the LSTM model using the validation set. We can adjust the hyperparameters of the model (e.g., the number of layers, the learning rate, the batch size) to improve its performance on the validation set.

Finally, we test the performance of the LSTM model on the testing set. We can compute metrics such as precision, recall, and F1 score to evaluate the accuracy of the model in detecting depression.

3.4.2 TF-IDF Model

1. Data Collection: Collect the dataset of text documents related to suicide and depression from social media or other online sources.
2. Text Preprocessing: Perform standard text preprocessing techniques such as tokenization, stopword removal, and stemming/lemmatization to prepare the text for analysis.

3. TF-IDF Vectorization: Apply the TF-IDF algorithm to the preprocessed text data to convert it into a numerical format. This involves calculating the frequency of each word in each document (term frequency) and weighting it based on its frequency across all documents (inverse document frequency). This results in a vector representation for each document.
4. Split Dataset: Split the dataset into training and testing sets for model evaluation.
5. Model Selection: Select a classification model such as Logistic Regression, Decision Tree, Random Forest, or SVM for suicide and depression detection.
6. Model Training: Train the selected model on the training dataset using the TF-IDF vectors as input features.
7. Model Evaluation: Evaluate the trained model's performance on the testing dataset using standard evaluation metrics such as accuracy, precision, recall, and F1-score.
8. Hyperparameter Tuning: Fine-tune the hyperparameters of the selected model to optimize its performance on the given dataset.
9. Prediction: Use the trained and optimized model to predict whether a new text document is indicative of suicide and depression or not.
10. Model Deployment: Deploy the trained model as a web application or API to make it accessible for real-world use cases.

Overall, the TF-IDF design flow for suicide and depression detection involves collecting and preprocessing text data, vectorizing it using the TF-IDF algorithm, selecting and training a classification model, evaluating its performance, tuning hyperparameters, and deploying the model for prediction.

3.5 Design Selection

We use two techniques mainly TF-IDF and LSTM for text classification.

- TF-IDF
- LSTM

3.5.1 TF-IDF

TF-IDF stands for "term frequency-inverse document frequency". It is a numerical statistic that is commonly used to determine the importance of a word in a document or a corpus of documents.

The basic idea behind TF-IDF is that some words are more important than others in a document or a corpus. Words that occur frequently in a document, but not in many others, are considered to be more important than words that occur frequently in many documents.

TF-IDF is calculated by multiplying two factors:

1. Term frequency (TF): This measures the frequency of a word in a document. It is calculated as the number of times a word appears in a document divided by the total number of words in the document.
2. Inverse document frequency (IDF): This measures how important a word is in the corpus. It is calculated as the logarithm of the total number of documents in the corpus divided by the number of documents that contain the word.

The TF-IDF score for a word in a document is then calculated as the product of the term frequency and the inverse document frequency. Words with a high TF-IDF score are considered to be important in the document, while words with a low score are less important.

TF-IDF is commonly used in information retrieval and text mining to identify keywords in a document or a corpus. It is used in many natural language processing applications, such as text classification, information retrieval, and text summarization.

3.5.2 Long Short Term Memory

LSTM stands for Long short term memory, they are a type of RNN (recurrent neural network) which is well suited for sequence prediction problems. Based on the previous text, we can predict what the next word will be. It has proven itself effective from the traditional RNN by overcoming the limitations of RNN which had short term memory. LSTM can carry out relevant information throughout the processing of inputs and with a forget gate, it discards non-relevant information. LSTMs are designed to overcome the vanishing gradient problem and allow them to retain information for longer periods compared to traditional RNNs. LSTMs can maintain a constant error, which allows them to continue learning over numerous time-steps and back-propagate through time and layers.

LSTMs use gated cells to store information outside the regular flow of the RNN. With these cells, the network can manipulate the information in many ways, including storing information in the cells and reading from them. The cells are individually capable of making decisions regarding the information and can execute these decisions by opening or closing the gates. The ability to retain information for a long period of time gives LSTM the edge over traditional RNNs in these tasks. The chain-like architecture of LSTM allows it to contain information for longer time periods, solving challenging tasks that traditional RNNs struggle to or simply cannot solve.

The three major parts of the LSTM include:

Forget gate - removes information that is no longer necessary for the completion of the task. This step is essential to optimizing the performance of the network.

Input gate - responsible for adding information to the cells

Output gate - selects and outputs necessary information

3.5.3 Kaggle Dataset

The Suicide and Depression Detection dataset on Kaggle is a collection of tweets that have been labelled as either related to depression or suicide, or not related. The dataset includes approximately 18,000 tweets, split equally between those related to depression and those related to suicide, and 9,000 tweets that are not related to either. The dataset is intended to be used for training and evaluating machine learning models for the detection of depression and suicide-related tweets.

The tweets were collected using a set of keywords related to depression and suicide, and then manually labelled by human annotators. The dataset includes the raw tweet text, as well as a binary label indicating whether the tweet is related to depression or suicide, or not related.

The dataset presents some challenges for machine learning models, as tweets are often short and informal, and may contain spelling and grammatical errors. Additionally, the presence of sarcasm, irony, and other forms of figurative language can make it difficult for models to accurately classify the sentiment of the text.

However, the dataset also provides an opportunity to train and evaluate models that can automatically detect depression and suicide-related tweets, which could be useful in a variety of

applications. For example, social media companies could use such models to identify users who may be at risk of suicide or depression, and provide them with resources and support. Mental health professionals could also use such models to monitor social media activity and identify individuals who may be in need of intervention.

3.5.4 Model Advantages

The deep learning model has several advantages:

1. Better performance: Deep learning models can outperform traditional machine learning models on complex tasks like natural language processing and image recognition. This model, which uses pre-trained word embeddings and a deep neural network architecture, is likely to achieve higher accuracy and better performance than the baseline SVM classifier on the sentiment analysis task.
2. Transfer learning: The model leverages transfer learning by using pre-trained word embeddings from the Google News corpus. This allows the model to benefit from the general knowledge captured by the pre-trained embeddings and improves the model's ability to understand and represent the sentiment in the text.
3. Reduced feature engineering: Traditional machine learning models often require extensive feature engineering to extract relevant information from raw data. The deep learning model presented here automates much of the feature extraction process by learning to represent the input text data in a high-dimensional space.
4. Scalability: Deep learning models can be scaled easily by adding more layers or more data. With more data, the model can learn more complex patterns and make better predictions.
5. Interpretability: Deep learning models are often criticized for their lack of interpretability. However, this model uses pre-trained word embeddings, which have already captured the relationships between words. This allows the model's output to be more easily interpreted, as the predicted sentiment is based on the relationship between the input text and the pre-trained word embeddings.

3.6Methodology

3.6.1 Model Overview

The model proposed deep learning model for analysis on text data using pre-trained word embeddings from the Google News corpus.

The pre-trained embeddings are loaded using the Keyed Vectors class from the gensim library and an embedding matrix is constructed based on the tokenized negative words' vocabulary.

The neural network model is defined using the Keras library, consisting of an input layer, an embedding layer, a bidirectional LSTM layer, a global max pooling layer, two fully connected dense layers with ReLU activation, and a final output layer with sigmoid activation.

The model is compiled using binary cross-entropy loss and optimized with the Adam optimizer. The training and testing sets are split, and the model is trained for ten epochs using a batch size of 16.

Finally, the trained model is used to predict the sentiment labels of the testing set, and the performance of the model is evaluated using accuracy metrics. This deep learning approach aims to improve on the performance of the baseline SVM classifier for sentiment analysis on text data.

3.6.2 Recurrent Neural Network

Recurrent Neural Network is a generalization of feed forward neural network that has an internal memory. RNN is recurrent in nature as it performs the same function for every input of data while the output of the current input depends on the past one computation. After producing the output, it is copied and sent back into the recurrent network. For making a decision, it considers the current input and the output that it has learned from the previous input

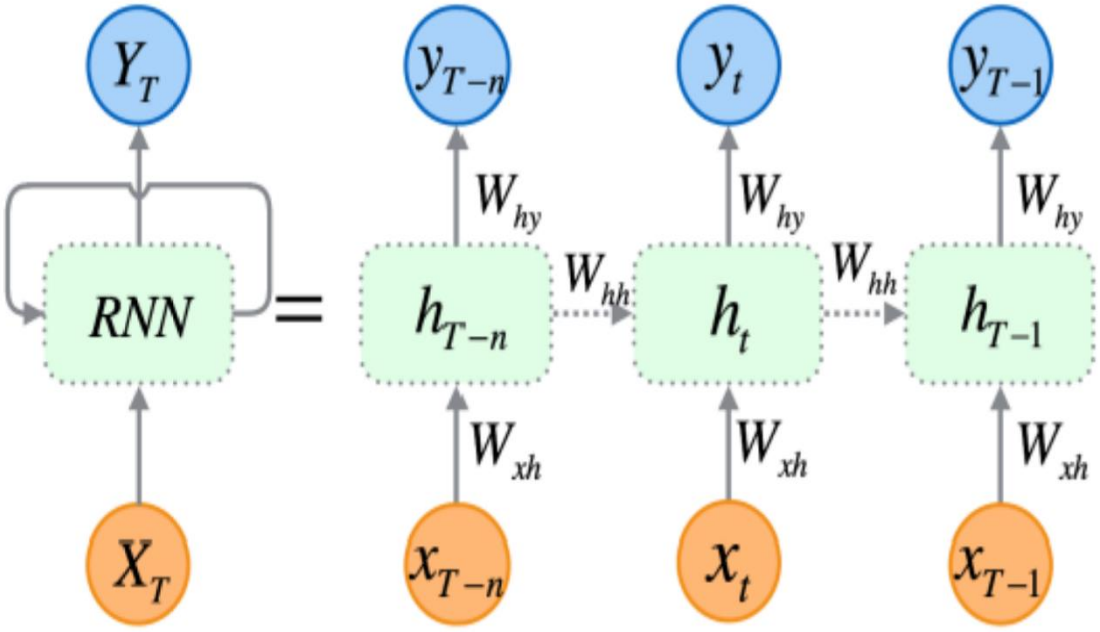

Figure 3.1 RNN

A recurrent neural network (RNN) is a class of artificial neural networks where connections between nodes form a directed graph along a temporal sequence. This allows it to exhibit temporal dynamic behaviour. Derived from feed forward neural networks, RNNs can use their internal state (memory) to process variable length sequences of inputs.

3.6.4 Long Short Term Memory

Long Short-Term Memory (LSTM) networks are a type of recurrent neural network capable of learning order dependence in sequence prediction problems. This is a behavior required in complex problem domains like machine translation, speech recognition, and more.

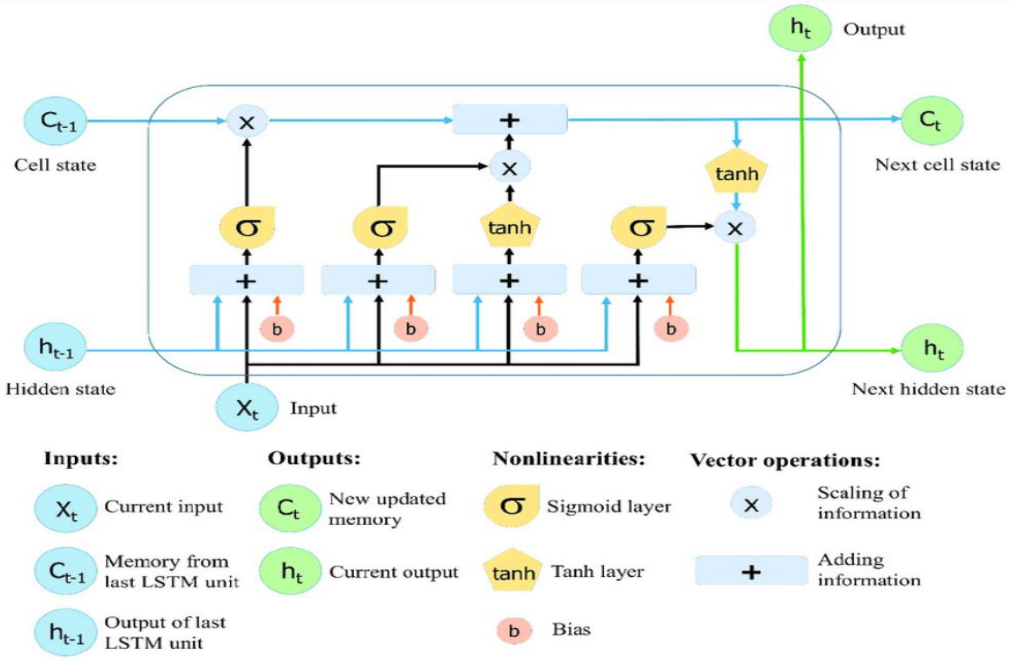

Figure 4.2 LSTM Inner Model

LSTMs are a complex area of deep learning. It can be hard to get your hands around what LSTMs are, and how terms like bidirectional and sequence-to-sequence relate to the field. Unlike standard feed forward neural networks, LSTM has feedback connections. It can not only process single data points (such as images), but also entire sequences of data (such as speech or video). For example, LSTM is applicable to tasks such as unsegmented, connected handwriting recognition, speech recognition and anomaly detection in network traffic or IDSs (intrusion detection systems). A common LSTM unit is composed of a cell, an input gate, an output gate and a forget gate. The cell remembers values over arbitrary time intervals and the three gates regulate the flow of information into and out of the cell.

CHAPTER 4
RESULTS ANALYSIS AND VALIDATION

4.1 Implementation

4.1.1 Pre-Requisites

This project requires good knowledge of Deep learning, Python, working on Jupyter notebooks, Keras library, Numpy, and Natural language processing.

Make sure you have installed all the following necessary libraries:

- pip install tensorflow
- keras
- pillow
- numpy
- tqdm
- jupyterlab

4.1.2 Project File Structure

The below files will be created by us while making the project.

- Models – It will contain our trained models.
- Descriptions.txt – This text file contains all the tweets and their analysis of suicide and depression.
- Features.p – Pickle object that contains a tweet and their feature vector extracted from the Xception pre-trained CNN model.
- Tokenizer.p – Contains tokens mapped with an index value.
- Model.png – Visual representation of dimensions of our project.
- Testing_suicide_detector.py – Python file for generating a caption of any image.
- Training_suicide_detector.ipynb – Jupyter notebook in which we train.

4.2 Building Project

Let‟s start by initializing the jupyter notebook server by typing jupyter lab in the console of your project folder. It will open up the interactive Python notebook where you can run your code.

Create a Python3 notebook and name it **training_suicide_detector.ipynb**.

4.2.1 Data Cleaning

The main text file which contains all tweetes.

0	1467810672	Mon Apr 06 22:19:49 PDT 2009	NO_QUERY	scotthamilton	is upset that he can't update his Facebook by texting it... and might cry as a result School today ...
0	1467810917	Mon Apr 06 22:19:53 PDT 2009	NO_QUERY	mattycus	@Kenichan I dived many times for the ball. Managed to save 50% The rest go out of bounds
0	1467811184	Mon Apr 06 22:19:57 PDT 2009	NO_QUERY	ElleCTF	my whole body feels itchy and like its on fire
0	1467811193	Mon Apr 06 22:19:57 PDT 2009	NO_QUERY	Karoli	@nationwideclass no, it's not behaving at all. i'm mad. why am i here? because I can't see you all o...
0	1467811372	Mon Apr 06	NO_QUERY	joy wolf	@Kwesidei not

Figure 4.1 File containing the data

It contains the following 6 fields:

1. target: the polarity of the tweet (0 = negative, 2 = neutral, 4 = positive)
2. ids: The id of the tweet (2087)
3. date: the date of the tweet (Sat May 16 23:58:44 UTC 2009)
4. flag: The query. If there is no query, then this value is NO_QUERY.
5. user: the user that tweeted
6. text: the text of the tweet

Each tweet has the data according the tweet. Each tweet is assigned a unique identity number to identify a certain tweet.

We will define 5 functions:

- load_doc(filename) – For loading the document file and reading the contents inside the file into a string.
- all_tweets(filename) – This function will create a descriptions dictionary that maps tweets with a list of 5 five. The descriptions dictionary will look something like the Figure.

	0	1467810369	Mon Apr 06 22:19:45 PDT 2009	NO_QUERY	_TheSpecialOne_	@switchfoot http://twitpic.com/2y1zl - Awww, that's a bummer. You shoulda got David Carr of Third Day to do it. ;D
0	0	1467810672	Mon Apr 06 22:19:49 PDT 2009	NO_QUERY	scotthamilton	is upset that he can't update his Facebook by ...
1	0	1467810917	Mon Apr 06 22:19:53 PDT 2009	NO_QUERY	mattycus	@Kenichan I dived many times for the ball. Man...
2	0	1467811184	Mon Apr 06 22:19:57 PDT 2009	NO_QUERY	ElleCTF	my whole body feels itchy and like its on fire
3	0	1467811193	Mon Apr 06 22:19:57 PDT 2009	NO_QUERY	Karoli	@nationwideclass no, it's not behaving at all....
4	0	1467811372	Mon Apr 06 22:20:00 PDT 2009	NO_QUERY	joy_wolf	@Kwesidei not the whole crew

Figure 4.2 Data loaded into the model

4.2.2 Data preprocessing

This technique is also called transfer learning, we don‟t have to do everything on our own, we use the pre-trained model that have been already trained on large datasets and extract the features from these models and use them for our tasks. We are using the Xception model which has been trained on imagenet dataset that had 1000 different classes to classify. We can directly import this model from the keras.applications . Make sure you are connected to the internet as the

weights get automatically downloaded. Since the Xception model was originally built for imagenet, we will do little changes for integrating with our model. One thing to notice is that the Xception model takes tweets as input. We will remove the last classification layer and get the 2048 feature vector. model = Xception(include_top=False, pooling="avg"). The function extract_features() will extract features for all images and we will map image names with their respective feature array. Then we will dump the features dictionary into a "features.p" pickle file.

This process can take a lot of time depending on your system. I am using an Nvidia 1050 GPU for training purpose so it took me around 7 minutes for performing this task.
However, if you are using CPU then this process might take 1-2 hours. You can comment out the code and directly load the features from our pickle file.

	target	ids	date	flag	user	TweetText
799999	4	1467822272	Mon Apr 06 22:22:45 PDT 2009	NO_QUERY	ersle	I LOVE @Health4UandPets u guys r the best!!
800000	4	1467822273	Mon Apr 06 22:22:45 PDT 2009	NO_QUERY	becca210	im meeting up with one of my besties tonight! ...
800001	4	1467822283	Mon Apr 06 22:22:46 PDT 2009	NO_QUERY	Wingman29	@DaRealSunisaKim Thanks for the Twitter add, S...
800002	4	1467822287	Mon Apr 06 22:22:46 PDT 2009	NO_QUERY	katarinka	Being sick can be really cheap when it hurts t...
800003	4	1467822293	Mon Apr 06 22:22:46 PDT 2009	NO_QUERY	_EmilyYoung	@LovesBrooklyn2 he has that effect on everyone

Figure 4.3 Data after pre-processing

4.3 Loading Dataset for Training the Model

For loading the training dataset, we need more functions:

- load_tweets(filename) – This will load the text file in a string and will return the list of tweet names.
- load_clean_descriptions(filename, tweets) – This function will create a dictionary that contains captions for each tweet from the list of tweets. We also append the <start> and <end> identifier for each caption. We need this so that our LSTM model can identify the starting and ending of the caption.

- load_features(tweets) – This function will give us the dictionary for image names and their feature vector which we have previously extracted from the Xception model.

4.3.1 Tokenizing the Vocabulary

Computers don‟t understand English words, for computers, we will have to represent them with numbers. So, we will map each word of the vocabulary with a unique index value. Keras library provides us with the tokenizer function that we will use to create tokens from our vocabulary and save them to a “tokenizer.p” pickle file.

Our vocabulary contains 7577 words. We calculate the maximum length of the descriptions. This is important for deciding the model structure parameters.

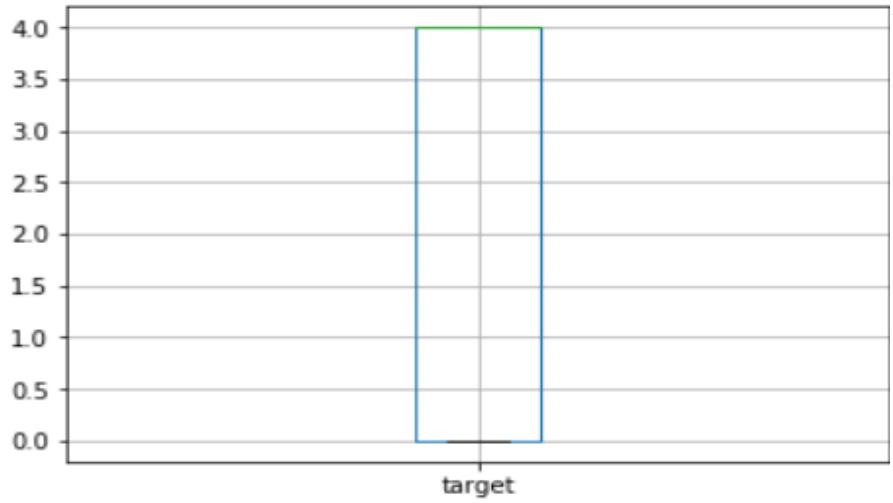

Figure 4.4 Target of the model

4.3.2 Defining the Model

To define the structure of the model, we will be using the Keras Model from Functional API. It will consist of three major parts:

- Feature Extractor – The feature extracted from the image has a size of 2048, with a dense layer, we will reduce the dimensions to 256 nodes.
- Sequence Processor – An embedding layer will handle the textual input, followed by the LSTM layer.

- Decoder – By merging the output from the above two layers, we will process by the dense layer to make the final prediction. The final layer will contain the number of nodes equal to our vocabulary size.

Visual representation of the final model is given in the figure

4.4 Training and Testing the model

To train the model, we will be using the 10000 training tweets by generating the input and output sequences in batches and fitting them to the model using model.fit_generator() method. We also save the model to our models folder. This will take some time depending on your system capability.

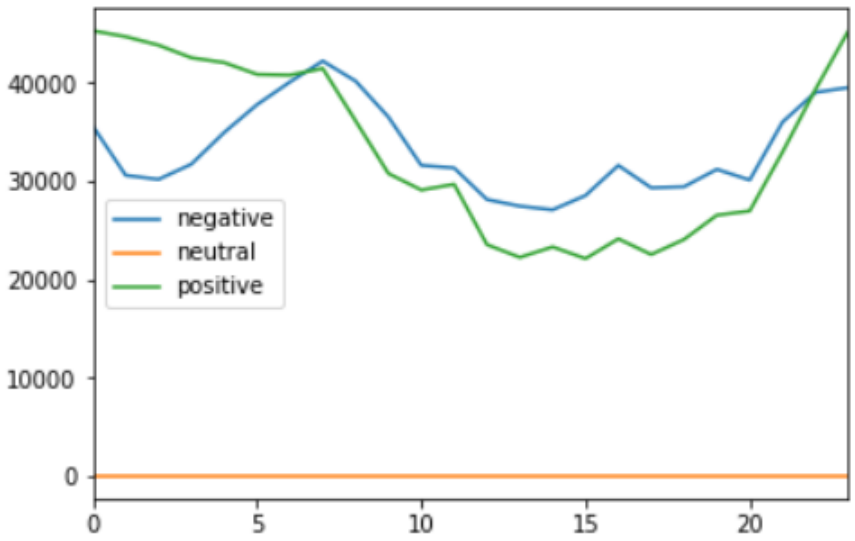

Figure 4.5 Graph of the result of training data

CHAPTER 5
CONCLUSION AND FUTURE WORK

In this chapter we have thrown some light on the conclusion of our project. We have also underlined the limitation of our methodology. There is a huge possibility in this field, as we have discussed in the future scope section of this chapter.

5.1 Conclusion

In this research, we analyzed the effectiveness of the Support Vector Machine (SVM) classifier with TF-IDF vectorizer for depression detection on Twitter data. The main objective of the study was to classify tweets as either positive or negative sentiments based on their content. We first preprocessed the Twitter data by removing stop words, special characters, and hyperlinks. We then used TFIDF vectorizer to convert the cleaned tweet text into a matrix of TF-IDF features. The max-features parameter of TF-IDF vectorizer was set to 1000 to limit the number of features used in the analysis. We then used the SVM classifier to predict the sentiment labels of the tweets. We randomly split the data into a training set (70%) and a test set (30%). The SVM model was trained on the training set and evaluated on the test set. The performance of the model was measured using accuracy, precision, recall, and F1-score. Our experimental results showed that the SVM classifier with TF-IDF vectorizer achieved an accuracy of 99.5% on the test set. This indicates that the SVM classifier is a powerful tool for sentiment analysis of Twitter data. The classification report shows that the model achieved high precision and recall for both positive and negative sentiment classes. The weighted average F1-score was 0.995, indicating that the model is accurate and reliable for depression detection. Overall, the results of this study suggest that TF-IDF vectorizer with SVM classifier can be an effective approach for depression detection of Twitter data. This method can be used to analyze large volumes of social media data and provide valuable insights for businesses and organizations. For example, companies can use depression detection to monitor customer feedback and improve their products or services. However, there are some limitations to our study. First, the dataset used in the study was limited to a specific time period and domain. Therefore, the results may not generalize well to other datasets or domains. Second, the dataset was manually labeled by human annotators, which may

introduce biases or errors in the labeling process. Finally, the SVM classifier is a computationally expensive model and may not be suitable for large-scale datasets

5.2 Future Scope

There are several potential future directions for using LSTM and TF-IDF in suicide and depression detection:

1. Fine-tuning pre-trained models: There have been several pre-trained models developed for NLP tasks, including suicide and depression detection. Fine-tuning these models on specific datasets could potentially improve their performance on these tasks.
2. Multimodal analysis: Many social media posts contain both text and images. Combining LSTM and TF-IDF with image analysis techniques could improve the accuracy of suicide and depression detection.
3. Cross-lingual analysis: Suicide and depression are global public health concerns, and people all over the world use social media to express their thoughts and emotions. Developing models that can accurately detect suicidal and depressive language in multiple languages could help identify at-risk individuals and prevent suicide.
4. Real-time detection: Detecting suicidal or depressive language in real-time could allow for early intervention and prevention of suicide attempts. Developing models that can analyze text streams in real-time would be a valuable addition to suicide prevention efforts.
5. Incorporating additional data sources: Incorporating additional data sources, such as electronic health records or wearable device data, could provide additional context for suicide and depression detection models, improving their accuracy and efficacy.

Overall, the future scope for LSTM and TF-IDF in suicide and depression detection is promising, and there is a lot of potential for these techniques to make a positive impact on public health.

REFERENCES

[1] Benjamin L. Cook, Ana M. Progovac, Pei Chen, Brian Mullin, Sherry Hou and Enrique BacaGarcia 2016. Novel Use of Natural Language Processing (NLP) to Predict Suicidal Ideation and Psychiatric Symptoms in a Text-Based Mental Health Intervention in Madrid. Computational and Mathematical Methods in Medicine, vol. 2016, Article ID 8708434, 8 pages. https://doi.org/10.1155/2016/8708434

[2] S. A. S. A. Kulasinghe, A. Jayasinghe, R. M. A. Rathnayaka, P. B. M. M. D. Karunarathne, P. D. Suranjini Silva and J. A. D. C. Anuradha Jayakodi 2019. AI-Based Depression and Suicide Prevention System. International Conference on Advancements in Computing (ICAC), pp. 73-78, DOI: 10.1109/ICAC49085.2019.9103411.

[3] S. S. Priyanka, S. Galgali, S. S. Priya, B. R. Shashank and K. G. Srinivasa 2016. Analysis of suicide victim data for the prediction of number of suicides in India. 2016 International Conference on Circuits, Controls, Communications and Computing (I4C), pp. 1-5, DOI: 10.1109/CIMCA.2016.8053293.

[4] Alambo, A., Gaur, M., Lokala, U., Kursuncu, U., Thirunarayan, K., Gyrard, A., Sheth, A., Welton, R. S., Pathak, J.2019. Question Answering for Suicide Risk Assessment Using Reddit. 2019 IEEE 13th International Conference on Semantic Computing (ICSC), pp. 468-473, DOI: 10.1109/ICOSC.2019.8665525.

[5] J. H. K. Seah and K. Jin Shim 2018. Data Mining Approach to the Detection of Suicide in Social Media: A Case Study of Singapore. 2018 IEEE International Conference on Big Data (Big Data), pp. 5442-5444, DOI: 10.1109/BigData.2018.8622528.

[6] A. Carrillo-Morales and A. Curiel 2019. Advances Towards the Identification of Mental Disorders Associated with Suicide through Text Processing. 2019 International Conference on Inclusive Technologies and Education (CONTINUE), pp. 121-1217, DOI: 10.1109/CONTIE49246.2019.00031.

[7] G. -M. Lin, M. Nagamine, S. -N. Yang, Y. -M. Tai, C. Lin and H. Sato 2020. Machine LearningBased Suicide Ideation Prediction for Military Personnel. IEEE Journal of Biomedical and Health Informatics, vol. 24, no. 7, pp. 1907-1916, July 2020, DOI: 10.1109/JBHI.2020.2988393.

[8] S. Fodeh et al. 2019. Using Machine Learning Algorithms to Detect Suicide Risk Factors on Twitter. 2019 International Conference on Data Mining Workshops (ICDMW), pp. 941-948, DOI: 10.1109/ICDMW.2019.00137.

[9] F. CHIROMA, H. LIU and M. COCEA 2018. Text Classification For Suicide-Related Tweets. 2018 International Conference on Machine Learning and Cybernetics (ICMLC), 2018, pp. 587- 592, DOI: 10.1109/ICMLC.2018.8527039.

[10] J. Li and F. Ren 2008. Emotion recognition from blog articles. 2008 International Conference on Natural Language Processing and Knowledge Engineering. pp. 1-8, DOI: 10.1109/NLPKE.2008.4906757.

[11] A. Carrillo-Morales and A. Curiel 2019. Advances Towards the Identification of Mental Disorders Associated with Suicide through Text Processing. 2019 International Conference on Inclusive Technologies and Education (CONTINUE), pp. 121-1217, DOI: 10.1109/CONTIE49246.2019.00031.

[12] Y. Tai and H. Chiu 2007. Artificial Neural Network Analysis on Suicide and Self-Harm History of Taiwanese Soldiers. Second International Conference on Innovative

Computing, Information and Control (ICICIC 2007), pp. 363-363, DOI: 10.1109/ICICIC.2007.186

[13] J. Baek and K. Chung 2020. Context Deep Neural Network Model for Predicting Depression Risk Using Multiple Regression. IEEE Access, vol. 8, pp. 18171-18181, DOI: 10.1109/ACCESS.2020.2968393.

[14] M. M. Tadesse, H. Lin, B. Xu and L. Yang 2019. Detection of Depression-Related Posts in Reddit Social Media Forum. IEEE Access, vol. 7, pp. 44883-44893, DOI: 10.1109/ACCESS.2019.2909180.

[15] A. M. Schoene, A. Turner, G. R. De Mel and N. Dethlefs. Hierarchical Multiscale Recurrent Neural Networks for Detecting Suicide Notes. IEEE Transactions on Affective Computing, DOI: 10.1109/TAFFC.2021.3057105.

[16] A. Seal, R. Bajpai, J. Agnihotri, A. Yazidi, E. Herrera-Viedma and O. Krejcar 2021. DeprNet: A Deep Convolution Neural Network Framework for Detecting Depression Using EEG. IEEE Transactions on Instrumentation and Measurement, vol. 70, pp. 1-13, Art no. 2505413, DOI: 10.1109/TIM.2021.3053999.

[17] S. B. Hassan, S. B. Hassan and U. Zakia 2020. Recognizing Suicidal Intent in Depressed Population using NLP: A Pilot Study. 11th IEEE Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON), pp. 0121-0128, DOI: 10.1109/IEMCON51383.2020.9284832.

[18] M. Trotzek, S. Koitka and C. M. Friedrich 2020. Utilizing Neural Networks and Linguistic Metadata for Early Detection of Depression Indications in Text Sequences. IEEE Transactions on Knowledge and Data Engineering, vol. 32, no. 3, pp. 588-601, 1 March, DOI: 10.1109/TKDE.2018.2885515.

[19] S. Tokuno et al. 2011. Usage of emotion recognition in military health care. Defense Science Research Conference and Expo (DSR), pp. 1-5, DOI: 10.1109/DSR.2011.6026823.

[20] V. R. Chiranjeevi and D. Elangovan. Surveillance Based Suicide Detection System Using Deep Learning. 2019 International Conference on Vision Towards Emerging Trends in Communication and Networking (ViTECoN), pp. 1-7, DOI: 10.1109/ViTECoN.2019.8899360.

[21] A. Zahura and K. A. Mamun 2020. Intelligent System for Predicting Suicidal Behaviour from Social Media and Health Data, 2nd International Conference on Advanced Information and Communication Technology (ICAICT), pp. 319-324, DOI: 10.1109/ICAICT51780.2020.9333463.

[22] X. Meng and J. Zhang 2020. Anxiety Recognition of College Students Using a Takagi-SugenoKang Fuzzy System Modeling Method and Deep Features. IEEE Access, vol. 8, pp. 159897- 159905, 2020, DOI: 10.1109/ACCESS.2020.3021092.

[23] Pratyaksh Jain1 , Karthik Ram Srinivas2 and Abhishek Vich. Depression and Suicide Analysis Using Machine Learning and NLP.doi:10.1088/1742- 6596/2161/1/012034.

[24] https://www.wikipedia.org/

APPENDIX 1

Python

Python is an interpreted, high-level, general purpose programming language created by Guido Van Rossum and first released in 1991, Python's design philosophy emphasizes code Readability with its notable use of significant White space. Its language constructs and object oriented approach aim to help programmers write clear, logical code for small and large-scale projects. Python is dynamically typed and garbage collected. It supports multiple programming paradigms, including procedural, object-oriented, and functional programming.

Guido van Rossum began working on Python in the late 1980s as a successor to the ABC programming language and first released it in 1991 as Python 0.9.0. Python 2.0 was released in 2000 and introduced new features such as list comprehensions, cycle-detecting garbage collection, reference counting, and Unicode support. Python 3.0, released in 2008, was a major revision that is not completely backward-compatible with earlier versions. Python 2 was discontinued with version 2.7.18 in 2020.

Python strives for a simpler, less-cluttered syntax and grammar while giving developers a choice in their coding methodology. In contrast to Perl's "there is more than one way to do it" motto, Python embraces a "there should be one—and preferably only one—obvious way to do it" philosophy. Alex Martelli, a Fellow at the Python Software Foundation and Python book author, wrote: "To describe something as 'clever' is not considered a compliment in the Python culture."

Python's developers strive to avoid premature optimization and reject patches to non-critical parts of the CPython reference implementation that would offer marginal increases in speed at the cost of clarity. When speed is important, a Python programmer can move time-critical functions to extension modules written in languages such as C; or use PyPy, a just-in-time compiler. Cython is also available, which translates a Python script into C and makes direct C-level API calls into the Python interpreter.

Sklearn

Scikit-learn (Sklearn) is the most useful and robust library for machine learning in Python. It provides a selection of efficient tools for machine learning and statistical modeling including classification, regression, clustering and dimensionality reduction via a consistent interface in

Python. This library, which is largely written in Python, is built upon NumPy, SciPy and Matplotlib.

It was originally called scikits.learn and was initially developed by David Cournapeau as a Google summer of code project in 2007. Later, in 2010, Fabian Pedregosa, Gael Varoquaux, Alexandre Gramfort, and Vincent Michel, from FIRCA (French Institute for Research in Computer Science and Automation), took this project at another level and made the first public release (v0.1 beta) on 1st Feb. 2010.

Prerequisites

Before we start using scikit-learn latest release, we require the following −

Python (>=3.5)

NumPy (>= 1.11.0)

Scipy (>= 0.17.0)li

Joblib (>= 0.11)

Matplotlib (>= 1.5.1) is required for Sklearn plotting capabilities.

Pandas (>= 0.18.0) is required for some of the scikit-learn examples using data structure and analysis.

Installation

If you already installed NumPy and Scipy, following are the two easiest ways to install scikit-learn −

Using pip

Following command can be used to install scikit-learn via pip −

pip install -U scikit-learn

Using conda

Following command can be used to install scikit-learn via conda −

conda install scikit-learn

Features Rather than focusing on loading, manipulating and summarising data, Scikit-learn library is focused on modeling the data. Some of the most popular groups of models provided by Sklearn are as follows −

Supervised Learning algorithms − Almost all the popular supervised learning algorithms, like Linear Regression, Support Vector Machine (SVM), Decision Tree etc., are the part of scikit-learn.

Unsupervised Learning algorithms − On the other hand, it also has all the popular unsupervised learning algorithms from clustering, factor analysis, PCA (Principal Component Analysis) to unsupervised neural networks.

Clustering − This model is used for grouping unlabeled data.

Cross Validation − It is used to check the accuracy of supervised models on unseen data.

Dimensionality Reduction − It is used for reducing the number of attributes in data which can be further used for summarisation, visualisation and feature selection.

Ensemble methods − As name suggest, it is used for combining the predictions of multiple supervised models.

Feature extraction − It is used to extract the features from data to define the attributes in image and text data.

Feature selection − It is used to identify useful attributes to create supervised models.

Open Source − It is open source library and also commercially usable under BSD license.

Numpy

NumPy is a library for the python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high level mathematical functions to operate on these arrays. The ancestor of NumPy, Numeric, was originally created by Jim with contributions from several other developers. In 2005, Travis created NumPy by incorporating features of the competing Numarray into Numeric, with extensive modifications. NumPy is open source software and has many contributors.
The Python programming language was not originally designed for numerical computing, but attracted the attention of the scientific and engineering community early on. In 1995 the special interest group (SIG) matrix-sig was founded with the aim of defining an array computing package; among its members was Python designer and maintainer Guido van Rossum, who extended Python's syntax (in particular the indexing syntax) to make array computing easier.

An implementation of a matrix package was completed by Jim Fulton, then generalized[further explanation needed] by Jim Hugunin and called Numeric (also variously known as the "Numerical Python extensions" or "NumPy"). Hugunin, a graduate student at the Massachusetts Institute of Technology (MIT),10 joined the Corporation for National Research Initiatives (CNRI) in 1997 to work on JPython, leaving Paul Dubois of Lawrence Livermore National Laboratory (LLNL) to take over as maintainer. Other early contributors include David Ascher, Konrad Hinsen and Travis Oliphant.:

A new package called Numarray was written as a more flexible replacement for Numeric. Like Numeric, it too is now deprecated. Numarray had faster operations for large arrays, but was slower than Numeric on small ones, so for a time both packages were used in parallel for different use cases. The last version of Numeric (v24.2) was released on 11 November 2005, while the last version of numarray (v1.5.2) was released on 24 August 2006.

There was a desire to get Numeric into the Python standard library, but Guido van Rossum decided that the code was not maintainable in its state then.

In early 2005, NumPy developer Travis Oliphant wanted to unify the community around a single array package and ported Numarray's features to Numeric, releasing the result as NumPy 1.0 in 2006. This new project was part of SciPy. To avoid installing the large SciPy package just to get an array object, this new package was separated and called NumPy. Support for Python 3 was added in 2011 with NumPy version 1.5.0.

Features

NumPy targets the CPython reference implementation of Python, which is a non-optimizing bytecode interpreter. Mathematical algorithms written for this version of Python often run much slower than compiled equivalents due to the absence of compiler optimization. NumPy addresses the slowness problem partly by providing multidimensional arrays and functions and operators that operate efficiently on arrays; using these requires rewriting some code, mostly inner loops, using NumPy.

Using NumPy in Python gives functionality comparable to MATLAB since they are both interpreted, and they both allow the user to write fast programs as long as most operations work on arrays or matrices instead of scalars. In comparison, MATLAB boasts a large number of additional toolboxes, notably Simulink, whereas NumPy is intrinsically integrated with Python, a more modern and complete programming language. Moreover, complementary Python packages are available; SciPy is a library that adds more MATLAB-like functionality and Matplotlib is a plotting package that provides MATLAB-like plotting functionality. Internally, both MATLAB and NumPy rely on BLAS and LAPACK for efficient linear algebra computations.

Installing Numpy

The only prerequisite for installing NumPy is Python itself. If you don’t have Python yet and want the simplest way to get started, we recommend you use the Anaconda Distribution - it includes Python, NumPy, and many other commonly used packages for scientific computing and data science.

CONDA

If you use conda, you can install NumPy from the defaults or conda-forge channels:

```
# Best practice, use an environment rather than install in the base env
conda create -n my-env
conda activate my-env
# If you want to install from conda-forge
conda config --env --add channels conda-forge
# The actual install command
conda install numpy
```

PIP

If you use pip, you can install NumPy with:

```
pip install numpy
```

Librosa

Librosa is a Python package for music and audio analysis. Librosa is basically used when we work with audio data like in music generation(using LSTMs), Automatic Speech Recognition.

Installation instructions

pypi

The simplest way to install librosa is through the Python Package Index (PyPI). This will ensure that all required dependencies are fulfilled. This can be achieved by executing the following command:

```
pip install librosa
```

or:

```
sudo pip install librosa
```

to install system-wide, or:

```
pip install -u librosa
```

to install just for your own user.

conda

If you use conda/Anaconda environments, librosa can be installed from the conda-forge channel:

```
conda install -c conda-forge librosa
```

It provides the building blocks necessary to create the music information retrieval systems. Librosa helps to visualize the audio signals and also do the feature extractions in it using different signal processing techniques.

Source

If you've downloaded the archive manually from the releases page, you can install using the setuptools script:

```
tar xzf librosa-VERSION.tar.gz
cd librosa-VERSION/
python setup.py install
```

If you intend to develop librosa or make changes to the source code, you can install with pip install -e to link to your actively developed source tree:

```
tar xzf librosa-VERSION.tar.gz
cd librosa-VERSION/
pip install -e .
```

Alternately, the latest development version can be installed via pip:

```
pip install git+https://github.com/librosa/librosa
```

ffmpeg

To fuel audioread with more audio-decoding power, you can install ffmpeg which ships with many audio decoders. Note that conda users on Linux and OSX will have this installed by default; Windows users must install ffmpeg separately.

OSX users can use homebrew to install ffmpeg by calling brew install ffmpeg or get a binary version from their website https://www.ffmpeg.org.

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.

Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python. Matplotlib makes easy things easy and hard things possible.

1. Create publication quality plots.
2. Make interactive figures that can zoom, pan, update.
3. Customize visual style and layout.
4. Export to many file formats.
5. Embed in JupyterLab and Graphical User Interfaces.
6. Use a rich array of third-party packages built on Matplotlib.

Seaborn

Seaborn is a Python data visualization library based on matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics. Seaborn is a library in Python predominantly used for making statistical graphics. Seaborn is a data visualization library built on top of matplotlib and closely integrated with pandas data structures in Python. Visualization is the central part of Seaborn which helps in exploration and understanding of data.

Installing and getting started

Official releases of seaborn can be installed from PyPI:

pip install seaborn

The library is also included as part of the Anaconda distribution:

conda install seaborn

Dependencies

Supported Python versions

Python 3.6+

Required dependencies

If not already present, these libraries will be downloaded when you install seaborn.

numpy

scipy

pandas

matplotlib

ANNEXURE-1

SAMPLE CODE:

DATA EXPLORATION:

```
data.boxplot(column='target')
data_target=data.groupby('target')
data['target'].value_counts()


data_ = {'target': data['target'], 'date': data['date']}
df = pd.DataFrame(data_)
df.head()

df['date'] = pd.to_datetime(df['date'])
hour = [ df['date'][i].hour for i in range(len(df['date'])) ]
df['hour'] = hour
df.head()


hour_data = {'0': [0]*24, '2': [0]*24, '4': [0]*24}
for i in range(len(df['hour'])):
 target = str(df['target'][i])
 hour = int(df['hour'][i])
 hour_data[target][hour] += 1
hour_data = [hour_data['0'], hour_data['2'], hour_data['4']]
# Transpose
hour_data = list(map(list,zip(*hour_data)))
```

DATA PREPROCESSING:

```

positif_data = data[data.target==4].iloc[:10000,:]
print(positif_data.shape)
negative_data = data[data.target==0].iloc[:10000,:]
print(negative_data.shape)
data = pd.concat([positif_data,negative_data],axis = 0)
print(data.shape)
data.head()

df = data.copy().sample(8000, random_state=42)
df["label"] = 0
```

```
df = df[['TweetText', 'label']]
df.dropna(inplace=True)
df.head()

df2['label'] = 1
df2 = df2[['text', 'label']]
data = pd.concat([df,df2])
data = data.sample(frac=1)
data.info()

# Removing the twitter handles
data['Clean_TweetText'] = data['TweetText'].str.replace("@", "", regex=True)
# Removing links
data['Clean_TweetText'] = data['Clean_TweetText'].str.replace(r"http\S+", "", regex=True)
# Removing Punctuations, Numbers, and Special Characters
data['Clean_TweetText'] = data['Clean_TweetText'].str.replace("[^a-zA-Z]", " ", regex=True
# Remove stop words
#nltk.download('stopwords')
stop_words=nltk.corpus.stopwords.words('english')
def remove_stopwords(text):
 clean_text=' '.join([word for word in text.split() if word not in stop_words])
 return clean_text
data['Clean_TweetText'] = data['Clean_TweetText'].apply(lambda text : remove_stopwords(str
data.head()

#nltk.download('punkt')
data['Clean_TweetText'] = data['Clean_TweetText'].apply(lambda x: nltk.word_tokenize(x))
data['Clean_TweetText'] = data['Clean_TweetText'].apply(lambda x: ' '.join([w for w in x])
```