

# **Biasness of World Leaders: A Sentiment Analysis of World Leaders' Twitter Data**



By

**Samir Khan**

**Fall-2019-MS-CS 00000319113 SEECS**

Supervisor

**Dr. Shah Khalid**

**Department of Computing**

A thesis submitted in partial fulfillment of the requirements for the degree of Masters  
of Science in Computer Science (MS CS)

In

School of Electrical Engineering & Computer Science (SEECS) ,

National University of Sciences and Technology (NUST),

Islamabad, Pakistan.

(July 2023)

### THESIS ACCEPTANCE CERTIFICATE

Certified that final copy of MS/MPhil thesis entitled "Biasness of World Leaders: A Sentiment Analysis of World Leaders' Twitter Data" written by SAMIR KHAN, (Registration No 00000319113), of SEECs has been vetted by the undersigned, found complete in all respects as per NUST Statutes/Regulations, is free of plagiarism, errors and mistakes and is accepted as partial fulfillment for award of MS/M Phil degree. It is further certified that necessary amendments as pointed out by GEC members of the scholar have also been incorporated in the said thesis.

Signature: \_\_\_\_\_ *Khalid* \_\_\_\_\_

Name of Advisor: \_\_\_\_\_ Dr. Shah Khalid \_\_\_\_\_

Date: \_\_\_\_\_ **04-Jul-2023** \_\_\_\_\_

HoD/Associate Dean: \_\_\_\_\_

Date: \_\_\_\_\_

Signature (Dean/Principal): \_\_\_\_\_

Date: \_\_\_\_\_

## Approval

It is certified that the contents and form of the thesis entitled "Biasness of World Leaders: A Sentiment Analysis of World Leaders' Twitter Data" submitted by SAMIR KHAN have been found satisfactory for the requirement of the degree

Advisor : Dr. Shah Khalid

Signature: Shah Khalid

Date: 04-Jul-2023

Committee Member 1:Mr Bilal Ali

Signature: Bilal Ali

04-Jul-2023

Committee Member 2:Prof. Hasan Ali Khattak

Signature: Hasan Ali Khattak

Date: 04-Jul-2023

Signature: \_\_\_\_\_

Date: \_\_\_\_\_


# Dedication

This thesis is dedicated to me.

## Certificate of Originality

I hereby declare that this submission titled "Biasness of World Leaders: A Sentiment Analysis of World Leaders' Twitter Data" is my own work. To the best of my knowledge it contains no materials previously published or written by another person, nor material which to a substantial extent has been accepted for the award of any degree or diploma at NUST SEECS or at any other educational institute, except where due acknowledgement has been made in the thesis. Any contribution made to the research by others, with whom I have worked at NUST SEECS or elsewhere, is explicitly acknowledged in the thesis. I also declare that the intellectual content of this thesis is the product of my own work, except for the assistance from others in the project's design and conception or in style, presentation and linguistics, which has been acknowledged. I also verified the originality of contents through plagiarism software.

Student Name: SAMIR KHAN

Student Signature: 

# Acknowledgments

In the name of Allah, the most beneficent and merciful. First of all, I would like to thank Allah Almighty for his countless blessings and my parents who were always there for me.

I would like to express my sincere gratitude to Dr. Shah Khalid and Dr. Bilal Ali for supervising, motivating, and guiding me; to Dr. Hasan Ali Khattak for his support and his valuable suggestions.

I would also like to thank Mr. Shaida Muhammad for his support.

**Samir Khan**

# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Motivation . . . . .	3
1.2	Problem Statement . . . . .	3
1.3	Contributions . . . . .	4
1.3.1	Dataset . . . . .	4
1.3.2	Tweets Classification . . . . .	4
1.4	Thesis Organization . . . . .	5
<b>2</b>	<b>Literature Review</b>	<b>6</b>
2.1	Twitter Sentiment Analysis . . . . .	6
2.2	Lexicon Based Sentiment Analysis . . . . .	8
2.3	Machine Learning Based Sentiment Analysis . . . . .	8
2.4	Sentiment Analysis of SDGs . . . . .	9
2.4.1	Summary . . . . .	10
<b>3</b>	<b>Methodology</b>	<b>11</b>
3.1	Machine Learning . . . . .	12
3.1.1	Decision Tree . . . . .	12
3.1.2	Logistic Regression . . . . .	16
3.1.3	K-nearest Neighbor . . . . .	17
3.2	Deep Learning . . . . .	20

## CONTENTS

3.2.1	Feed-forward Neural Network . . . . .	20
3.2.2	Long short-term memory (LSTM) . . . . .	24
3.2.3	Transformers . . . . .	26
<b>4</b>	<b>Dataset</b> . . . . .	<b>33</b>
4.1	World Leaders . . . . .	33
4.2	Dataset Creation . . . . .	34
4.3	Tweets Preprocessing . . . . .	35
4.4	Dataset Labeling . . . . .	37
<b>5</b>	<b>Results &amp; Discussion</b> . . . . .	<b>40</b>
5.1	Experimental Setup . . . . .	40
5.2	Results . . . . .	41
5.2.1	Decision Tree . . . . .	42
5.2.2	Logistic Regression . . . . .	42
5.2.3	K-nearest neighbour . . . . .	42
5.2.4	FFNN . . . . .	43
5.2.5	LSTM . . . . .	44
5.2.6	BERT . . . . .	46
5.3	Summary . . . . .	46
<b>6</b>	<b>Conclusion and Future Work</b> . . . . .	<b>47</b>
6.1	Conclusion . . . . .	47
6.2	Future Work . . . . .	48



# List of Abbreviations and Symbols

## Abbreviations

<b>ML</b>	Machine Learning
<b>DL</b>	Deep Learning
<b>SA</b>	Sentiment Analysis
<b>TSA</b>	Twitter Sentiment Analysis
<b>SDG</b>	Sustainable Development Goal

# List of Figures

1.1	Sustainable Development Goals. Source <a href="https://sdgs.un.org/goals">https://sdgs.un.org/goals</a> . . . . .	2
2.1	Sentiment Analysis Taxonomy . . . . .	7
2.2	Literature comparison . . . . .	10
3.1	Abstract architecture diagram . . . . .	13
3.2	Architecture diagram . . . . .	14
3.3	Technology to layer mapping . . . . .	15
3.4	Log Loss cost function graph . . . . .	17
3.5	K-nearest Neighbor . . . . .	18
3.6	Euclidean Distance . . . . .	19
3.7	Manhattan Distance . . . . .	19
3.8	Gradient Descent . . . . .	23
3.9	Gradient Descent: Small vs Large learning rate . . . . .	23
3.10	LSTM Cell . . . . .	24
3.11	LSTM Network . . . . .	25
3.12	Transformers: Encoder and Decoder . . . . .	27
3.13	Self-attention mechanism . . . . .	28
3.14	Multi-head self-attention . . . . .	29
3.15	How Residual Layers works . . . . .	30
3.16	Transformers: A detailed look . . . . .	30

## LIST OF FIGURES

3.17 BERTbase and BERTlarge . . . . .	31
4.1 Tweets Distribution By World Leaders . . . . .	36
4.2 Before Cleaning . . . . .	37
4.3 After Cleaning . . . . .	37
4.4 Tweets with Labels . . . . .	38
4.5 Tweets with Encoded Labels . . . . .	39
4.6 SDG Distribution . . . . .	39
5.1 SDG Distribution . . . . .	41
5.2 FFNN training loss: 500 epochs . . . . .	43
5.3 FFNN: Train vs Test loss . . . . .	44
5.4 LSTM: Train vs Test loss . . . . .	45
5.5 LSTM: Train vs Test exact accuracy . . . . .	45
5.6 BERT: Train vs Test exact accuracy . . . . .	46

# List of Tables

4.1	World Leaders on Twitter . . . . .	34
5.1	System Information . . . . .	40
5.2	All models results . . . . .	41

# Abstract

The United Nations' Sustainable Development Goals (SDGs) are a set of 17 goals that were adopted in 2015 as a blueprint for a better and more sustainable future. The goals are interconnected and address diverse global challenges such as poverty, inequality, climate change, peace, and justice. World leaders play an integral role in creating awareness and establishing policies that support attaining the goals. Understanding the opinions of world leaders towards the SDGs can help the United Nations (UN) identify areas of support and potential barriers to achieving the goals. Furthermore, it can be used by policymakers to design strategies to accelerate progress toward the SDGs.

Twitter is a widespread social media platform founded in 2006 that allows users to send short messages called "tweets" containing up to 280 characters. It has over 350 million monthly active users as of 2023. Sentiment analysis on Twitter data can help researchers understand public opinions, attitudes, and emotions toward a particular viewpoint or an event. Researchers have been using Twitter data for multiple research purposes such as sentiment analysis due to its popularity and public availability. For instance, stock prices predictions, and public sentiments toward Covid19 outbreak

This study aims to classify world leaders' sentiments on the 17 Sustainable Development Goals (SDGs) using machine learning algorithms. English language Twitter data of world leaders has been collected from January 2015 to February 2023 and then labeled with SDGs ontologies. As per our research, there are no SDGs labeled world leaders' Twitter data datasets publically available. We used a Python-based tool, SNScrape library to fetch world leaders' Twitter data. There are other procedures to label a dataset but we prefer the ontology-based for this study since it is time efficient. This study performs an empirical comparison of different machine learning algorithms and evaluates effectiveness in classifying tweets based on their relevance to the 17 SDGs. The results provide a comprehensive picture of world leaders' views and opinions on the

## LIST OF TABLES

SDGs, as expressed on social media, which can be used to inform decision-making and drive progress toward achieving the SDGs.

## CHAPTER 1

# Introduction

In 2015, the United Nations General Assembly established a set of 17 Sustainable Development Goals (SDGs) to be achieved by 2030, aiming to encourage a more sustainable and prosperous future for humanity. Despite the significance of these goals, there is currently a scarcity of research examining the biasness or sentiment of Twitter users in relation to the SDGs. Given that world leaders and heads of state possess the potential to drive awareness and contribute to the attainment of these goals, it is essential to investigate their biases or sentiments toward the SDGs. Twitter serves as a platform where world leaders express their perspectives, and their tweets can be associated with specific SDGs.

Sustainable Development Goals (SDGs) are a set of 17 interrelated goals, as seen in Figure 1.1 (the image is retrieved from [7]), and 169 targets, they encompass a wide range of economic, social, and environmental dimensions, aiming to eliminate poverty, reduce inequality, promote inclusive economic growth, ensure environmental sustainability, and promote peace and justice. These goals build upon the Millennium Development Goals (MDGs) and incorporate new dimensions such as climate change, sustainable consumption and production, and peace-building. By setting targets to be achieved by 2030, the SDGs provide a roadmap for governments, international organizations, civil society, and other stakeholders to align their efforts and work collectively toward a more sustainable future. These goals also promote a participatory and multi-stakeholder approach, encouraging partnerships and collaboration among governments, businesses, academia, and civil society organizations.

Twitter sentiment analysis (TSA) refers to the examination of bias or sentiment ex-



**Figure 1.1:** Sustainable Development Goals. Source <https://sdgs.un.org/goals>

pressed in tweets towards a specific viewpoint. Given that tweets are concise and limited in length, proper preprocessing techniques are necessary to prepare the data for analysis. This preprocessing stage typically involves tasks such as removing irrelevant information (e.g., URLs, hashtags, and mentions), tokenization to break the text into individual words or tokens, removing stop words, and handling special characters and emoticons. Preprocessing ensures that the tweet data is transformed into a suitable format for subsequent sentiment analysis algorithms. Sentiment analysis is a field within natural language processing that involves classification of text into various categories based on the sentiments and opinions expressed by users. A sentiment or biasness can be quantified as positive or negative. Text data for sentiment analysis is commonly collected from social media platforms such as Twitter, Reddit, and others, where users express their opinions on diverse topics. These platforms attract individuals from various domains, including politics, business, sports, technology, and education, who utilize social networks to share their views and publish news. Twitter in particular, is a widespread social media platform founded in 2006 that allows users to send short messages called "tweets" containing up to 280 characters<sup>1</sup>. It has over 350 million monthly active users

<sup>1</sup><https://developer.twitter.com/en/docs/counting-characters>



as of 2023<sup>2</sup>. Sentiment analysis on Twitter data can help researchers understand public opinions, attitudes, and emotions toward a particular viewpoint or an event. Researchers have been using Twitter data for multiple research purposes such as sentiment analysis due to its popularity and public availability.

## 1.1 Motivation

The sentiment and biasness of world leaders have the potential to significantly influence public opinion and shape policy decisions. [22] In the context of social media platforms like Twitter, world leaders utilize these platforms to communicate their perspectives, initiatives, and stances on various issues such as Sustainable Development Goals (SDGs). However, there is a lack of comprehensive understanding regarding the sentiment and biasness exhibited by world leaders through their Twitter data. The analysis of world leaders' Twitter data can provide valuable insights concerning the SDGs is of crucial importance for the United Nations (UN) as it enables the identification of areas requiring support and potential barriers that may hinder the achievement of these goals. Such understanding can inform policymakers in the development of targeted strategies aimed at accelerating progress toward the SDGs

## 1.2 Problem Statement

The sentiments expressed by world leaders have the potential to influence policy-making processes about the Sustainable Development Goals (SDGs). However, there exists a significant knowledge gap in comprehensively understanding the sentiments of world leaders, particularly those that are publicly expressed. In this study, we leverage Twitter, one of the prominent social media platforms, to gather data from world leaders and conduct sentiment analysis specifically related to the SDGs. By analyzing their sentiments, we aim to determine any biasness exhibited by world leaders towards the SDGs.

---

<sup>2</sup><https://www.statista.com/statistics/303681/twitter-users-worldwide>

## 1.3 Contributions

This thesis makes two significant contributions.

### 1.3.1 Dataset

In this study, we collected Twitter data from world leaders using a Python library - SNScrape <sup>3</sup>, and subsequently processed the data to remove unnecessary information, including URLs and emojis, utilizing another library. More about dataset creation and labeling will be discussed in the Dataset chapter.

### 1.3.2 Tweets Classification

To classify the tweets based on the 17 Sustainable Development Goals (SDGs), we trained various machine learning and deep learning models, including Logistic Regression, K-Nearest Neighbor, Decision Tree, Feedforward Neural Network, Long Short-Term Memory (LSTM) [4], and Bidi-Directional Encoder Representations from Transformers (BERT) [11].

For the Logistic Regression classifier, we trained 17 binary classifiers, each corresponding to a specific SDG category. The classifiers output probabilities indicating the association between the tweet and the particular SDG.

The Decision Tree model employed the Gini algorithm [3] to construct the tree by iteratively splitting the data based on the input features' embeddings. The leaf nodes determined the assigned categories for the input features, with GloVe embeddings used for this model. The K-Nearest Neighbor (KNN) model calculated distances between the tweets and the training examples, predicting the categories based on the closest training tweets using a voting system.

The Feedforward Neural Network model learned a complex function between the input features and the output categories. Training involved the backpropagation algorithm and the Adam optimizer to update the weights and minimize the model's loss function.

The LSTM model processed input text sequentially, word by word, taking into account grammatical and structural information. It utilizes the Adam optimizer and backpropa-

---

<sup>3</sup><https://github.com/JustAnotherArchivist/snscape>

gation algorithm to obtain optimal weights. BERT, a transformer-based model, captured word dependencies using self-attention mechanisms. It processed the input in parallel while preserving structural information through positional embeddings.

To evaluate the models, we employed four different metrics: exact accuracy, macro average accuracy, macro average precision, and macro average recall. These metrics provided insights into the performance of the models in accurately classifying the tweets based on the SDGs.

## 1.4 Thesis Organization

The thesis document is divided into the following six chapters.

1. Introduction: Provides motivation, problem statement, and contributions.
2. Literature Review: Summarizes recent related work.
3. Methodology: Explains machine learning and deep learning models.
4. Dataset Creation and Labeling: Describes data collection and labeling process.
5. Results and Discussion: Presents implementation details and experimental results.
6. Conclusion and Future Work: Concludes the thesis and suggests future research.

# Literature Review

The literature review for this study is structured into the following four sections.

## 2.1 Twitter Sentiment Analysis

Due to the abundance of easily accessible public data on Twitter, researchers have been using Twitter data for a wide range of analyses. For instance, researchers collected 16 million tweets between September and October of 2019 to analyze the impact of Brexit (a portmanteau of "British exit") on Britain currency exchanges and stock markets [19]. The team used Latent Dirichlet Allocation (LDA) based topic modeling along with the VADER Python library to perform sentiment analysis. Latent Dirichlet Allocation is one of the most commonly used topic modeling techniques. It is an unsupervised machine learning-based approach in nature. The team used human judgment to test model performance. Another group of researchers collected 1.6 million tweets to analyze user opinions about US Presidential Election 2020[24]. They collected the data using Twitter API and then used Bidirectional Long Short-Term Memory (BiLSTM) to perform sentiment analysis. In this paper [9] biasness of talk show hosts is identified regarding their response to Panama Leaks in Pakistan. The dataset was created via the Twitter Streaming API of Pakistani top 5 television shows' hosts. Researchers used Support Vector Machine and Naïve Bayes and got 71.4% and 65.6% accuracy on both models, respectively. [23] used Twitter data to analyze mental diseases i.e. depression, anxiety, bipolar disorder, ADHD, and PTSD. Twitter Sentiment Analysis (TSA) has also been used for a variety of case studies to better understand product feedback and brand loy-

ality ([10] [21]). These datasets are used in most of the recent studies to study product feedback.

- Stanford Sentiment140 [2] - 1.6 million tweets
- US Airline [6] - 14,640 tweets
- IMDB Movie Reviews - 25,000 tweets
- Stanford's IMDB Movie Reviews dataset - 50,000 tweets
- Cornell Movie Reviews [5] - 10,662 tweets

Figure 2.1 shows different techniques currently used for sentiment analysis.

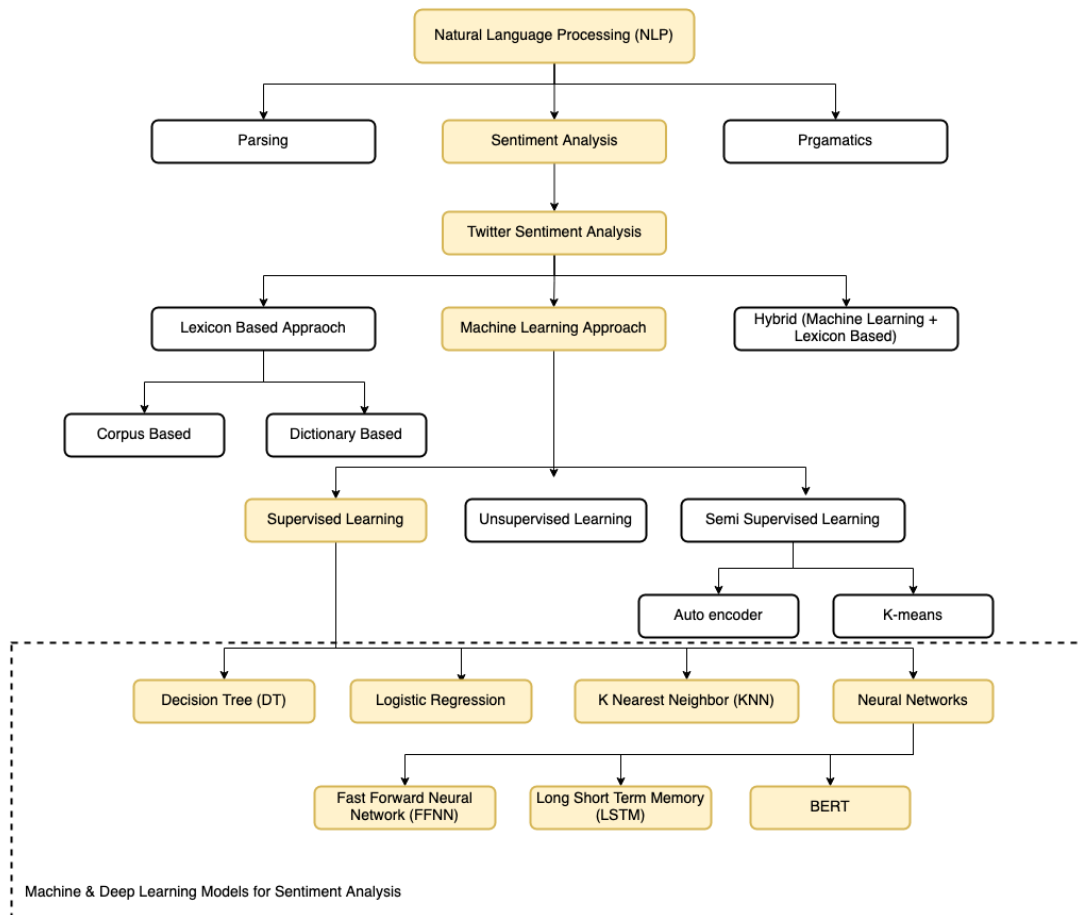


Figure 2.1: Sentiment Analysis Taxonomy

## 2.2 Lexicon Based Sentiment Analysis

The lexicon-based sentiment analysis is conducted with a help of a word dictionary where each word is annotated with polarity score - semantic orientation. The dictionary can be manually and automatically created from seed words, which are mostly adjectives. Seed words are a set of words with strong polarity. For the sake of discussion, we can assume that in lexicon-based sentiment analysis, the text is represented as a bag of words. Polarity is assigned to each word and phrase in the text and then a combining function is used to find the sentiment of the whole text. One of the main advantages of lexicon-based sentiment analysis is, one doesn't need a dataset to perform it. This type of sentiment analysis cannot perform well for novel vocabulary and contextual information. One of the most commonly used lexicons is SentiWordNet [1], it has 6300 words and each word has a score from -100 to +100, representing the positive to negative spectrum.

This [14] study has used lexicon-based sentiment analysis to examine people's feelings about Diabetes. The researchers collected 67421 tweets and used SentiStrength, a lexicon-based open-source software to detect the polarity of tweets. Emojis - small digital images for expressions have not been removed in pre-processing. The sentiment analysis tool detected polarity in tweets text and emojis separately. The emojis are given high weightage, such that a positive tweet with negative emoji will be considered a negative sentiment.

## 2.3 Machine Learning Based Sentiment Analysis

The machine learning-based sentiment analysis starts with feature extraction and feature representation using known algorithms. Machine learning algorithms can be broadly categorized as supervised, unsupervised, and semi-supervised. Supervised learning is generally used for classification tasks and requires a labeled dataset. Some of the commonly used supervised machine learning algorithms are SVM, Naive Bayes Maximum entropy (NB-ME), K Nearest neighbor (KNN), and decision tree (DT). Creating a labeled dataset can be a tiresome task and unsupervised learning techniques resolve this issue by clustering similar text for classification tasks. Semi-supervised learning is the hybrid form of supervised and unsupervised learning techniques. A comprehensive study

[8] has been done on sentiment analysis using machine learning techniques where the researchers compare the performance of most commonly used classifiers on multiple datasets. One of the key challenges the researchers have observed is finding or creating a labeled dataset. Creating such datasets can be expensive and time-consuming. That is why researchers have moved to unsupervised learning-based approaches.

Recently, researchers have been performing sentiment analysis using deep learning models. Deep learning models are different in nature than machine learning models in such a way that they have more inner layers. Some of the most used deep learning models are Recurrent Neural networks (RNN), Convolutional Neural Networks (CNN), Long Short-Term memory networks (LSTM), and transformer-based models. Term Frequency, Inverse Document Frequency (TF-IDF), and word embedding are used as input features for Natural Language Processing (NLP) based deep learning model tasks. Several studies, for instance, this [12] have shown that deep learning models outperform the traditional machine learning models for example Naive Bayes and Support Vector Machines. The researchers experimented with two different datasets and the results indicate that context knowledge is important in sentiment analysis because it gives a deep understanding of the problem. Another comparative study on deep learning models for sentiment analysis is confirming the same claim [18]. The study reviews the latest research papers on sentiment analysis using deep learning models (They are 32 in number). The researchers collected the eight most commonly used datasets for sentiment analysis and used them to train DNN, CNN, and RNN models - the most used deep learning models. They also used different preprocessing methods for preparing inputs to study their influence on the results: namely, word embedding and TF-IDF. The comparison is not straightforward and depends on preprocessing and datasets but the accuracy has been reported higher than those of traditional machine learning models.

## 2.4 Sentiment Analysis of SDGs

A study conducted by A. Ramesh [26] analysis of tweets from Indian politicians focusing on SDG 16, which pertains to peace and harmony, was conducted. The study used the BERT model to analyze tweets from eight prominent political leaders, achieving an accuracy of 0.99%. The data collection process utilized the Tweepy library, resulting in a total dataset of 504 tweets. However, one limitation of the study is the relatively

small size of the dataset, which can impact the generalizability of the findings.

TITLE	DATASET	MODELS	ACCURACY	SHORTFALLS
Sentiment analysis on Twitter data towards climate action	Created their own. 3 million	BERT, lexicon-based	69%	Only targets SDG-7, SDG-11, and SDG-13
DG 2030: Analysis of Political Tweets Using Deep Learning Approach	Tweets of 6 Indian political leaders between 8 June and 8 July 2022	BERT	99%	Small dataset

**Figure 2.2:** Literature comparison

This study [27] aimed to gain insights into the progress of SDGs, particularly regarding climate-related sentiment. A large-scale dataset containing 3 million live tweets was collected using the Twitter Open API. The data collection process involved searching for 47 keywords specifically related to three Sustainable Development Goals (SDGs): SDG-7, SDG-11, and SDG-13. A small subset of the dataset was manually labeled as either negative or positive sentiment. To analyze the sentiment of the collected tweets, pre-trained BERT, lexicon-based, and NLP techniques were used. The evaluation of these models delivered an accuracy of 69%.

### 2.4.1 Summary

In this chapter, we conducted a comprehensive literature review focusing on three main areas: sentiment analysis, sentiment analysis using Twitter data, and Twitter sentiment analysis on SDGs. We examined recent studies, their findings, limitations, and highlighted the unique aspects of our research in comparison to existing work. Additionally, we explored the taxonomy of sentiment analysis in detail. The next chapter will discuss details of the machine learning and deep learning models that will be employed for training on the dataset.



## CHAPTER 3

# Methodology

The methodology of the thesis involved several steps. Firstly, a dataset was created by collecting tweets from world leaders on Twitter. These tweets were gathered using the SNScrape Python library. Once the dataset was collected, it was labeled using SDGs (Sustainable Development Goals) ontology. The SDGs provide a comprehensive framework for addressing global challenges and achieving sustainable development. The ontology consists of a set of predefined keywords or concepts related to each SDG, allowing for the classification and labeling of tweets based on their relevance to specific SDGs. More on this is discussed in Chapter 4. After labeling the dataset, machine learning and deep learning models were trained using various algorithms. These included decision trees, K-nearest neighbors (KNN), logistic regression, BERT (Bidirectional Encoder Representations from Transformers), LSTM (Long Short-Term Memory), and a feed-forward neural network. Each model was trained using the labeled dataset. The models were optimized through parameter tuning and cross-validation techniques to improve their performance and generalization abilities. The trained models were then evaluated using appropriate evaluation metrics such as accuracy, precision, and recall. These metrics provided insights into the effectiveness of each model in classifying tweets based on their alignment with specific SDGs. More on this is discussed in Chapter 5.

The layered architecture, depicted in figure 3.1, comprises three primary layers: the application layer, business logic layer, and data management layer. A more detailed representation of this architecture can be found in figure 3.2. The architecture diagram is further categorized into two sections: the offline process and the online process. The offline process involves dataset creation and labeling, while the online process is utilized

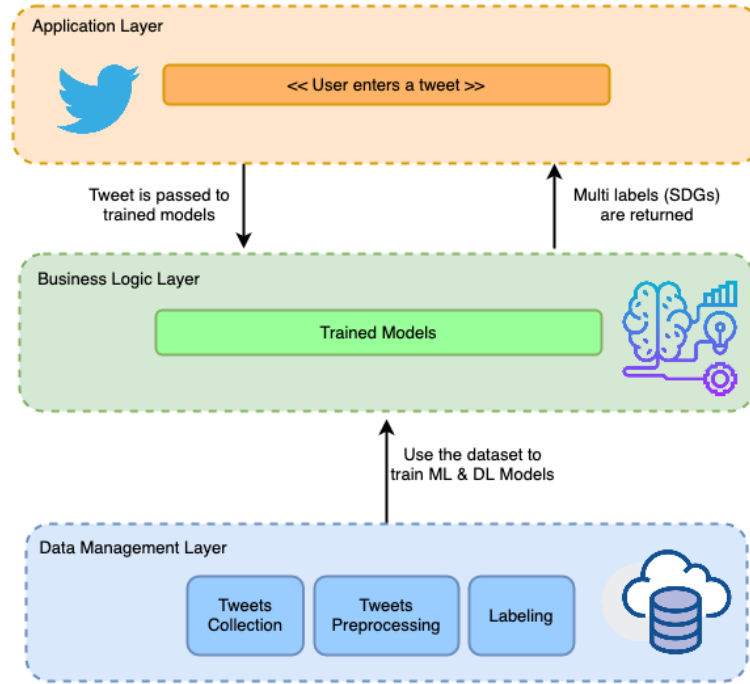
to analyze sentiments in social media data using the trained models. Within the application layer, three components are present. The Social API Linker establishes connections with social media platforms to fetch relevant data. The Output Dispatcher component displays meaningful outputs, such as tweet labels. The Social Media Loader component serves a similar purpose to the Social API Linker, but is specifically designed to retrieve smaller amounts of recent data compared to the latter, which can fetch large volumes of data. The data management layer is responsible for dataset creation and labeling. The Social Media Data component gathers data from Twitter and passes it to subsequent components for Twitter preprocessing and labeling, resulting in a labeled dataset. This dataset is then stored locally or in Google Drive for training the models. In the Business Logic Layer, the Data Loader fetches data from Google Drive or local storage, and the Data Encoder encodes the data in a format suitable for machine learning and deep learning models. The Real-time Data Loader module retrieves data from social networks to analyze sentiments related to SDGs using the Social Media Loader component. The Stream Manipulator component removes hashtags and URLs from the text data, followed by selecting an appropriate model through the Model Storage component to determine the sentiment of the text. Figure 3.3 illustrates the mapping of technologies into the layered architecture.

## 3.1 Machine Learning

We used three famous algorithms on our dataset: **Logistic Regression (LR)**, **K-nearest Neighbor (KNN)**, **Decision Tree (DT)**.

### 3.1.1 Decision Tree

Decision Trees (DTs) are widely used algorithms for regression and classification problems. DTs are non-parametric algorithms. DTs work by learning some simple rules from the data and classifying new examples based on those rules. The simplicity of Decision Trees algorithms makes them easy to interpret and visualize. Logistic Regression and other parametric machine learning algorithms require data to be normalized to work well. DTs don't need any normalization. DTs are also capable of performing multi-label classification. A generic problem with the DTs is overfitting when the tree becomes over-complex.



**Figure 3.1:** Abstract architecture diagram

There are various Tree algorithms to be used. We used CART (Classification and Regression Trees) which uses gini impurity.

**Algorithm 3.1.** *Training data:*  $\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \dots, (x^{(n)}, y^{(n)})\}$  *Trained decision tree model Calculate GloVe word embedding of all words Calculate tweets embedding by averaging word embedding of text Split dataset for training and testing 85:15 CART CART FnFunction: data all samples in data have the same label a leaf node with label as the common class stopping criterion is met a leaf node with the most frequent label in data Find the best split attribute and value based on Gini impurity Split data into two subsets:  $data_{left}$  and  $data_{right}$  Create a decision node with the split attribute and value Set the left child of the decision node as  $CART(data_{left})$  Set the right child of the decision node as  $CART(data_{right})$  the decision node*

*TrainCART TrainCART FnFunction: training\_data training\_data*

$\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \dots, (x^{(n)}, y^{(n)})\}$

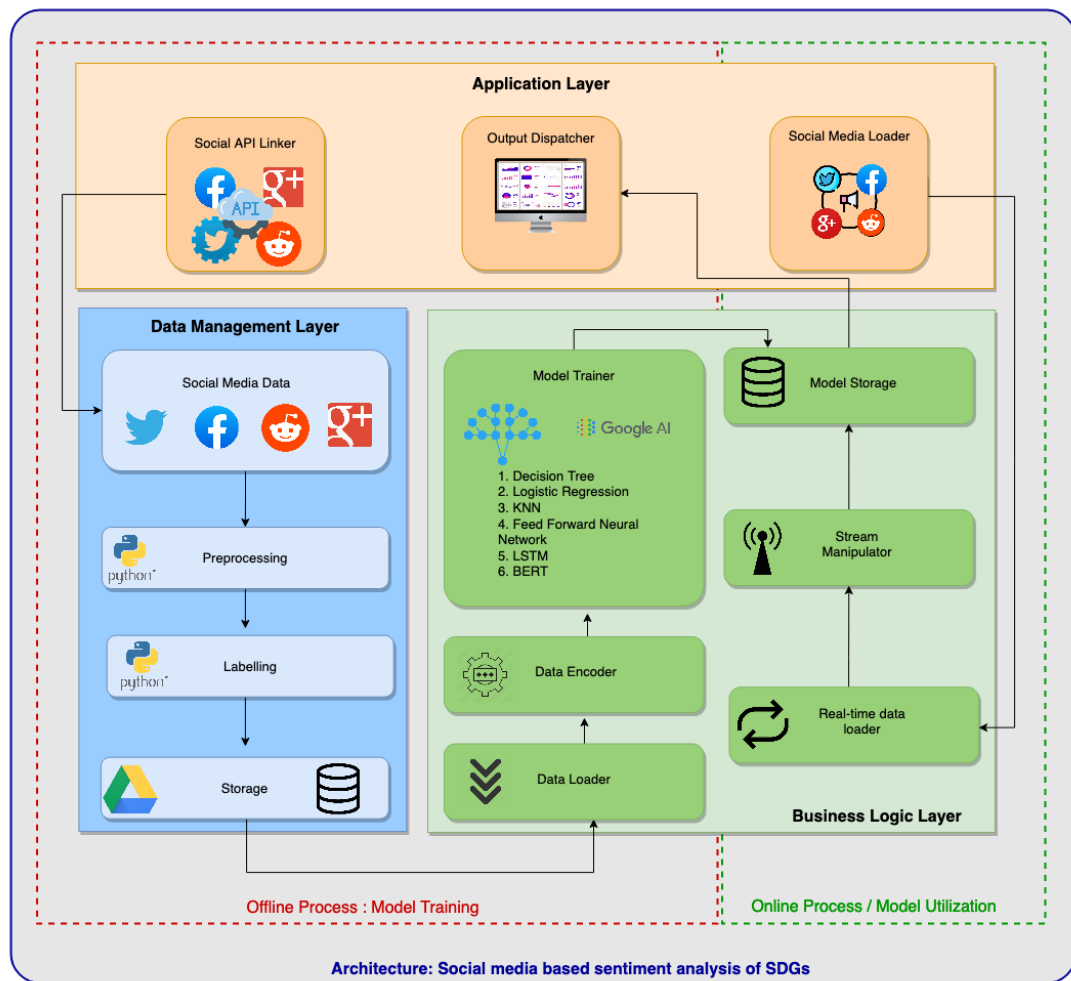


Figure 3.2: Architecture diagram

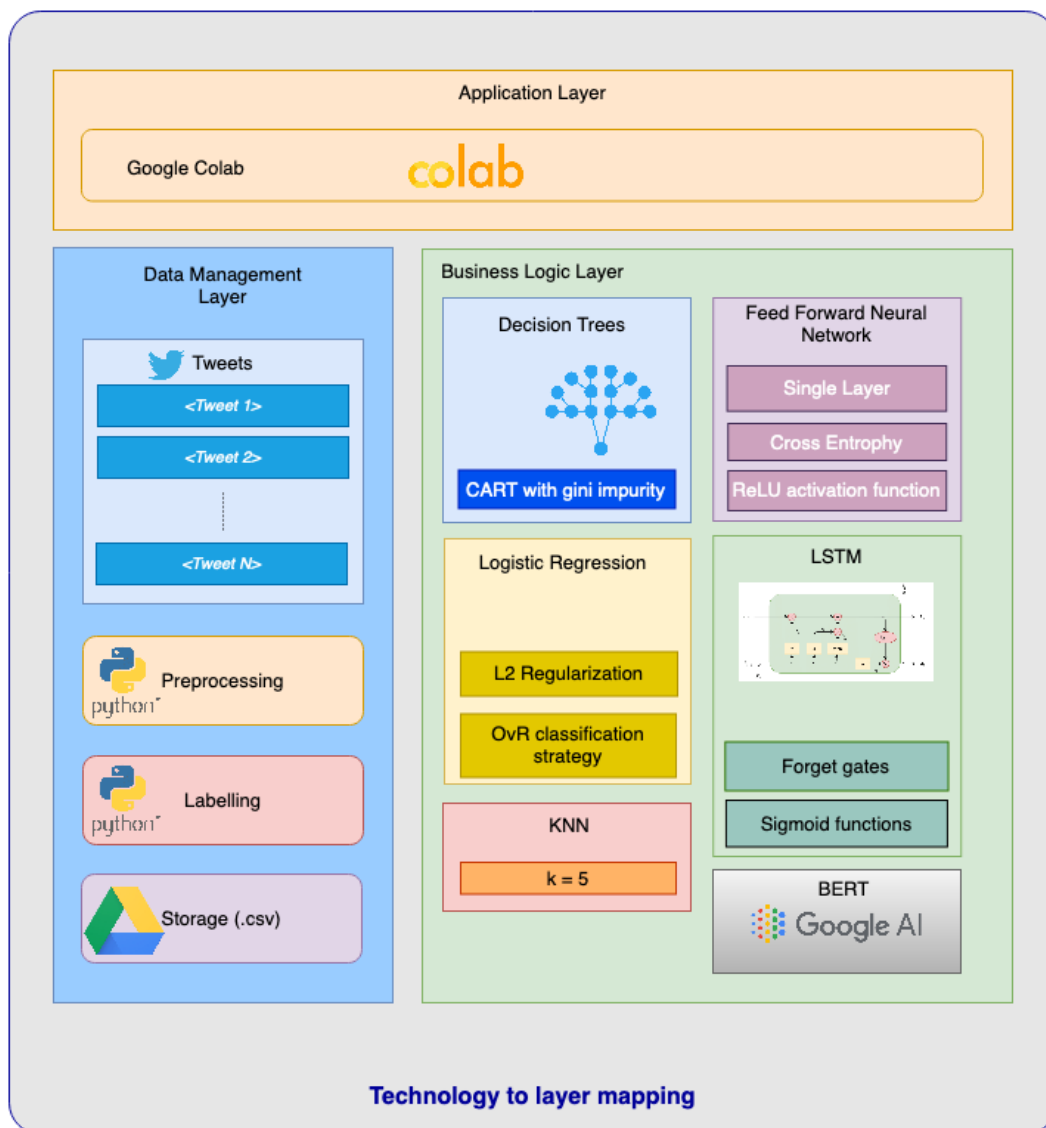


Figure 3.3: Technology to layer mapping

### 3.1.2 Logistic Regression

LR is a linear classifier that uses a logistic function on top of Linear Regression.

$$LR(z) = \frac{1}{1 + e^{-z}} \quad (3.1.1)$$

Equation 3.2.2 is referred to as the sigmoid function. In 3.2.2,  $e$  is Euler's constant whose value is 2.71828.  $z$  is the output of linear regression function from equation 3.1.2. The output of a sigmoid function is a probability distribution whose values are between 0 and 1.

$$z = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_3 + \dots + \beta_m X_m \quad (3.1.2)$$

$\beta$  are the parameters are linear the linear function while  $X$  are the features in the input data. Logistic Regression utilizes the Log Loss function to compute the cost.

$$\text{Log Loss} = \frac{1}{m} \sum_{(x,y) \in D} -y \log(y') - (1 - y) \log(1 - y')$$

**Algorithm 3.2.** *Training data  $\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \dots, (x^{(n)}, y^{(n)})\}$  Trained logistic regression parameters  $\theta$*

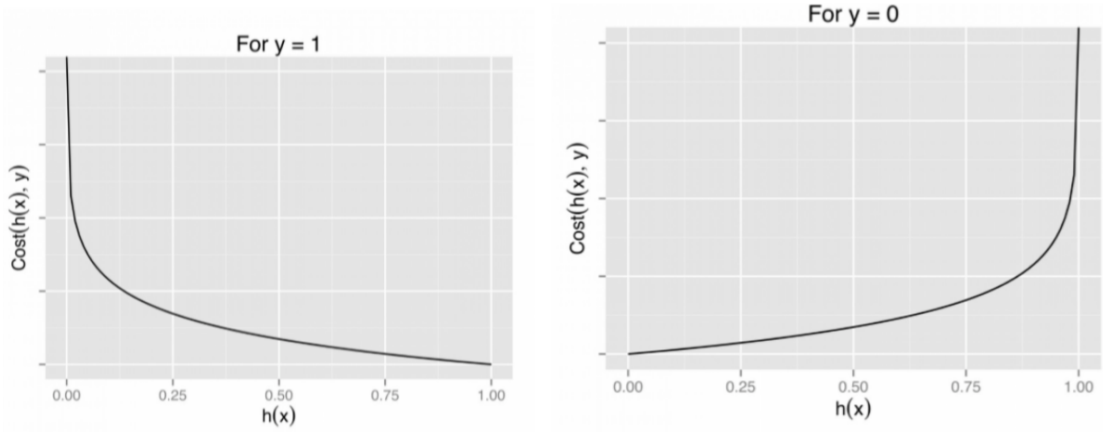
*Dataset preprocessing* Tokenize text with gensim tokenizer Calculate GloVe word embedding of all words Calculate tweets embedding by averaging word embedding of text Split dataset for training and testing 85:15 Initialize parameters  $\theta$  randomly or with zeros Set the learning rate  $\alpha$  Set the number of iterations  $N$

$i = 1$  to  $N$  Compute the hypothesis:  $h_\theta(x) = \frac{1}{1 + e^{-\theta^T x}}$  Compute the cost function:  $J(\theta) = -\frac{1}{n} \sum_{i=1}^n (y^{(i)} \log(h_\theta(x^{(i)})) + (1 - y^{(i)}) \log(1 - h_\theta(x^{(i)})))$  Compute the gradients:  $\frac{\partial J(\theta)}{\partial \theta_j} = \frac{1}{n} \sum_{i=1}^n (h_\theta(x^{(i)}) - y^{(i)}) \cdot x_j^{(i)}$  Update parameters:  $\theta_j = \theta_j - \alpha \cdot \frac{\partial J(\theta)}{\partial \theta_j}$  for all  $j$

*trained parameters  $\theta$*

LR is a binary classifier that distinguished between two classes. The loss function calculates the distance between actual labels in data and the prediction of the model. The prediction of the model is a value between 0 and 1. The closer prediction gets to the actual value, the lower the cost gets, and vice versa. see figure 3.4

LR is a binary classification algorithm i.e. it can be used to classify objects in two classes only. We are dealing with multi-class problems. Our problem consists of 17



**Figure 3.4:** Log Loss cost function graph

mutually exclusive classes which makes it a multi-label classification problem. We use the one-vs-rest (OvR) strategy to make the algorithm capable of dealing with multi-label classification problems. We use the L2 regularization penalty to overcome the overfitting problem. The loss function becomes 3.1.3 after adding the L2 regularization penalty.

$$\text{Loss regularized} = \left( \frac{1}{m} \sum_{(x,y) \in D} -y \log(y') - (1 - y) \log(1 - y') \right) + \frac{\lambda}{2m} \sum_{i=1} \beta_i \quad (3.1.3)$$

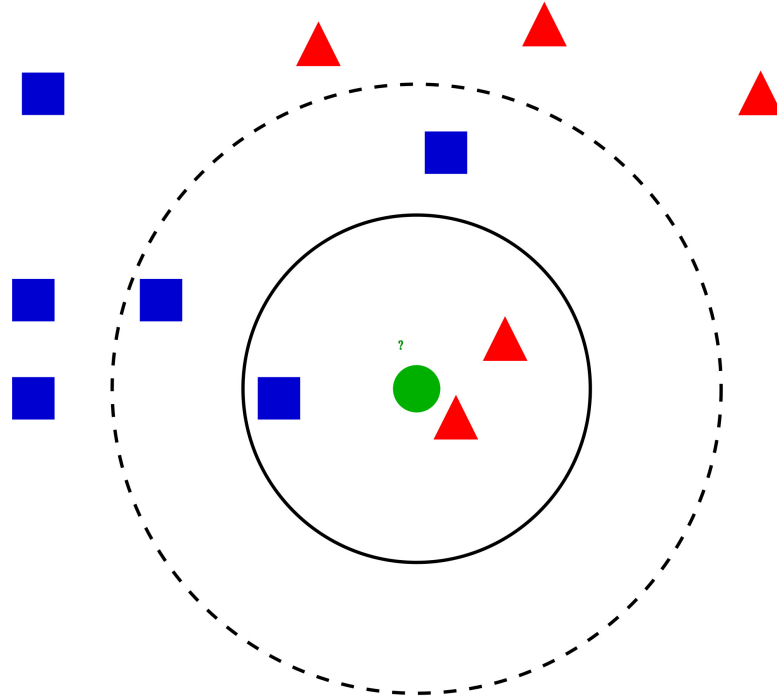
The regularized loss function is used to train the model.

### 3.1.3 K-nearest Neighbor

K-nearest Neighbor (KNN) is another widely used machine learning algorithm mainly used for classification problems. KNN works by calculating the distance between test data points and all training data points. KNN decides the class of a data point by looking at its nearest point. The number of closest points to look at is called  $K$  and is decided before starting the algorithm. The value of  $K$  is a hyperparameter and its optimum value can be calculated by the hit-and-trial method.

Keeping the value of  $K$  very small tends to be under-fitting. The higher value of  $K$  tends to overfit. Thus an optimum value is required. If the value of  $K$  is selected as  $n$  then the class of the data point is decided by majority vote among the classes of nearest  $n$  points.

In 3.5, the green point needs to be classified. The number of  $K$  is the radius of the



**Figure 3.5:** K-nearest Neighbor

circle. If we take the value of  $k$  as 3 then the point is classified as a triangle based on majority votes. If the value of  $k$  is selected as 5, then the predicted class for the data point is square.

Various distance functions can be used to calculate the distance. Two widely used distance functions with KNN are Euclidean Distance [3.1.4](#) and Manhattan Distance [3.1.5](#).

$$d(p, q) = \sqrt{\sum_{i=1}^n (q_i - p_i)^2} \quad (3.1.4)$$

The Euclidean Distance finds the straight distance between two points  $p$  and  $q$ . See [figure 3.6](#).

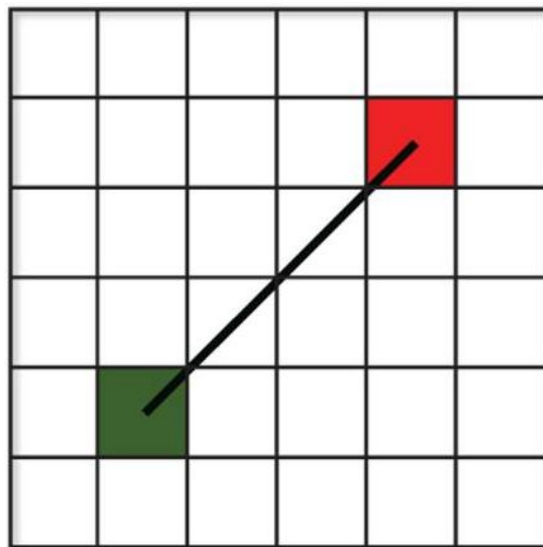
The Manhattan Distance can be calculated by equation [3.1.5](#)

$$d(p, q) = \sum_{i=1}^n |q_i - p_i| \quad (3.1.5)$$

The Manhattan Distance block-wise distance between two points  $p$  and  $q$ : see [figure 3.7](#).

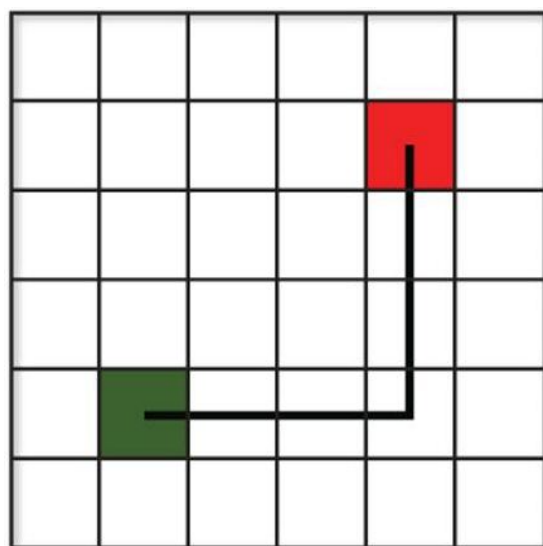
There is no training process in the KNN. The KNN just remembers all training examples





Euclidean Distance

**Figure 3.6:** Euclidean Distance



Manhattan Distance

**Figure 3.7:** Manhattan Distance

and predict new example by calculating the distance between training examples and new example.

**Algorithm 3.3.** *Training data:  $\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \dots, (x^{(n)}, y^{(n)})\}$  Unlabeled test data:  $x_{test}$  Predicted class label for  $x_{test}$ :  $\hat{y}_{test}$*

*$i = 1$  to  $n$  Compute the distance between  $x^{(i)}$  and  $x_{test}$*

*Sort the distances in ascending order and select the top 5 nearest neighbors Count the frequency of each class label among the 5 nearest neighbors  $\hat{y}_{test} =$  the class label with the highest frequency*

## 3.2 Deep Learning

Deep learning methods are well-suited algorithms for solving complex supervised and unsupervised problems. In supervised learning, deep learning models learn a mapping function from input data to the labels by minimizing some objective function. Deep learning models achieve it by iterating through data. Mathematically speaking, it becomes an optimization problem.

We apply different deep learning techniques to our dataset including feed-forward neural network, LSTM, and Transformer based state of the art models.

### 3.2.1 Feed-forward Neural Network

Feed-forward neural network (FFNN) is a parametric type machine learning algorithm. In supervised machine learning, FFNN learns a complex mathematical mapping from input data to data labels. In an unsupervised setup, the algorithm tries to learn the underlying data distribution of input data. FFNN performs this learning by continuously optimizing a set of weights  $W$  in the network.

FFNN consists of two passes: forward pass and backward pass. In the forward pass, the input data is fed as a real value vector to the input layer of the FFNN.

$$a^{(l)} = \sigma(\mathbf{W}a^{l-1} + \mathbf{b}) \quad (3.2.1)$$

In equation 3.2.1, the  $W$  represents the weights/parameters of the network,  $b$  is bias and  $a$  represents the activation of neurons.  $l$  represent the current layer and  $\sigma$  represents

the activation function. In the first layer (input layer), the activations  $a$  becomes the data  $x$ . In the remaining layers, the activation  $a$  of the previous layer is multiplied by the weight matrix of the current layer and added with the bias of the current layer.

**Algorithm 3.4.** *Data:  $\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \dots, (x^{(n)}, y^{(n)})\}$  Trained FFNN model parameters*

*Initialize the weights and biases of the FFNN randomly*

*epoch = 1 to num\_epochs  $(x^{(i)}, y^{(i)})$  in training\_data Compute the activations of hidden layers and output layer using current parameters Compute the loss or cost function based on the prediction and true label Update the weights and biases using backpropagation and gradient descent*

Tons of activation functions are proposed in machine learning literature. These activation functions are problem-specific and layer-specific in the network. For example, ReLU 3.2.4 is used in all layers except the final layer instead of sigmoid 3.2.2 and tanh 3.2.3 function because empirically ReLU works really well on the middle layers. The activation function of the final layer is task specific. For example, for binary classification tasks, sigmoid is preferred, or sometimes there is no activation function at all on the final layer.

$$\text{sigmoid}(z) = \frac{1}{1 + e^{-z}} \quad (3.2.2)$$

$$\text{tanh}(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}} = \frac{1 - e^{-2z}}{1 + e^{-2z}} \quad (3.2.3)$$

$$\text{ReLU}(z) = \max(0, z) \quad (3.2.4)$$

In the forward pass, the last step is calculating the loss of the network. The loss is the difference between the actual labels and the predicted labels. Loss functions are also task-specific. A common choice for regression problems is Mean Square Error (MSE) 3.2.5.

$$MSE = \frac{1}{n} \sum_{i=1}^n (\hat{y}_i - y_i)^2 \quad (3.2.5)$$

In equation 3.2.5,  $n$  represents a number of examples in the batch,  $\hat{y}$  is the prediction and  $y$  is the actual class label. And a common choice for classification tasks is Cross Entropy Loss 3.2.6.

$$CEL = - \sum_{i=0}^C y_i \cdot \log(\hat{y}_i) \quad (3.2.6)$$

The cross-entropy loss function calculates the difference between two probability distributions: the actual class labels and the predicted class probabilities for each class. The predicted class probabilities are calculated by applying a softmax function on the output of the network.

$$\sigma(z_i) = \frac{e^{z_i}}{\sum_{j=1}^K e^{z_j}} \quad \text{for } i = 1, 2, \dots, K \quad (3.2.7)$$

$e$  is Euler's constant and  $K$  represent the number of classes. The softmax function calculates the probability distribution of all classes for each example. The second phase of the FFNN is the backpropagation algorithm. The backpropagation algorithm finds the contribution of error of each weight and bias parameter. The backpropagation combined with an optimization algorithm like gradient descent, Adam, etc., updates the weights of the network by backpropagating the error through the network, starting from the cost/loss function till the weights of the first layer.

There are various optimization algorithms available. The simplest one is gradient descent 3.2.8.

$$\begin{aligned} w_{new} &= w - \alpha \frac{\delta L}{w} \\ b_{new} &= b - \alpha \frac{\delta L}{b} \end{aligned} \quad (3.2.8)$$

In gradient descent, updated weights  $w_{new}$  are obtained by subtracting the error contribution of the weight  $w$  to loss. The error rate of the weight  $w$  is also multiplied by the learning rate  $\alpha$  which caused the weight to be updated gradually toward its optimum value.

The  $\alpha$  decides the step size of the network toward the optimal weights: see figure 3.8. The value of the  $\alpha$  should be chosen very carefully as a large value causes the network to miss the optimum value and a small value slower down the convergence: see figure 3.9.

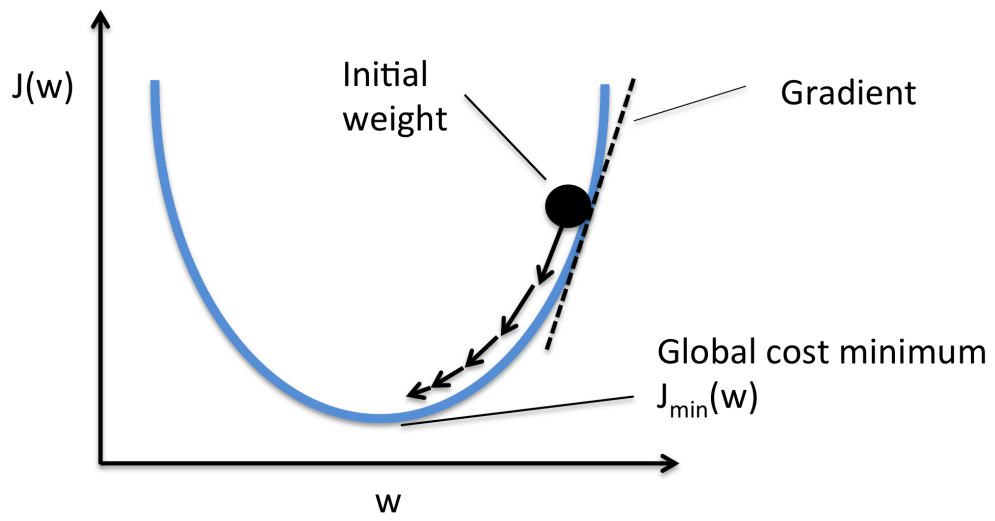


Figure 3.8: Gradient Descent

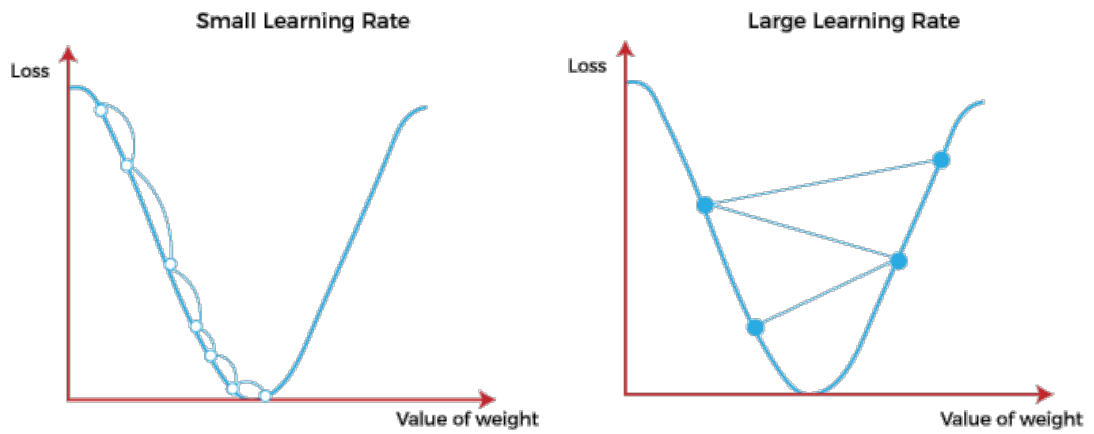


Figure 3.9: Gradient Descent: Small vs Large learning rate

The FFNN can contain any arbitrary number of layers. More layers mean more complex functions to learn. But as the number of layers increases, the network faces a vanishing gradient problem. When the network faces a vanishing gradient, the weights in the earlier layer are not updated and the network does not optimize. If we keep fewer number of layers in the network, the model learns a very simple function which may not be the ideal model for the problem. choosing the optimal number of layers and number of neurons in each layer are difficult tasks and they can be achieved by trial and error method.

### 3.2.2 Long short-term memory (LSTM)

LSTM is a type of Recurrent Neural Network (RNN) where the network uses memory cells to retain long-term information. Even though RNNs are designed to capture previous context and dependency but simple RNNs can't remember long-term information. It loses information as the size of the input increases. The LSTM tries to tackle that issue by introducing memory cells. The memory cells work really well to capture all previous dependencies. As the input size increases in simple RNNs, the network also faces vanishing and exploding gradient problems. LSTM is also good at avoiding that issue up to some context.

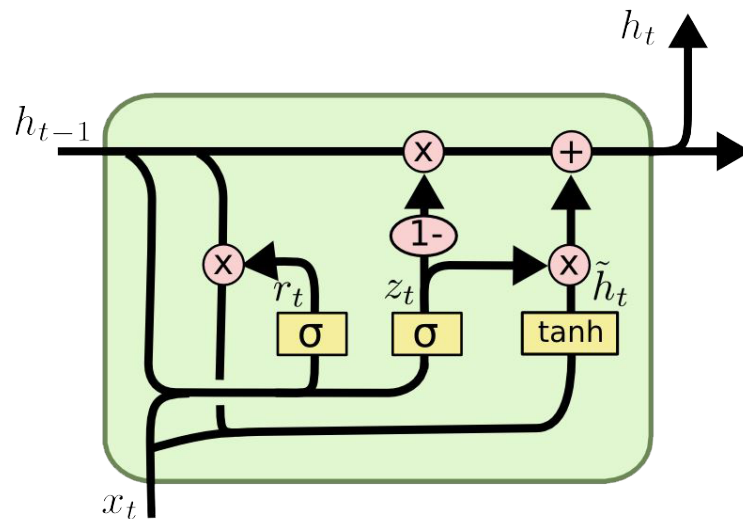


Figure 3.10: LSTM Cell

An LSTM network is made up of multiple LSTM cells [3.10](#). An LSTM cell takes information from the previous LSTM cell, performs some operations, and forwards it to

the next LSTM cell in the network: see figure 3.11.

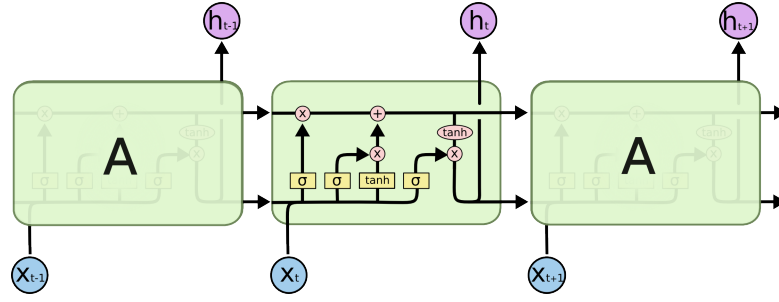


Figure 3.11: LSTM Network

An LSTM cell is made up of multiple components called gates. The gates are used to add/remove/update information in the LSTM cell.

$$\begin{aligned}
 f_t &= \sigma(W_f \bullet [h_{t-1}, x_t] + b_f) \\
 i_t &= \sigma(W_i \bullet [h_{t-1}, x_t] + b_i) \\
 \tilde{C}_t &= \tanh(W_C \bullet [h_{t-1}, x_t] + b_C) \\
 C_t &= f_t * C_{t-1} + i_t * \tilde{C}_t \\
 o_t &= \sigma(W_o[h_{t-1}, x_t] + b_o) \\
 h_t &= o_t * \tanh(C_t)
 \end{aligned}
 \tag{3.2.9}$$

The *forget gate* removes irrelevant information from the cell state. The cell state is updated in the next cell in the network. The sigmoid function does this task in the cell by updating the values of the cell state between 0 and 1. 0 means that particular information needed to be removed completely. 1 means the information has to be kept there as it is necessary information and needed to be passed on to the next cell. The *input gate* selects new candidate information to be added to the memory cell. The *update gate* updates information in the network. The LSTM cell has limited memory so it removes the irrelevant information from the network and adds relevant information from the current input and it goes on till the last timestamp of the network.

**Algorithm 3.5.** *Data:*  $\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \dots, (x^{(n)}, y^{(n)})\}$  *Trained LSTM model parameters*

*Initialize LSTM model parameters: weights and biases Set the learning rate  $\alpha$  Set the number of epochs  $N$*

$i = 1$  to  $N$  ( $x^{(i)}, y^{(i)}$ ) in *training\_data* time step  $t$  in input sequence  $x^{(i)}$  Perform forward pass through the LSTM layer Compute the predicted output  $\hat{y}_t$  Compute the loss or cost function based on  $\hat{y}_t$  and  $y^{(i)}$  Perform backward pass through the LSTM layer using backpropagation and gradient descent Update LSTM model parameters using the computed gradients and learning rate

### 3.2.3 Transformers

Transformers are a type of deep neural network where a self-attention mechanism is applied to capture long-term dependency on context. Transformers have the ability to process the information in parallel, not like RNNs, which enables the transformers to get full advantage of GPU parallel processing. Moreover, the self-attention mechanism is better at dependency capturing as it only focuses on the relevant word in a very long sentence. Transformers possess long-term memory. Theoretically speaking, the attention mechanism can capture infinite dependency of the input, thus transformers-based model i.e. BERT has been used for stock market sentiment analysis [16], the impact of Coronavirus on social life [25], Twitter sentiment analysis[20] and other Natural Language Processing (NLP) based tasks [17].

Transformers are made up of two components: Encoder and Decoder. see figure 3.12.

The encoder works as a feature extractor for transformers and the decoder uses those features to generate output domain sentences. Hence we are working on the classification problem, we will be using the encoder part as a feature extractor to perform text classification. The encoder is made-up of different parts. First of all the input text is tokenized with WordPiece tokenizer. Word-Piece uses a sub-word tokenization technique where the sentences are split into words and words are further split into sub-words if needed. Those small words or sub-words are called tokens. Each token in the vocabulary has its own numeric id. The tokens are replaced with their respective numbers in the vocabulary and fed into the input layer of the encoder. The numeric ids are called *Input Embeddings*.

The transformers process input embeddings in parallel which causes the transformers to lose the sequential information of the input sentence. This problem can be addressed by positional encodings. *Positional Encoding* 3.2.10 uses sin and cos functions to achieve this.



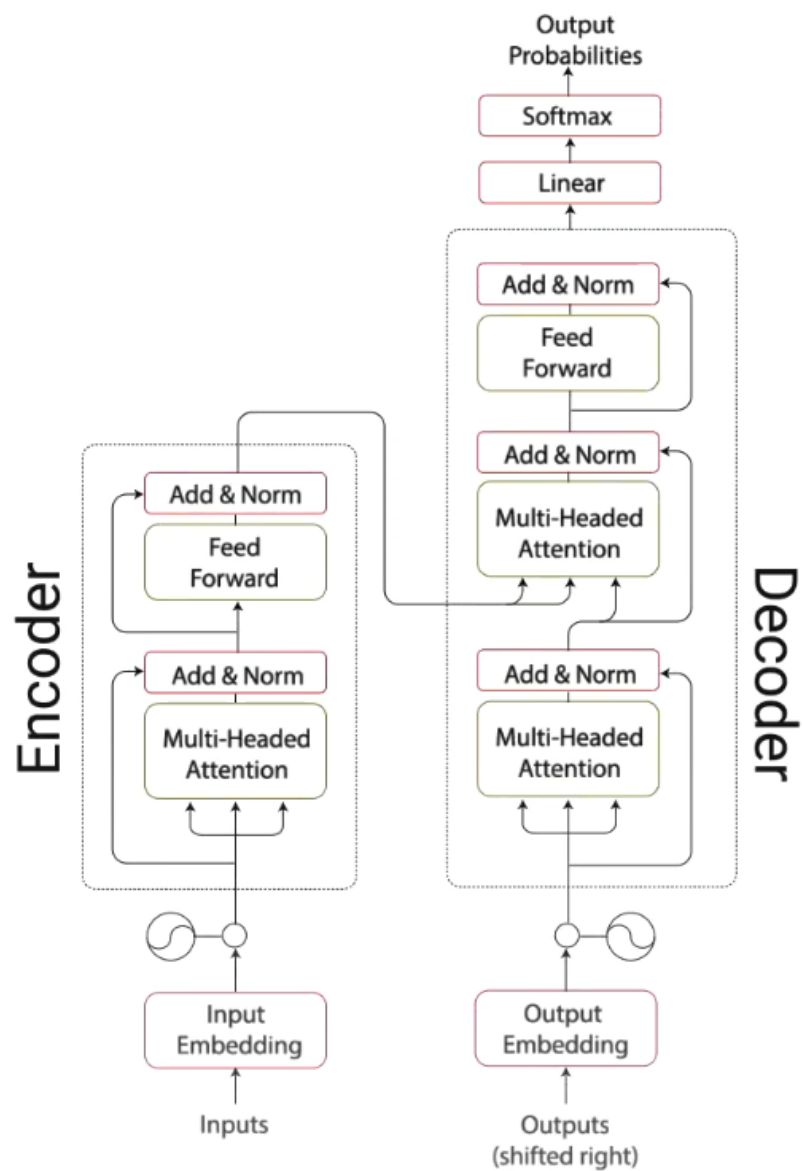
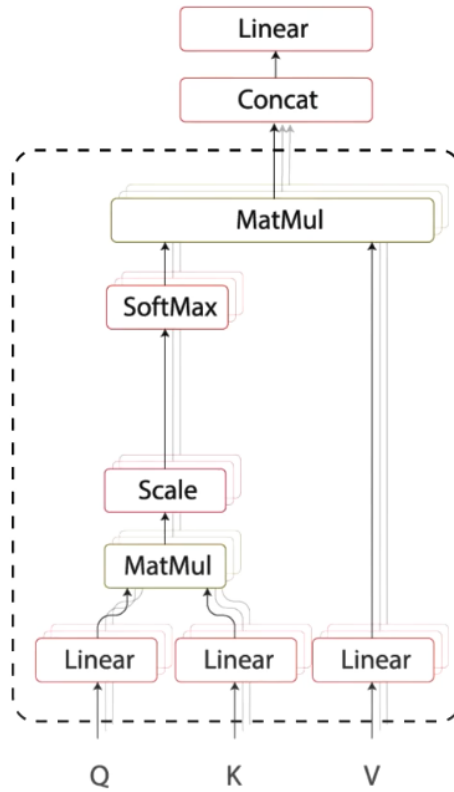


Figure 3.12: Transformers: Encoder and Decoder

$$\begin{aligned} \text{PositionalEncoding}_{pos,2i} &= \sin(pos/10000^{2i/d_{model}}) \\ \text{PositionalEncoding}_{pos,2i+1} &= \sin(\cos/10000^{2i/d_{model}}) \end{aligned} \quad (3.2.10)$$

$i$  is the index of the target dimension in the input vector. The sin function is used for even index numbers, cos function is used for odd number index to calculate the positional encodings. The positional encodings vector is added to input embeddings to calculate the final input. The input is fed to three different feed-forward networks to generate three distinct vectors called Query, Key, and Value.



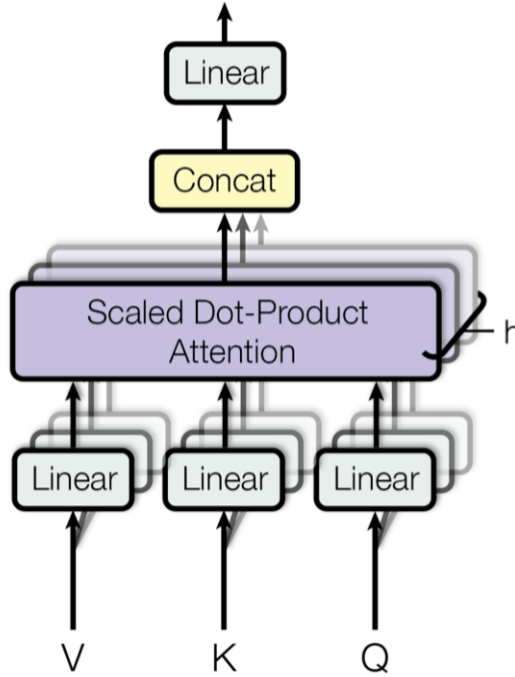
**Figure 3.13:** Self-attention mechanism

The Query and Key values are multiplied to get the attention scores matrix. The score matrix is scaled down by the square root of the dimension of the query/key vectors to avoid exploding gradient problems due to large multiplications. The scaled attention score shows how much attention each token in the input sentence gives to each token in the. Finally the scaled attention scores are converted to probabilities by applying the softmax function and finally multiplying those probabilities with a value vector to complete the self-attention mechanism. All these operations can be written as equation

## 3.2.11.

$$\text{SelfAttention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d}}\right) * V \quad (3.2.11)$$

Multiple self-attention heads are parallelly grouped together to form a multi-head attention mechanism: see figure 3.14. Each attention head in self-attention has its own Query, Key, and Value matrices.



**Figure 3.14:** Multi-head self-attention

Outputs of all heads are concatenated and applied a pointwise feed-forward network to it in the next step. Residual Layer (RL) 3.15 is applied to output at this step and input embeddings at this stage. RL helps in stabilizing the training and reducing the training time significantly. RL is followed by Layer Normalization where the output is further normalized. A couple of linear layers with ReLU 3.2.4 activation function is applied to further process the information. Finally, another RL is applied which takes the output of the previous layer and the output of the pointwise feed-forward network and followed by final Layer Normalization to get the meaningful extracted features from the input text.

There are various models that use Transformers Encoder, Decoder, or Both to perform

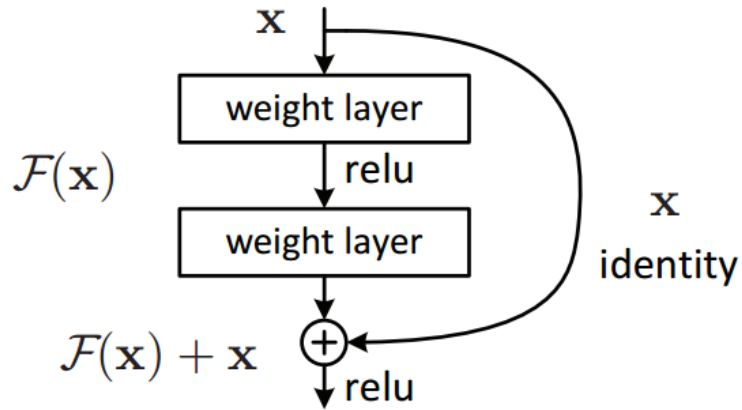


Figure 3.15: How Residual Layers works

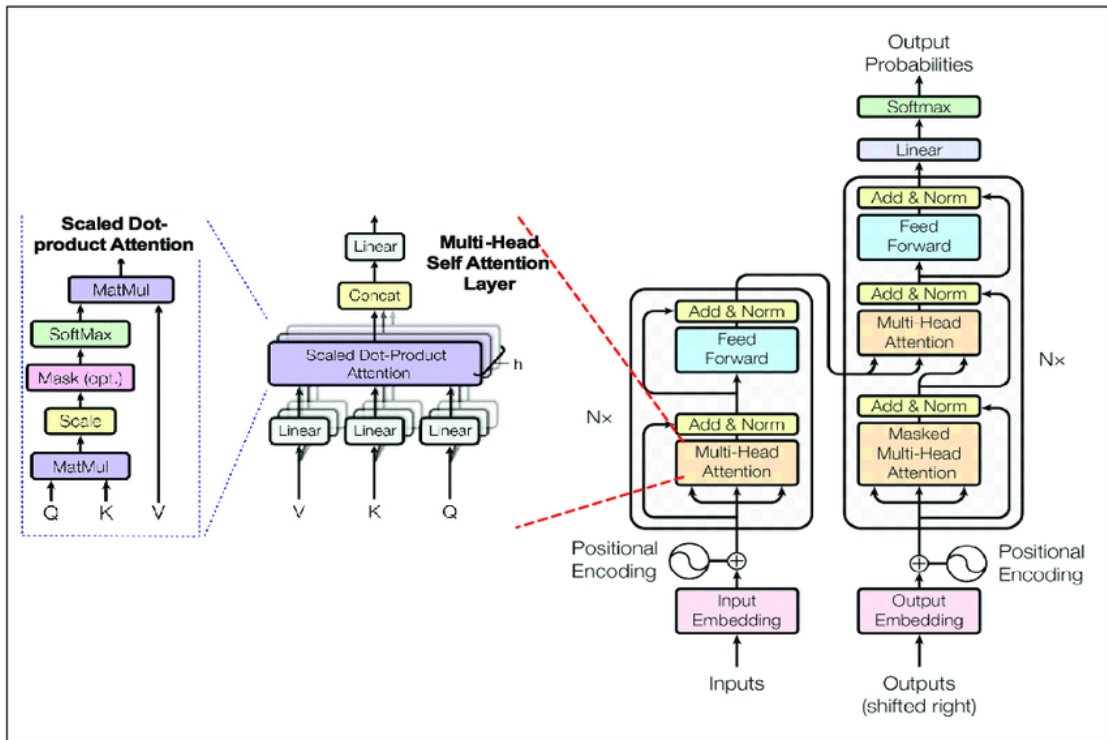
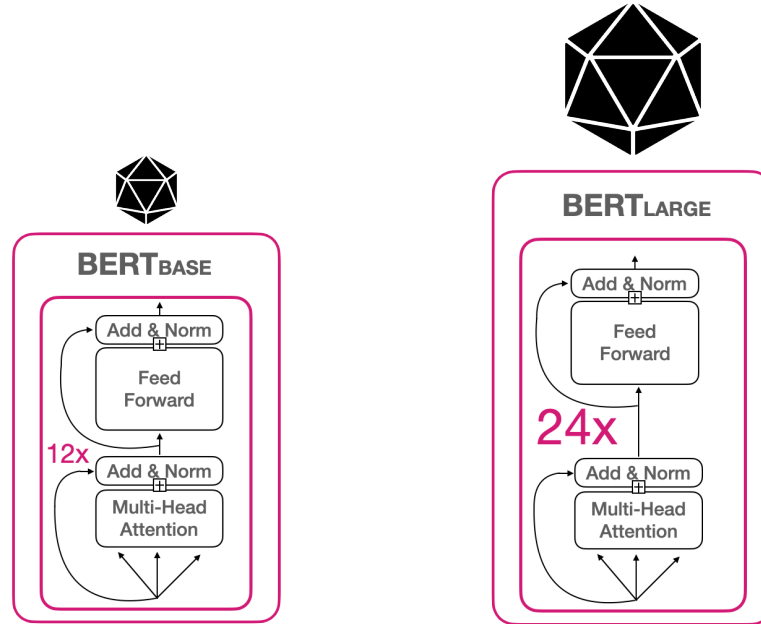


Figure 3.16: Transformers: A detailed look

certain tasks. We will be using the BERT encoder to classify this work. BERT stacks multiple layers of encoders. BERT has two variations: BERTbase and BERTlarge.



**Figure 3.17:** BERTbase and BERTlarge

BERTbase stacks 12 encoder layers while BERTlarge stacks 24 encoder layers. The hidden size of BERTbase is 768 and BERTlarge is 1024. The attention heads used by BERTbase are 12 while BERTlarge uses 16 heads. Each attention has an output size of 64. BERTbase has 110 million trainable parameters while BERTlarge has 340 million. We will be using Google Colab for our experiments that have a finite amount of GPU memory. It is not possible to train BERTlarge due to a large number of parameters so we will be training the BERTbase model only at the expense of a small margin of accuracy.

**Algorithm 3.6.** *Data:  $\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \dots, (x^{(n)}, y^{(n)})\}$  Trained BERT model*

*Initialize BERT model architecture Initialize model parameters with random weights*

*Set the learning rate  $\alpha$  Set the number of training epochs  $N$*

*$i = 1$  to  $N$   $(x^{(i)}, y^{(i)})$  in training\_data Tokenize the input text  $x^{(i)}$  and convert it to input IDs Generate attention masks and segment IDs for the input Perform forward pass through the BERT model Compute the predicted output  $\hat{y}$  Compute the loss or cost function based on  $\hat{y}$  and  $y^{(i)}$  Perform backward pass through the BERT model*

## CHAPTER 3: METHODOLOGY

*using backpropagation and gradient descent Update BERT model parameters using the computed gradients and learning rate*

## CHAPTER 4

# Dataset

Chapter 4 of the thesis presents a comprehensive overview of the dataset utilized for conducting experiments. It describes the procedures involved in dataset creation and labeling, providing insights to ensure the dataset's quality and relevance to the research objectives. We will also discuss the cleaning and preprocessing techniques applied to the dataset.

### 4.1 World Leaders

The selection of world leaders for the research posed a significant challenge following the understanding of their influence on SDGs. To address this challenge, three prominent surveys conducted by Statista and Twiplomacy for the year 2020 <sup>1</sup>, 2021 <sup>2</sup> and 2022 <sup>3</sup> were utilized. These surveys focused on identifying the most followed world leaders on Twitter. For the purpose of inclusivity and diversity, the union of all surveys was considered, resulting in a compilation of the top 15 world leaders, as seen in 4.1. It is important to note that the term "world leaders" in this context refers specifically to country heads.

---

<sup>1</sup><https://www.twiplomacy.com/twiplomacy-study-2020>

<sup>2</sup><https://www.statista.com/chart/17898/world-leaders-with-the-most-followers-on-twitter/>

<sup>3</sup><https://www.twiplomacy.com/top-50-world-leader-power-ranking>

S.No	World Leader	Twitter ID	Followers	Country
1	Narendra Modi	narendramodi	88.6M	India
2	Joe Biden	JoeBiden	37.2M	USA
3	Recep Tayyip Erdoğan	RTErdogan	20.5M	Turkiye
4	Pope Francis	pontifex	18.8M	Vatican City
5	Joko Widodo	jokowi	19.5M	Indonesia
6	Imran Khan	ImranKhanPTI	19.2M	Pakistan
7	Jair Bolsonaro	jairbolsonaro	11.3M	Brazil
8	Mohammed bin Rashid Al Maktoum	HHShkMohd	11M	UAE
9	Rania Al Abdullah	QueenRania	10.1M	Jordan
10	Volodymyr Zelenskyy	ZelenskyyUa	7.2M	Ukraine
11	Gustavo Petro	petrogustavo	6.7M	Colombia
12	Justin Trudeau	justinTrudeau	6.4M	Canada
13	Nayib Bukele	nayibbukele	5.1M	El Salvador
14	Nicolas Maduro	NicolasMaduro	4.5M	Venezuela
15	Gabriel Boric	gabrielboric	1.9M	Chile

Table 4.1: World Leaders on Twitter

## 4.2 Dataset Creation

In this research, we utilized the SNScrape Python library <sup>4</sup> to extract tweets from Twitter. SNScrape offers a range of functionalities for scraping user profiles, posts, hashtags, and other relevant information from various social media platforms. Specifically, SNScrape allows fetching data from the following platforms:

- Public posts from Facebook profiles and pages.
- User profiles, hashtags, and locations from Instagram.
- User posts and subreddits from Reddit.
- Tweets, hashtags, and profiles from Twitter.

<sup>4</sup><https://github.com/JustAnotherArchivist/snsrape>



One of the notable advantages of SNScrape is its independence from additional specialized libraries during installation. It seamlessly integrates with Python version 3.8 or higher, automatically managing the installation of its necessary dependencies. In our data collection process, we supplied the usernames of world leaders' Twitter accounts and specified the timeframe from 2015 to 2023. SNScrape efficiently downloaded data for each user individually and stored it in separate CSV files. Subsequently, we consolidated these files to create a comprehensive dataset encompassing the data of all world leaders.

Our data collection efforts resulted in a total of 50,270 tweets from world leaders. To ensure a robust evaluation of our models, we partitioned the dataset into two distinct subsets: a training set and a testing set. The training set accounted for 85% of the data, while the remaining 15% constituted the testing set. This division allowed us to train our models on a substantial portion of the dataset while maintaining an independent set of tweets for evaluating their performance. By adhering to this division, we aimed to accurately assess the effectiveness and generalization of our models on unseen data.

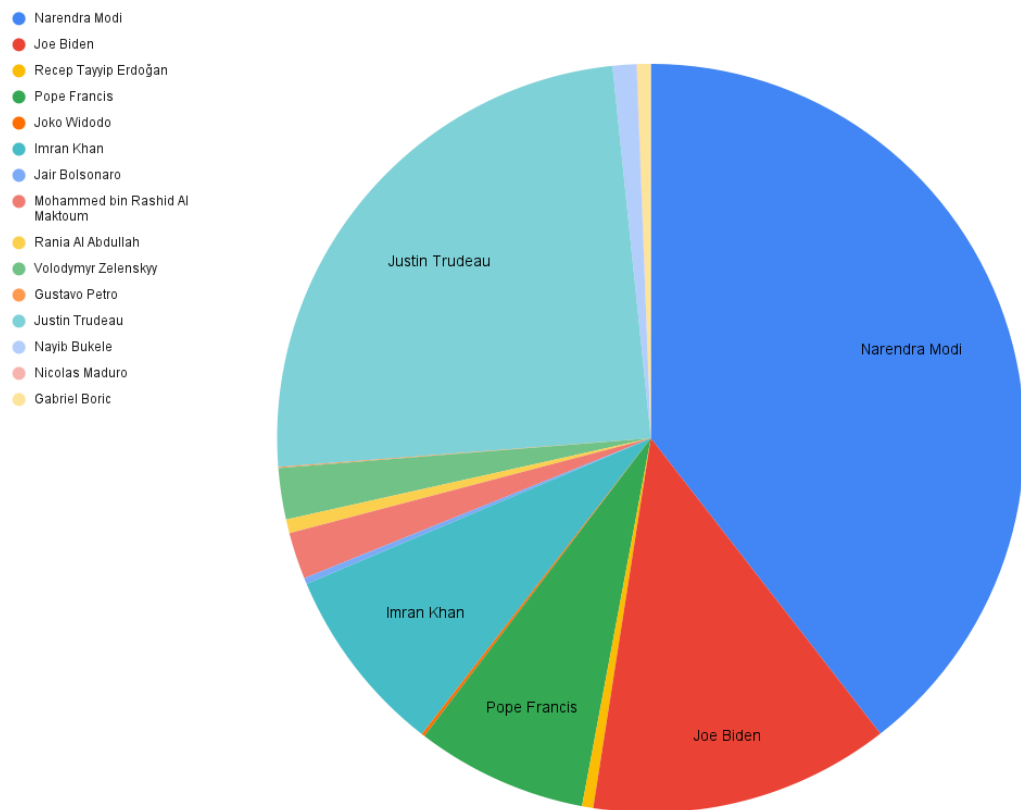
### 4.3 Tweets Preprocessing

Data collected from Twitter using the SNScrape library contained various elements such as links, emojis, and other irrelevant information. It is essential to preprocess the data by removing these elements as they can potentially impact the performance of machine learning models. By eliminating links and emojis we aim to focus solely on the textual content of the tweets. This preprocessing step helps streamline the analysis process and enhances the accuracy and reliability of the machine learning models used for sentiment analysis and bias detection.

In our data preprocessing pipeline, we utilized the *tweet-preprocessor* library<sup>5</sup>, a Python-based tweet processing tool specifically designed for cleaning tweets. By incorporating this library, we were able to streamline our text cleaning process effectively. To remove unwanted elements from the tweets, we configured the parameters of the *tweet-preprocessor* library to eliminate links, emojis, and smilies. Additionally, we implemented a custom function that replaced the "@" and "#" symbols with an empty string, ensuring that usernames and tags were included in the final text. This preprocessing step

---

<sup>5</sup><https://github.com/s/preprocessor>



**Figure 4.1:** Tweets Distribution By World Leaders

was consistently applied to both the training and test examples, ensuring a standardized and cleaned dataset for further analysis and model training. Retaining usernames and hashtags in tweet preprocessing can offer valuable insights and preserve important contextual information.

Here is a sample tweet from the dataset that contains hashtags and a link.

*Under our vision of promoting #EcoTourism the historic #Monroe hiking trail has been restored in #KP under our #10BillionTreeTsunami - traversing 50 km of pristine natural forest with two overnight #glamp sites.  
<https://t.co/LCC4Jrshjz>*

**Figure 4.2:** Before Cleaning

After the cleaning, the above tweet looks like this.

*Under our vision of promoting EcoTourism the historic Monroe hiking trail has been restored in KP under our 10BillionTreeTsunami - traversing 50 km of pristine natural forest with two overnight glamp sites.*

**Figure 4.3:** After Cleaning

## 4.4 Dataset Labeling

Labeling a large dataset of over 50,000 tweets manually requires substantial human resources and time. To streamline the labeling process, we utilized an ontology dataset consisting of 4,116 keywords to label the dataset automatically. This dataset is created by Bautista-Puig and Mauleon (2019) [13]. Each tweet was searched for these keywords, and if a keyword was found, the tweet was flagged as relevant to the corresponding Sustainable Development Goal (SDG).

To encode the labels, an array of 17 labels, representing the 17 SDGs, was created. Each tweet was assigned a single or more labels based on its association with the respective SDG. A value of 1 was placed in the corresponding SDG column if the tweet contained

keywords related to that specific SDG; otherwise, a value of 0 was assigned. It is important to note that a tweet can be related to zero or more SDGs, and multiple SDGs can be represented by a single tweet. As a result of this labeling process, a dataset was generated where each tweet was labeled with one or more of the 17 SDGs. The encoded labels are represented by an array of 0s and 1s, such as  $[0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,1]$ , where each element corresponds to an SDG from 1 to 17. A value of 1 in a specific cell indicates that the corresponding tweet contains keywords associated with that particular SDG.

A few examples of labels and encoded labels are shown in ??, and ??, respectively.

<b>Tweet</b>	<b>Labels</b>
On 29th April nation will witness Mother of all Jalsas at Minar-i-Pakistan and the corrupt mafia will see how strongly nation rejects these money launderers, tax evaders & asset concealers and how steadfastly it stands with our Judiciary.	[SDG8]
On the special occasion of his 80th birthday, greetings to the versatile K. J. Yesudas Ji. His melodious music and soulful renditions have made him popular across all age groups. He has made valuable contributions to Indian culture. Wishing him a long and healthy life.	[SDG3]
At 11 AM tomorrow, 21st November, I would be addressing the Convocation of PDPU, Gandhinagar. Will also be inaugurating various Centres that would boost research, innovation and learning at PDPU.	[SDG4, SDG9]

**Figure 4.4:** Tweets with Labels

In our analysis, it was observed that a significant proportion of the tweets (23%) is focused on a single Sustainable Development Goal (SDG). This finding indicates that world leaders often prioritize and emphasize a specific SDG in their discourse. Approximately 17% of the tweets addressed another SDG, while 10% touched upon seven SDGs. About 1.3 percent of the tweets are not labeled, since they don't belong to any of the SDGs. For a comprehensive overview of the distribution, refer to Figure 4.6. The imbalanced distribution of SDGs in the dataset presents a significant challenge, increasing the complexity of addressing the issue.

Tweet	Encoded Labels
On 29th April nation will witness Mother of all Jalsas at Minar-i-Pakistan and the corrupt mafia will see how strongly nation rejects these money launderers, tax evaders & asset concealers and how steadfastly it stands with our Judiciary.	[0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0]
On the special occasion of his 80th birthday, greetings to the versatile K. J. Yesudas Ji. His melodious music and soulful renditions have made him popular across all age groups. He has made valuable contributions to Indian culture. Wishing him a long and healthy life.	[0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
At 11 AM tomorrow, 21st November, I would be addressing the Convocation of PDPU, Gandhinagar. Will also be inaugurating various Centres that would boost research, innovation and learning at PDPU.	[0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0]

Figure 4.5: Tweets with Encoded Labels

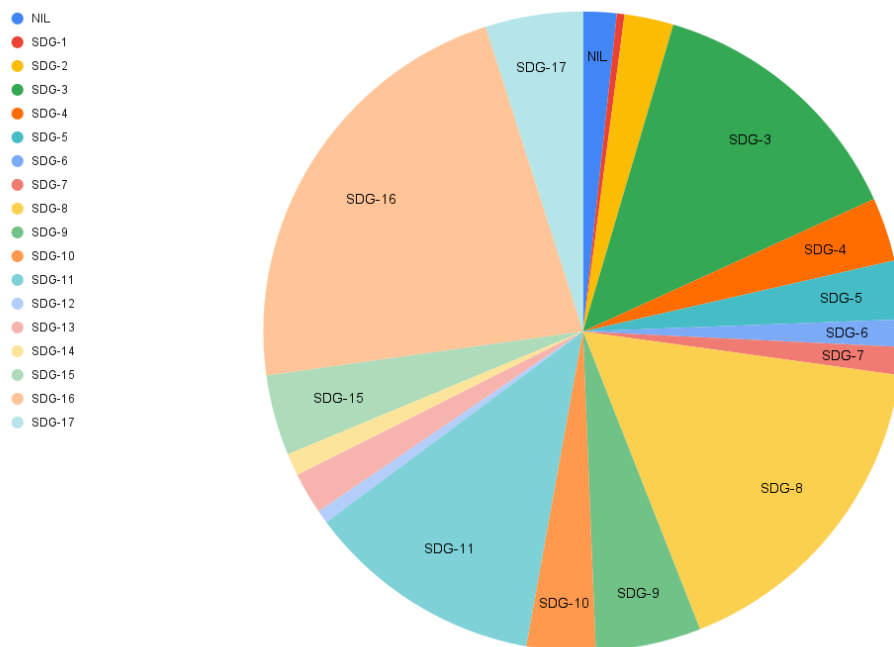


Figure 4.6: SDG Distribution

# Results & Discussion

This chapter explains the experimental setup, and results and discusses the results in detail.

## 5.1 Experimental Setup

We carried out all the experiments on Google Colab. Google Colab provides us with a free GPU having 12-16GB of CPU and GPU memory, respectively. All machine learning experiments are performed on CPU while GPU was used for deep learning-based approaches i.e. feed-forward neural network, LSTM, and BERT. The overall relevant system information is logged in table 5.1.

Architecture	x86_64
CPU(s)	2
Model name	Intel(R) Xeon(R) CPU @ 2.20GHz
CPU MHz	2199.998
L1d cache	32K
L2 cache	256K
L3 cache	56320K
Memory	12.985 GBs

**Table 5.1:** System Information

The GPU information is provided in Figure 5.1.

GPU Name		Persistence-M		Bus-Id	Disp.A	Volatile Uncorr. ECC	
Fan	Temp	Perf	Pwr:Usage/Cap	Memory-Usage	GPU-Util	Compute M.	MIG M.
0	Tesla T4		Off	00000000:00:04.0	Off	0	
N/A	62C	P8	10W / 70W	0MiB / 15109MiB		0%	Default N/A

Figure 5.1: SDG Distribution

## 5.2 Results

We logged 4 evaluation metrics of the test set on all techniques: *exact accuracy*, *average accuracy*, *average precision*, and *average recall*. The confusion matrix is not logged as there are 17 categories and we have a multi-label problem. For multi-label problems, independent confusion matrices are recorded for each category. It becomes challenging to include or analyze 17 confusion matrices. The exact accuracy checks whether all 17 predicted labels are the same as all 17 true labels. Average accuracy calculates the independent accuracy of each class on the test set and then calculates the average of it. Average precision calculates the independent precision of each class on the test set and then calculates the average of those values. The same goes for average recall [15]. All model evaluation results on the test set are recorded in table 5.2.

Model	Exact Accuracy	Average Accuracy	Average Precision	Average Recall
<b>Logistic Regression</b>	47.9%	95.3%	69.2%	32.1%
<b>K-nearest neighbour</b>	45.1%	94.8%	68.9%	30.8%
<b>Decision Tree</b>	22.9%	90.4%	14.9%	17.3%
<b>FFNN</b>	51.2%	95.8%	63.3%	45%
<b>LSTM</b>	86.04%	99%	91.1%	83.6%
<b>BERT</b>	91.6%	99.4%	87.9%	97%

Table 5.2: All models results

Transformer-based BERT model outperforms all other models on all 4 evaluation metrics.

### 5.2.1 Decision Tree

We trained Decision Tree models with different hyperparameters to get the best accuracy but looks like DT is the worst choice for our problem. We empirically get the lowest accuracy among all other models. We used Gini impurity to split the tree. The average precision and average recall values are very low which indicates that the model often selects 1's as 0's and 0's as 1's. The average accuracy is 90.43% but the exact accuracy is only 22.91%, the reason for this could be because of the high imbalance in our dataset.

### 5.2.2 Logistic Regression

We trained LR with different hyperparameters and logged the best results. We used the L2 penalty to regularize the model to avoid overfitting. LR can not handle multi-label classification natively. We train 17 different models for all categories. Each model is a binary classifier. All models are trained individually. The test set is passed through all the models and evaluations are combined in one set. LR performs better than other machine learning models on all evaluation metrics. Even it outperforms simple FFNN.

### 5.2.3 K-nearest neighbour

We trained different KNN models with different hyperparameters. The accuracy of KNN heavily relies on a number of neighbors' hyperparameters. We tried out a different number of neighbors and found out that we get the highest accuracy when the number of neighbors are 7. We used Minkowski distance with the value of  $p$  as 1, 2, and 3. We found out that euclidean distance i.e. when  $p$  value is 3 empirically works well on our problem. KNN supports multi-label classification inherently so we don't need to train 17 different models for each category. KNN performs better than Decision Tree on all evaluation metrics. Also, KNN outperforms FFNN on exact accuracy and average accuracy but is unable to defeat it on average precision and average recall. The reason is that our dataset is highly imbalanced and most of the labels are 0. So the model often predicts 0 even if it encounters 1 label in a category. Another proof is the average precision and average recall values. Average precision is relatively higher as it indicates that the model is exact at predicting 1 label. The model often predicts 1 when the actual label is 1. Low average recall indicates that mostly 1's are predicted as 0's. The model



does not take the risk of predicting a category in the input text which is not present.

#### 5.2.4 FFNN

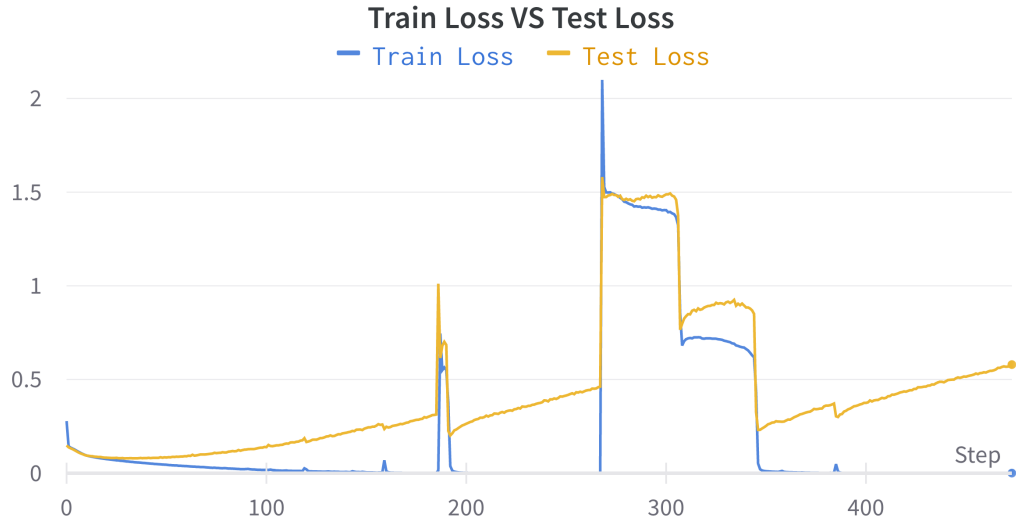
We perform different experiments with different numbers of layers and numbers of neurons and tried to figure out the optimal number of layers and neurons in each layer. We designed a three-layer neural network where the input layer size is 300 (from GloVe embeddings), 2nd layer with a dimension of 256, and 3rd layer with a size of 128. Finally, we have 17 1-layer 128x1 FFNN. All 17 networks are having a Sigmoid unit on them for binary classification. We connected the first network which has 3 layers to all the other 17 networks and trained it jointly. We trained the model for 500 epochs to see which checkpoint gives us the best accuracy. We choose Adam optimizer over Stochastic Gradient Descent (SGD) as empirically it works well for any problem in comparison to SGD. The model converges very smoothly due to the selection of the optimizer.



**Figure 5.2:** FFNN training loss: 500 epochs

We recorded training and testing loss at each epoch and compare them in figure 5.3. The model converges at different epochs, similarly, the train and test loss meet at different epochs. At the 52 epoch, we get the best accuracy. After that, the test loss is increasing, indicating that the model is overfitting now. We did not use any regularization with our FFNN.

We choose the batch size of 1024 due to the specification of our system hardware limi-



**Figure 5.3:** FFNN: Train vs Test loss

tation. The exact accuracy metric was not good but the average accuracy, the average precision, and the average recall were better than Decision Tree. It also beat the LG and KNN with average recall.

### 5.2.5 LSTM

We trained a simple LSTM model with a single layer. We trained the model for 100 epochs with Adam optimizer.

The training loss converges at the 2nd epoch on training data. The test decreases to 9 epochs. After epoch 10th, the test loss increases slowly which indicates that the model is now overfitting. We avoid overfitting by introducing dropout with a probability of 0.5.

The exact accuracy [5.5](#) is not decreasing after the 4th epoch due to the dropout regularization. Also, further training will lead the model to be around the same accuracy. The model achieves better exact accuracy, average accuracy, average precision, and average recall than LR, KNN, DT, and FFNN models due to the sequential input processing nature of LSTM. As the English language is written from left to write and the words on the left side of a word have a greater impact on the meaning of that word. That's the reason that we get better accuracy than the previous models. We kept the batch size

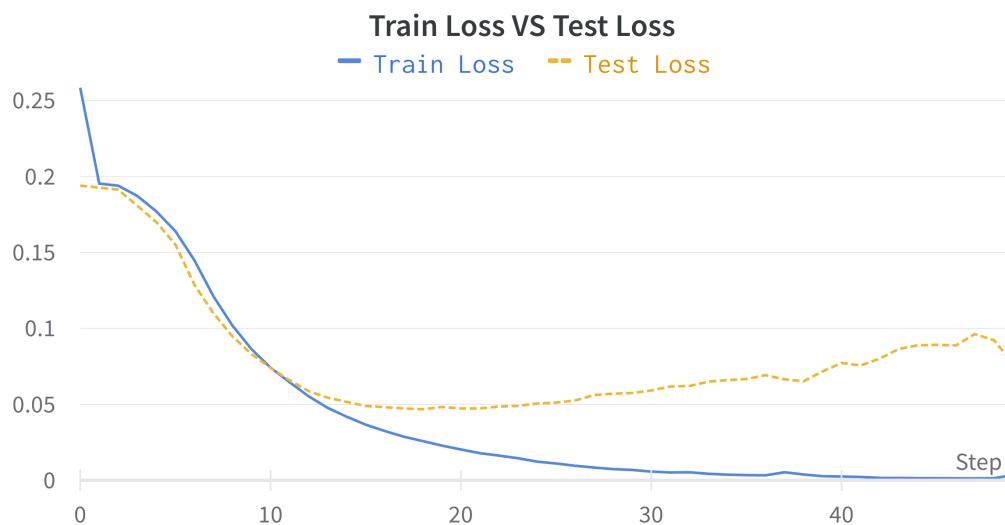


Figure 5.4: LSTM: Train vs Test loss

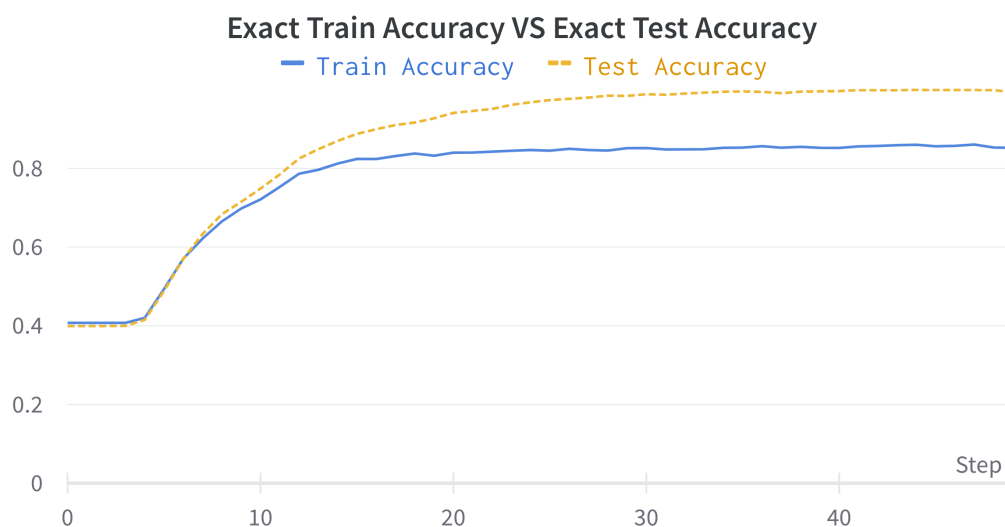
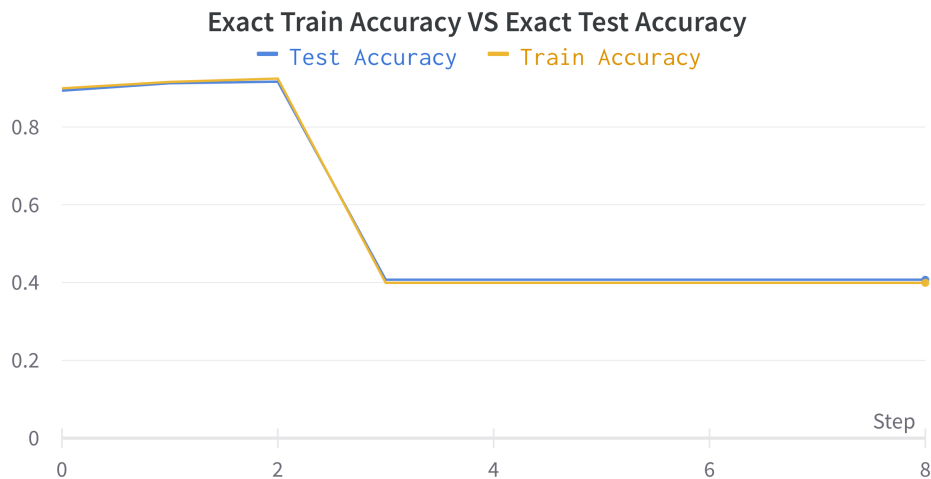


Figure 5.5: LSTM: Train vs Test exact accuracy

as 1024 instances per training iteration.

### 5.2.6 BERT

BERT is from the family of transformers models which makes it relatively bigger than all previous models. We keep the batch size as 16 examples per training iteration due to memory limitation. We train the model for 10 epochs and test the model after each epoch. We get out optimal results on 2nd epoch. BERT achieves the highest exact accuracy of 91.66%, an average accuracy of 99.44%, average precision of 87.88%, and an average recall of 97%. BERT utilizes a self-attention mechanism that takes the past and future context of words into consideration which makes it better in terms of accuracy than all other models. A large number of parameters increases the training time of the model. It also increases the model size on the disk and inference time as well.



**Figure 5.6:** BERT: Train vs Test exact accuracy

## 5.3 Summary

In this chapter, we provided an analysis of the results obtained from six machine learning and deep learning models trained on our dataset. A thorough comparison of the model performances is presented, along with a discussion on the factors influencing their respective outcomes. In the next chapter, we will conclude our thesis and outline potential future research.

# Conclusion and Future Work

## 6.1 Conclusion

Our research is based on predicting the labels of the tweets of world leaders which will help in finding the intent of world leaders in Sustainable Development Goals (SDGs) based on their tweets. For this purpose, we created a dataset by collecting tweets of world leaders from Twitter. We labeled the dataset with ontology by finding the relative keywords to each SDG and classified the Tweets based on those keywords. This makes our problem a multi-label classification problem i.e. each tweet can contain zero or more categories. Zero means that the tweet does not talk about any SDG whereas one or more categories means that the tweet is about those SDGs.

We trained 3 machine learning and 3 deep learning-based models on our dataset to predict the categories of tweets. Machine Learning based models are Logistic Regression, K-nearest neighbor, and Decision Tree whereas Deep Learning based models are Feed-Forward Neural Network, LSTM-1-layer, and BERT (transformers based). Machine Learning based and Feed-Forward Neural Network uses GloVe 300 dimension word embeddings for converting tweets to their numerical representation where LSTM-based models and BERT get their own word embedding during training. LSTM-based models use 100 tokens as input size whereas BERT uses 128 tokens as input. Machine Learning based models, FFNN, and LSTM-based model tokenizers are based on spaces between words in the sentence whereas BERT uses WordPiece tokenizer which is a sub-word-based tokenizer.

The performance of these models was tested against each other in terms of exact accu-

racy, average accuracy, average precision, and average recall. The experimental results show that BERT performs better than all other models in terms of exact accuracy and average recall at 91.6% and 97%, respectively. LSTM performs better than all models except in terms of average accuracy at 99%. The reason for the performance of the BERT model is the self-attention mechanism. LSTM models incorporate previous words with current words during the processing of tweets to perform better than FFNN, Linear Regression, KNN, and Decision Tree.

## 6.2 Future Work

1. This work can be expanded by gathering data from other Twitter users to study their sentiments toward Sustainable Development Goals (SDGs). A larger and balanced dataset will also improve the performance of deep learning-based models.
2. Applying unsupervised machine learning models for labeling the dataset instead of relying solely on ontologies has the potential to improve the results.
3. Exploring the training of the latest transformer-based models like XLNet and RoBERTa could be a valuable research endeavor, as these models have the potential to surpass the accuracy achieved in this thesis.

# Bibliography

- [1] Fabrizio Sebastiani and Andrea Esuli. “Sentiwordnet: A publicly available lexical resource for opinion mining”. In: *Proceedings of the 5th international conference on language resources and evaluation*. European Language Resources Association (ELRA) Genoa, Italy. 2006, pp. 417–422.
- [2] Stanford. “Sentiment140”. In: (2009). URL: <http://help.sentiment140.com/home>.
- [3] Heum Park and Hyuk-Chul Kwon. “Improved Gini-index algorithm to correct feature-selection bias in text classification”. In: *IEICE transactions on information and systems* 94.4 (2011), pp. 855–865.
- [4] Alex Graves and Alex Graves. “Long short-term memory”. In: *Supervised sequence labelling with recurrent neural networks* (2012), pp. 37–45.
- [5] Bo Pang and Lillian Lee. “Cornell Movie Review Data”. In: (2012). URL: <https://www.cs.cornell.edu/people/pabo/movie-review-data/>.
- [6] “Twitter US Airline Sentiment”. In: (2015). URL: <https://www.kaggle.com/datasets/crowdfunder/twitter-airline-sentiment>.
- [7] “United Nation Sustainable Development Goals”. In: (2015). URL: <https://sdgs.un.org/goals>.
- [8] Ebru Aydođan and M Ali Akcayol. “A comprehensive survey for sentiment analysis tasks using machine learning techniques”. In: *2016 International Symposium on INnovations in Intelligent SysTems and Applications (INISTA)*. IEEE. 2016, pp. 1–7.
- [9] Sidra Ijaz et al. “Biasness identification of talk show’s host by using twitter data”. In: *2017 13th International Conference on Emerging Technologies (ICET)*. IEEE. 2017, pp. 1–6.

- [10] Paramita Ray and Amlan Chakrabarti. “Twitter sentiment analysis for product review using lexicon method”. In: *2017 International Conference on Data Management, Analytics and Innovation (ICDMAI)*. IEEE. 2017, pp. 211–216.
- [11] Jacob Devlin et al. “Bert: Pre-training of deep bidirectional transformers for language understanding”. In: *arXiv preprint arXiv:1810.04805* (2018).
- [12] Ahmed Sulaiman M Alharbi and Elise de Doncker. “Twitter sentiment analysis with a deep neural network: An enhanced approach using user behavioral information”. In: *Cognitive Systems Research* 54 (2019), pp. 50–61.
- [13] Nuria Bautista. “SDG ontology”. In: (Nov. 2019). DOI: [10.6084/m9.figshare.11106113.v1](https://doi.org/10.6084/m9.figshare.11106113.v1). URL: [https://figshare.com/articles/dataset/SDG\\_ontology/11106113](https://figshare.com/articles/dataset/SDG_ontology/11106113).
- [14] Elia Gabarron et al. “Diabetes on Twitter: a sentiment analysis”. In: *Journal of diabetes science and technology* 13.3 (2019), pp. 439–444.
- [15] Brendan Juba and Hai S Le. “Precision-recall versus accuracy and the role of large data sets”. In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 33. 01. 2019, pp. 4039–4048.
- [16] Matheus Gomes Sousa et al. “BERT for stock market sentiment analysis”. In: *2019 IEEE 31st international conference on tools with artificial intelligence (ICTAI)*. IEEE. 2019, pp. 1597–1601.
- [17] Ian Tenney, Dipanjan Das, and Ellie Pavlick. “BERT rediscovers the classical NLP pipeline”. In: *arXiv preprint arXiv:1905.05950* (2019).
- [18] Nhan Cach Dang, Maria N Moreno-Garcia, and Fernando De la Prieta. “Sentiment analysis based on deep learning: A comparative study”. In: *Electronics* 9.3 (2020), p. 483.
- [19] Sardar Haider Waseem Ilyas et al. “Analyzing Brexit’s impact using sentiment analysis and topic modeling on Twitter discussion”. In: *The 21st Annual International Conference on Digital Government Research*. 2020, pp. 1–6.
- [20] Marco Pota et al. “An effective BERT-based pipeline for Twitter sentiment analysis: A case study in Italian”. In: *Sensors* 21.1 (2020), p. 133.



## BIBLIOGRAPHY

- [21] Sudhir Kumar Sharma, Mohit Daga, and Bhawna Gemini. “Twitter sentiment analysis for brand reputation of smart phone companies in India”. In: *Proceedings of ICETIT 2019: Emerging Trends in Information Technology*. Springer. 2020, pp. 841–852.
- [22] Purva Grover et al. “Influence of political leaders on sustainable development goals—insights from twitter”. In: *Journal of Enterprise Information Management* (2021).
- [23] Ankit Murarka, Balaji Radhakrishnan, and Sushma Ravichandran. “Classification of mental illnesses on social media using RoBERTa”. In: *Proceedings of the 12th international workshop on health text mining and information analysis*. 2021, pp. 59–68.
- [24] Dhruval Shah, Yanyan Li, and Ahmad Hadaegh. “Twitter based sentiment analysis of each presidential candidate using long short-term memory”. In: *International Journal of Computer Science and Security* 15 (2021), pp. 87–96.
- [25] Mrityunjay Singh, Amit Kumar Jakhar, and Shivam Pandey. “Sentiment analysis on the impact of coronavirus in social life using the BERT model”. In: *Social Network Analysis and Mining* 11.1 (2021), p. 33.
- [26] Ashwini Ramesh. “" SDG 2030: Analysis of Political Tweets Using Deep Learning Approach”. In: *Global Media Journal: Indian Edition* 14.2 (2022).
- [27] Emelie Rosenberg et al. “Sentiment analysis on Twitter data towards climate action”. In: (2023).