STRESS DETECTION FROM TWEETS

A Summer internship Report Submitted in partial fulfillment of the requirements for the award of the degree of

BACHELOR OF TECHNOLOGY IN
COMPUTER SCIENCE AND ENGINEERING-DATA SCIENCE

Submitted by

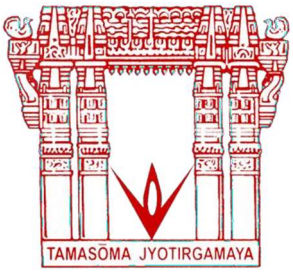
Ms. Bhavya Kesi Reddy (21071A6777)

Mr. Eppa Sai Vardhan Reddy (21071A6784)

Mr. P. Lokesh (21071A67B2)

Under the guidance of

Mrs. Y. Bhanu Sree
(Assistant Professor, Department of CSE- (CYS, DS) and AI&DS)



DEPARTMENT OF CSE- (CYS, DS) and AI&DS

VALLURUPALLI NAGESWARA RAO VIGNANA JYOTHI INSTITUTE OF ENGINEERING AND TECHNOLOGY

An Autonomous Institute, NAAC Accredited with 'A++' Grade NBA Accredited for CE, EEE, ME, ECE, CSE, EIE, IT, AME B. Tech Courses Approved by AICTE, New Delhi, Affiliated to JNTUH Recognized as "College with Potential for Excellence" by UGC ISO 9001:2015 Certified, QS I GUAGE Diamond Rated

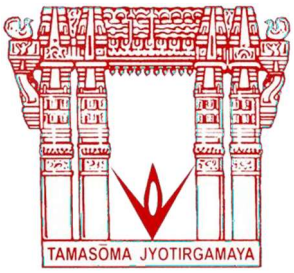
Vignana Jyothi Nagar, Pragathi Nagar, Nizampet (S.O), Hyderabad – 500 090, TS, India

VALLURUPALLI NAGESWARA RAO VIGNANA JYOTHI INSTITUTE OF ENGINEERING AND TECHNOLOGY

An Autonomous Institute, NAAC Accredited with 'A++' Grade NBA Accredited for CE, EEE, ME, ECE, CSE, EIE, IT, AME B. Tech Courses Approved by AICTE, New Delhi, Affiliated to JNTUH Recognized as "College with Potential for Excellence" by UGC ISO 9001:2015 Certified, QS I GUAGE Diamond Rated

Vignana Jyothi Nagar, Pragathi Nagar, Nizampet (S.O), Hyderabad – 500 090, TS, India

DEPARTMENT OF CSE-(CYS, DS) and AI&DS



CERTIFICATE

This is to certify that the project report entitled "**Stress Detection From Tweets**" is a bonafide work done under our supervision and is being submitted by team **Ms. Bhavya Kesi Reddy (21071A6777), Mr. Eppa Sai Vardhan Reddy (21071A6784), Mr. P. Lokesh (21071A67B2)** in partial fulfilment for the award of the degree of **Bachelor of Technology** in Computer Science and Engineering Data Science, of the VNRVJIET, Hyderabad during the academic year 2023-2024.Certified further that to the best of our knowledge the work presented in this thesis has not been submitted to any other University or Institute for the award of any Degree or Diploma.

Mrs. Y. Bhanu Sree
Assistant Professor
Dept. of CSE- (CYS, DS) and AI&DS
VNR VJIET

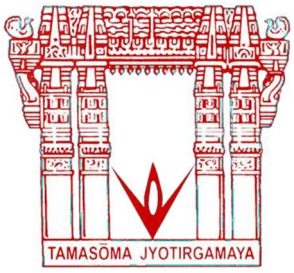
Dr.M.Raja Sekar
Professor and Head
Dept. of CSE- (CYS, DS) and AI&DS
VNR VJIET

VALLURUPALLI NAGESWARA RAO VIGNANA JYOTHI INSTITUTE OF ENGINEERING AND TECHNOLOGY

An Autonomous Institute, NAAC Accredited with 'A++' Grade NBA Accredited for CE, EEE, ME, ECE, CSE, EIE, IT, AME B. Tech Courses Approved by AICTE, New Delhi, Affiliated to JNTUH Recognized as "College with Potential for Excellence" by UGC ISO 9001:2015 Certified, QS I GUAGE Diamond Rated

Vignana Jyothi Nagar, Pragathi Nagar, Nizampet (S.O), Hyderabad – 500 090, TS, India

DEPARTMENT OF CSE-(CYS, DS) and AI&DS



DECLARATION

We declare that the major project work entitled "**Stress Detection From Tweets**" submitted in the department of **CSE-(CYS, DS) and AI&DS**, Vallurupalli Nageswara Rao Vignana Jyothi Institute of Engineering and Technology, Hyderabad, in partial fulfillment of the requirement for the award of the degree of **Bachelor of Technology** in **CSE- (CYS, DS) and AI&DS** is a bonafide record of our own work carried out under the supervision of Mrs. Y Bhanu sree**, Assistant Professor, Department of CSE- (CYS, DS) and AI&DS, VNRVJIET**. Also, we declare that the matter embodied in this thesis has not been submitted by us in full or in any part thereof for the award of any degree/diploma of any other institution or university previously.

Place: Hyderabad

Bhavya Kesi Reddy	**Eppa Sai Vardhan Reddy**	**P. Lokesh**
(21071A6777)	(21071A6784)	(21071A67B2)

ACKNOWLEDGEMENT

Firstly, we would like to express our immense gratitude towards our institution VNR Vignana Jyothi Institute of Engineering and Technology, which created a great platform to attain profound technical skills in the field of Computer Science, thereby fulfilling our most cherished goal.

We are very much thankful to our Principal, **Dr. Challa Dhanunjaya Naidu,** and our Head of Department, **Dr. M. Raja Sekar**, for extending their cooperation in doing this project within the stipulated time.

We extend our heartfelt thanks to our guide, **Mrs. Y. Bhanu Sree,** and the project coordinators **Dr. Vempaty Prashanthi and Mrs. B. Deepika** for their enthusiastic guidance throughout the course of our project.

Last but not least, our appreciable obligation also goes to all the staff members of the Computer Science & Engineering department and to our fellow classmates who directly or indirectly helped us.

Ms. Bhavya Kesi Reddy (21071A6777)

Mr. Eppa Sai Vardhan Reddy (21071A6784)

Mr. P Lokesh (21071A67B2)

ABSTRACT

In our rapidly evolving digital society, the "Stress Detection from Tweets" project emerges as a pioneering initiative to navigate the intricate intersection of language and mental health within the expansive realm of Twitter. This undertaking harnesses cutting-edge technologies such as Natural Language Processing (NLP) and machine learning to unravel the language patterns embedded in tweets, with a specific focus on identifying indicators of stress. As stress continues to be a pervasive challenge in our fast-paced world, the project seeks to contribute to proactive intervention and support by deciphering the linguistic nuances within tweets.

The objectives of the project center around the comprehensive exploration of linguistic cues that signify stress. Through a meticulous analysis of how individuals express stress, anxiety, and related emotions on Twitter, the project aims to build a model capable of recognizing subtle yet significant markers of stress. To achieve this, a diverse dataset of tweets will be collected, emphasizing posts that delve into personal experiences related to stress. This dataset will undergo thorough annotation, providing a robust foundation for training and validating the stress detection model.

The technical core of the project involves the application of advanced NLP techniques, including tokenization, sentiment analysis, and feature extraction, to process and analyze the textual content of tweets. These techniques aim to capture the nuances of language that convey stress, enabling the development of a machine learning model. This model, likely incorporating supervised learning algorithms, will be trained on the annotated dataset to predict, in real-time, whether a given tweet contains indicators of stress.

Beyond the technical dimensions, the project places paramount importance on ethical considerations. With the sensitivity of mental health data in mind, the project adheres to stringent ethical standards. This commitment includes safeguarding user privacy, obtaining informed consent, and adopting responsible data handling practices throughout the project's lifecycle. The ultimate significance of the "Stress Detection from Tweets" project lies in its potential to contribute to a nuanced understanding of mental health dynamics in the digital age. By identifying stress indicators early on, the project aims to foster a supportive environment for individuals navigating the complex landscape of stress within the online social sphere.

INDEX

List of Tables

1. INTRODUCTION

In the ever-evolving digital landscape, social media platforms have become an integral part of our daily lives, serving as virtual arenas for individuals to express their thoughts, emotions, and experiences. Among these platforms, Twitter stands out as a dynamic space where users share succinct insights into their lives in real-time. Within this vast reservoir of user-generated content lies a unique opportunity to explore the intricate interplay between language and mental well-being. The "Stress Detection from Tweets" project aims to navigate this digital frontier, employing advanced technologies such as Natural Language Processing (NLP) and machine learning to decipher the language patterns within tweets and identify potential indicators of stress.

In contemporary society, characterized by an incessant flow of information and high-paced living, stress has emerged as a prevalent concern impacting individuals globally. Understanding and addressing stress early on is pivotal for the overall well-being of individuals. Twitter, with its open and real-time nature, provides a unique lens into the immediate thoughts and feelings of users. By unraveling the linguistic nuances embedded in tweets, this project seeks to develop a model capable of discerning subtle markers of stress, thereby contributing to timely intervention, support, and a nuanced understanding of mental health in the digital age.

To achieve this goal, the project outlines several key objectives. First and foremost, the focus is on identifying stress indicators through a meticulous examination of language patterns. By analyzing how individuals express stress, anxiety, and related emotions on Twitter, we aim to build a comprehensive understanding of the diverse ways stress is articulated in the digital space. This intricate exploration of linguistic cues will be complemented by a robust data collection and annotation process, emphasizing tweets that delve into personal experiences related to stress.

The next phase involves the application of cutting-edge Natural Language Processing (NLP) techniques to preprocess and analyze the textual content of tweets. Tokenization, sentiment analysis, and feature extraction will be employed to capture the nuances of language that convey stress. Subsequently, a machine learning model, likely a combination of supervised learning algorithms, will be trained on the annotated dataset. Rigorous testing and validation will ensure the model's accuracy in predicting whether a given tweet contains indicators of stress.

The project's significance extends beyond the technical realm to ethical considerations. Given the sensitivity of mental health data, the project is committed to upholding stringent ethical standards. User privacy, informed consent, and responsible data handling practices will be prioritized throughout the project lifecycle. In summary, the "Stress Detection from Tweets" project embarks on a journey to unravel the intricate relationship between language and mental well-being in the digital era, with the ultimate aim of fostering a proactive and supportive environment for individuals navigating the complex landscape of stress.

2. LITERATURE SURVEY/ EXISTING SYSTEM

2.1 FEASIBILITY STUDY

A feasibility study on stress detection from tweets using machine learning and Python indicates promising potential. Leveraging Natural Language Processing (NLP) techniques and the Bernoulli Naive Bayes algorithm, coupled with Python's flexibility, the streamlined integration process and adaptability of Python make this solution practical for implementation. Stress detection from tweets holds feasibility and could significantly contribute to mental health awareness by providing real-time insights into users' stress levels.

2.1.1 ORGANIZATIONAL FEASIBILITY

Conducting an organizational study on stress detection from tweets reveals the strategic viability of implementing NLP techniques and machine learning. The study emphasizes the seamless integration of these technologies into existing systems, showcasing their potential to improve mental health monitoring and awareness. Recommendations include investing in training and implementation processes to capitalize on the efficiency and accuracy for effective stress detection solutions.

2.1.2 ECONOMIC FEASIBILITY

An economic study on stress detection from tweets employing NLP techniques and machine learning demonstrates cost-effectiveness. The streamlined integration and adaptability of these technologies result in efficient deployment, minimizing development and maintenance expenses. The positive economic impact of implementing technologies for stress detection is reinforced by the potential reduction in healthcare costs associated with mental health issues.

2.1.3 TECHNICAL FEASIBILITY

A technical study on stress detection from tweets using NLP techniques and machine learning underscores its robust efficacy. The chosen algorithm enables accurate analysis of language patterns indicative of stress in posts. The efficient integration process facilitates seamless implementation into existing systems, showcasing the technical feasibility of this solution.

2.1.4 BEHAVIORAL FEASIBILITY

A behavioral study on stress detection from tweets employing NLP and machine learning reveals significant positive impacts. Python scripts contribute to the seamless integration of this behavioral monitoring system into social media platforms, fostering a proactive approach to mental health awareness. Behavioral insights provided by technologies enhance the overall effectiveness of stress detection, positively influencing user behavior and mental well-being.

2.2 LITERATURE REVIEW

Social Media Text Analysis for Stress Detection Using BERT and LSTM [1]:

This research proposes a stress detection framework based on Bidirectional Encoder Representations from Transformers (BERT) and Long Short-Term Memory (LSTM) networks. BERT is employed for contextualized word embeddings, and LSTM is utilized for sequence modeling. The model is trained and evaluated on a large social media dataset, demonstrating high accuracy in stress detection. The combination of BERT and LSTM offers improved contextual understanding and sequential dependencies, making it a robust solution for stress detection in diverse social media posts.

Detecting Stress Signals in Twitter Data Using Sentiment Analysis and Machine Learning [2]:

This study explores stress detection from Twitter data through sentiment analysis and machine learning. Various sentiment analysis techniques, including lexicon-based and machine learning-based approaches, are employed to identify stress signals in tweets. The study emphasizes the significance of contextual understanding in distinguishing between positive and negative stress expressions. The proposed model achieves competitive results and provides insights into the nuanced nature of stress communication on social media platforms.

Stress Recognition in Online Communication Using Deep Learning [3]:

Focusing on stress recognition in online communication, this research introduces a deep learning-based approach. The model leverages convolutional neural networks (CNNs) and recurrent neural networks (RNNs) to capture spatial and temporal dependencies in textual data. Trained on a diverse dataset of online communication, the model exhibits promising results in stress detection. The study emphasizes the need for context-aware models to address the complexities of stress expression in different online contexts.

Investigations in Psychological Stress Detection from social media Text using Deep Architectures [4]:

The study focuses on detecting tweets containing mentions of psychological stress, particularly considering the introverted nature of the stressed community. Experimental results on a standard Twitter dataset highlight that Multichannel CNN outperforms other architectures, achieving an impressive accuracy of 97.5%. This research contributes significantly to the field of automatic stress detection from social media, providing valuable insights into the complexities of mental health expression online.

The Improvement of Stress Level Detection in Twitter: Imbalance Classification Using SMOTE [5]:

This study developed a model to improve stress level detection using Synthetic Minority Oversampling Technique (SMOTE) imbalanced data classification. SMOTE is a method to address imbalanced datasets to oversample the minority class. This research used the framework model of Data, Experts Data Annotation, Text Pre-processing, and Text Representation and Classification. The Bag of Word (BoW), Term Frequency-Inverse Document Frequency (TFIDF), and Lemma were used for the text representation. The Subject Matter Experts (SMEs) on mental health problems have annotated the text from the tweets based on four levels: Normal, Mild, Moderate, and Severe. The data group for the Normal stress level was relatively large compared to the other groups. Due to the imbalanced data group, the SMOTE technique was used for data argumentation. The result showed that the model classification using Support Vector Machine with SMOTE increased by improving the cardinality of the minority class label through the significant Macro Avg Recall and Macro Avg F1-Score analysis results compared to the baseline.

2.3 EXISTING SYSTEM

- Sentiment Analysis: These algorithms assess the emotional tone of the posts, identifying sentiment shifts that may signify stress, anxiety, or depressive expressions.
- Behavioral Indicators: Systems observe online behavior, such as posting frequency, changes in writing style, and engagement patterns, to discern signs of stress in social media users.
- Contextual Understanding: Advanced models aim to understand the context of posts, considering nuances, sarcasm, or cultural references that may impact stress interpretation accurately.
- Multimodal Approaches: Some systems integrate text, image, and audio analysis to capture a broader range of expressions and emotions related to stress in social media content.

2.4 DRAWBACKS OF THE EXISTING SYSTEM

- Lack of Contextual Understanding: Difficulty in capturing the complete context of a post may lead to misinterpretation of expressions, impacting the precision of stress detection.
- Ethical Concerns: Analyzing user-generated content for mental health signals raises ethical concerns related to privacy, consent, and potential misuse of sensitive information.
- Dynamic Language Evolution: The ever-changing nature of online language, slang, and evolving expressions necessitates continuous model adaptation to remain effective over time.
- Limited Non-Textual Features: While text-based analysis is prominent, incorporating non-textual features like image or audio sentiment could enhance the overall accuracy of stress detection.

3. SOFTWARE REQUIREMENT ANALYSIS

3.1 INTRODUCTION

Following requirement elicitation, the analysis process is crucial for creating uniform, unambiguous software requirements. This involves examining, improving, and scrutinizing the obtained requirements, providing a graphical representation of the full system.

3.1.1 DOCUMENT PURPOSE

The purpose is to guide software makers on developing industry-specific software.

3.1.2 DEFINITIONS

Machine Learning:

Machine Learning (ML) is a subset of artificial intelligence that focuses on the development of algorithms and models enabling computers to learn patterns and make decisions from data without explicit programming. The learning process involves exposure to labeled data, allowing the system to generalize and make predictions on new, unseen data.

Data Preprocessing

Data preprocessing is a process of preparing the raw data and making it suitable for a machine learning model. It is the first and crucial step while creating a machine learning model

- Getting the dataset
- Importing libraries
- Importing datasets
- Finding Missing Data
- Encoding Categorical Data
- Splitting dataset into training and test set
- Feature scaling

Label Encoder:
Features which define a category are Categorical Variables. E.g. Color (red, blue, green), Gender (Male, Female). Machine learning models expect features to be either floats or integers therefore categorical features like color, gender etc. need to be converted to numerical values. Label encoder converts categorical feature to integers

Nominal Data:
Nominal data is the simplest data type. It classifies data purely by labeling or naming values e.g. measuring marital status, hair, or eye color. It has no hierarchy to it.

3.2 SYSTEM ARCHITECTURE

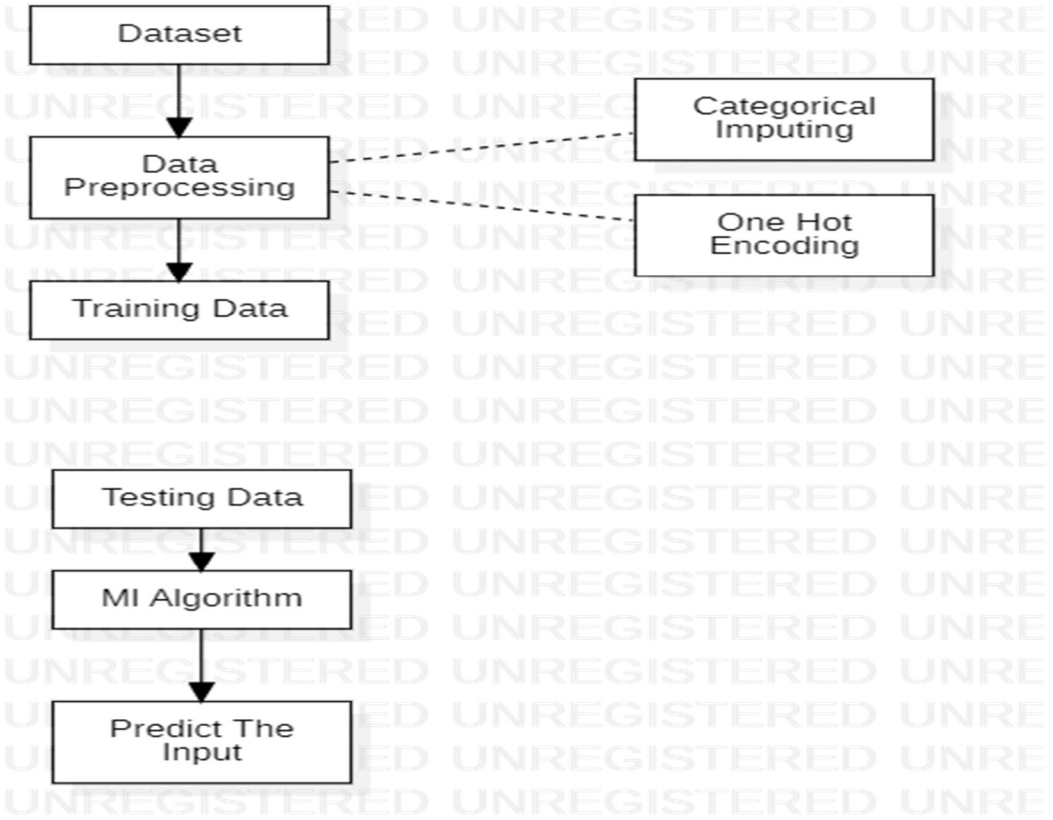


Fig 3.2.1: System Architecture of the proposed system

It is a conceptual model that describes the structure, viewpoints, and behavior of a system. A formal description and representation of a system that is organized in a way that facilitates reasoning about the system's structures and behaviors is known as an architecture description. A system architecture consists of system components and created sub-systems that work together to implement the overall system. The proposed System begins by ingesting tweets and subjecting them to data preprocessing to ensure uniformity and relevance. Extracted features are then used as input for the classification algorithm, which employs machine learning to identify stress patterns. The system compares the identified features with those stored in the database, enhancing accuracy. The stress classification results are displayed, providing insights into the percentage of posts expressing stress.

3.3 FUNCTIONAL REQUIREMENTS

- Alert Generation
- Adaptability to Different Conditions
- Behavioral Analysis
- User Interface

3.4 SYSTEM ANALYSIS

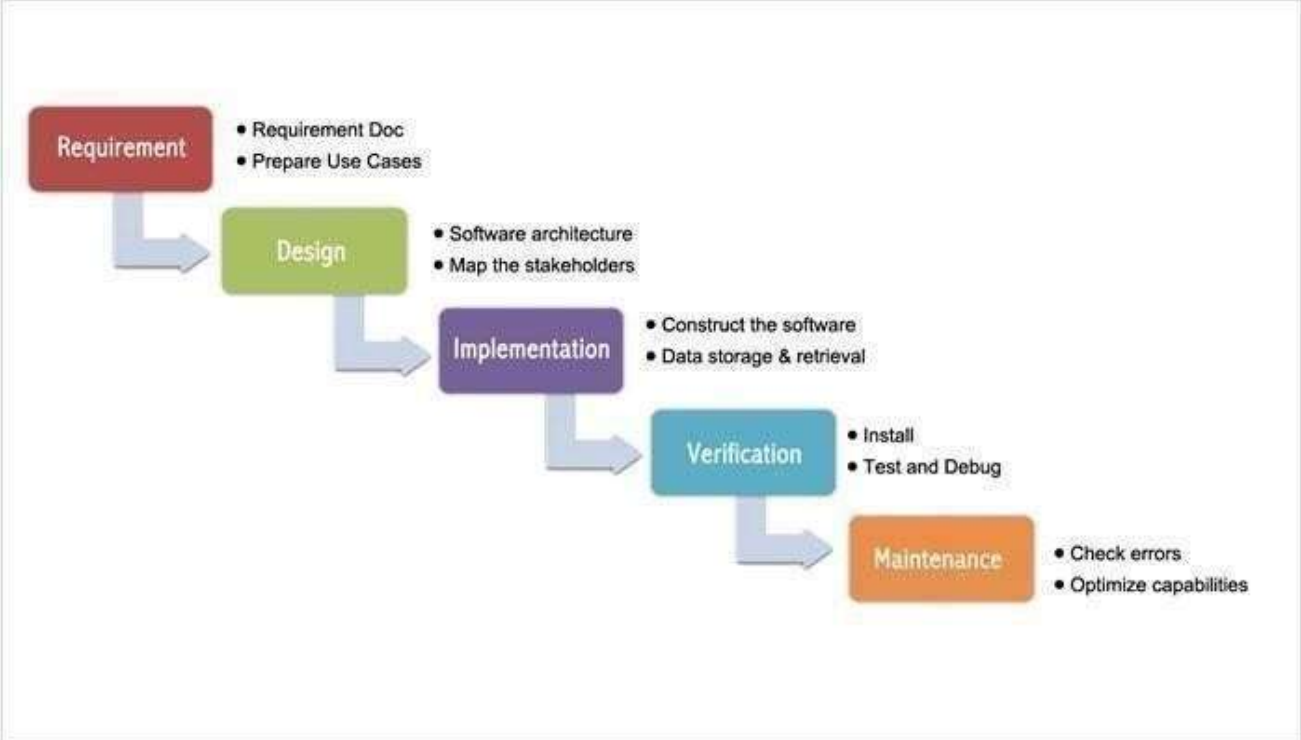


Fig.3.4.1 System Analysis of the proposed system

It was the first Process Model to be introduced. A linear sequential life cycle model is another name for it. It's quite simple to use and understand. Phases do not overlap in this paradigm, and each phase must be finished before the next one begins. The model shows that the development of software is linear and is a sequential process. Only after one phase of the development is completed, we can go to the next phase. In this waterfall paradigm, the phases do not overlap. The steps in the waterfall model are explained below.

Requirements: The search has become more intense and concentrated on the software's requirements at this time. To comprehend the nature of the programs to be developed, the software engineer must first comprehend the software's information domain, which includes the required functionalities, user interface, and so on. The customer must be informed about the second activity, which must be recorded and presented.

Design: This step is used to transform the above criteria as a representation in the form of "blueprint" software before coding begins. The design must be able to meet the criteria laid out in the previous stage.

Implementation: The design was converted into a machine-readable format in order for it to be interpreted by a computer in some circumstances, i.e., through the coding process into a programming language. This was the stage in which the programmer will put the technical design phase into action.

Verification: It, like anything else constructed, must first be put to the test. The same may be said for software. To ensure that the application is error-free, all functions must be checked, and the results must closely comply with the previously specified requirements.

Maintenance: Software maintenance, including development, is essential since the software that is being generated is not always exactly like that. It may still have minor faults that were not identified previously when it runs, or it may require additional capabilities that were not previously available in the software.

Useful factors: The Waterfall model has its advantages like it is simple to use. Additionally, while using the model all the system requirements can be defined as a whole, explicitly and at the start the product can run without many issues.

It is economic to make changes in the early stages of the project when there are problems with system requirements then when the problems which arise in later stages.

3.5 NON-FUNCTIONAL REQUIREMENTS

- **Reliability**: The system has a reliability of at least 76% in accurately identifying stress.
- **Interoperability**: The system should be designed to operate seamlessly across various social media platforms.
- **Security**: The system implements appropriate security measures to safeguard user data, ensuring the privacy of individuals expressing stress on social media.
- **Adaptability**: The system should provide users with the ability to customize stress detection parameters based on their preferences

3.6 SOFTWARE REQUIREMENT SPECIFICATION

GOOGLE COLAB

Colab is a free Jupyter notebook environment that runs entirely in the cloud. Most importantly, it does not require a setup and the notebooks that you create can be simultaneously edited by your team members - just the way you edit documents in Google Docs. Colab supports many popular machine learning libraries which can be easily loaded in your notebook

- Write and execute code in Python
- Document your code that supports mathematical equations
- Create/Upload/Share notebooks
- Import/Save notebooks from/to Google Drive
- Import external datasets e.g. from Kaggle
- Free Cloud service with free GPU

3.6 SOFTWARE REQUIREMENTS

- Software : Google colab
- Operating System : Windows family
- Technology : Machine Learning

3.7 HARDWARE REQUIREMENTS

- Minimum 8GB Ram Laptop
- Internet Connection

4. SOFTWARE DESIGN

4.1 UML DIAGRAMS

Unified Modeling Language (UML) diagrams are visual representations used in software engineering to model and document the design and structure of a system. UML provides a standardized set of diagrams that h elp visualize different aspects of a system's architecture, behavior, and interactions. The components are similar to modules that can be combined ina variety of ways to create a complete UML diagram. As a result, comprehension of the various diagrams is essential for utilizing the knowledge in real-world systems. The best method to understand any complex system is to draw diagrams or images of it. These designs have a bigger influence on our understanding. Looking around, we can see that info-graphics are not a new concept, but they are frequently utilized in a variety of businesses in various ways.

User Model View

The perspective refers to the system from the clients' point of view. The exam's depiction depicts a situation of utilization from the perspective of end-clients. The user view provides a window into the system from the perspective of the user, with the system's operation defined in light of the user and what the user wants from it.

Structural model view

This layout represents the details and functionality of the device. This software design maps out the static structures. This view includes activity diagrams, sequence diagrams and state machine diagrams

Behavioral Model View

It refers to the social dynamics as framework components, delineating the assortment cooperation between various auxiliary components depicted in the client model and basic model view. UML Behavioral Diagrams illustrate time-dependent aspects of a system and communicate the system's dynamics and how they interact. Behavioral diagrams include interaction diagrams, use case diagrams, activity diagrams and state–chart diagrams.

Implementation Model View

The essential and actions as frame pieces are discussed in this when they are to be manufactured. This is also referred to as the implementation view. It uses the UML Component diagram to describe system components. One of the UML diagrams used to illustrate the development view is the Package diagram.

Environmental Model View

The systemic and functional component of the world where the program is to be introduced was expressed within this. The diagram in the environmental view explains the software model's after-deployment behavior. This diagram typically explains user interactions and the effects of software on the system. The following diagrams are included in the environmental model: Diagram of deployment.

The UML model is made up of two separate domains:

- Demonstration of UML Analysis, with a focus on the client model and auxiliary model perspectives on the framework.
- UML configuration presenting, which focuses on demonstrations, usage, and natural model perspectives.

4.1.1 USE CASE DIAGRAM

The objective of a use case diagram is to portray the dynamic nature of a system. However, because the aim of the other four pictures is the same, this description is too broad to characterize the purpose. We'll look into a specific purpose that distinguishes it from the other four diagrams.

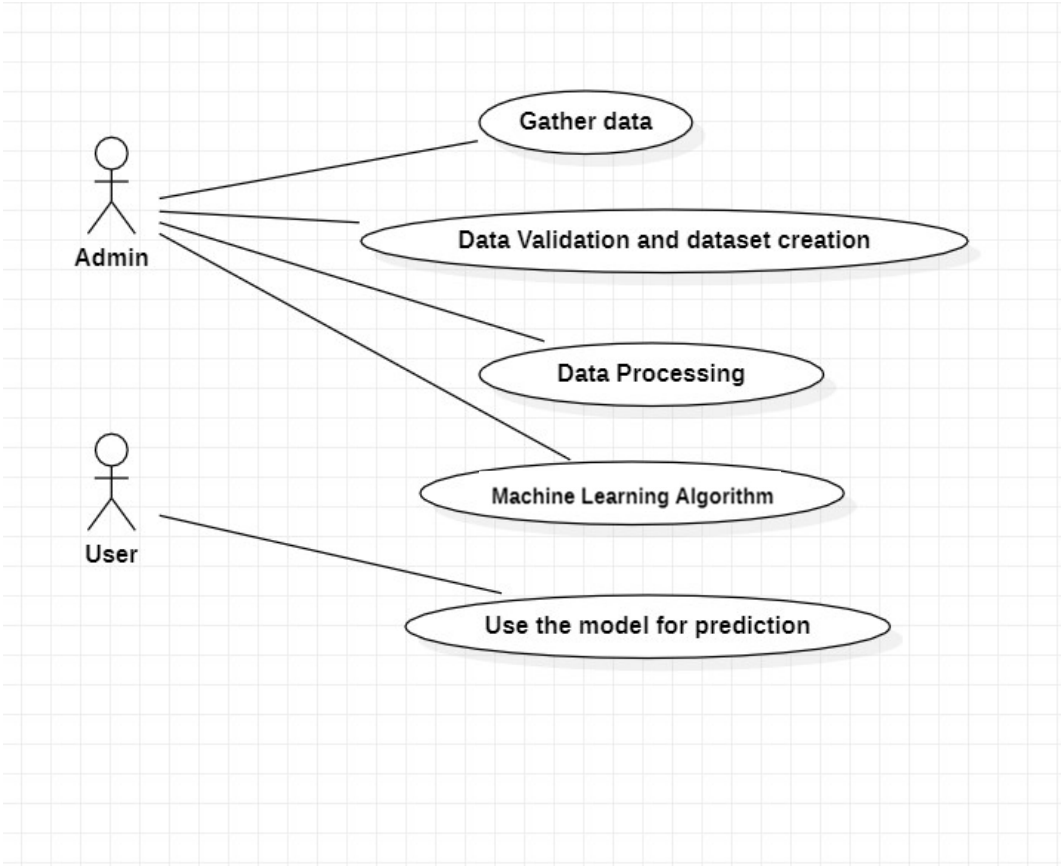


Fig 4.1.1.1: Use Case diagram for Stress Detection Model.

Actors:

- Admin
- User

Use Cases:

- Collect Data
- Data Validation And Data Set Creation
- Data Processing
- Machine Learning Algorithm
- Use The Model For Prediction

Connections:

- Admin must gather data required to choose which algorithm to be used
- Then the admin will be processing the data and and choose the algorithm based on accuracy
- User will be using the efficient model for prediction on the data

4.1.2 SEQUENCE DIAGRAM

Because it illustrates how a group of items interact with one another, a sequence diagram is a form of interaction diagram. These diagrams are used by software engineers and business people to comprehend the requirements for a new system or to document a current process. Sequence diagrams are sometimes known as event diagrams or event scenarios. Sequence diagrams can be useful as a reference for businesses and other organizations. Make the diagram to show:

- Describe the specifics of a UML use case.
- Create a model of the logic of a complex procedure, function, or operation.
- Examine how objects and components interact with one another in order to complete a process.
- Plan and comprehend the specific functionality of a current or future scenario.

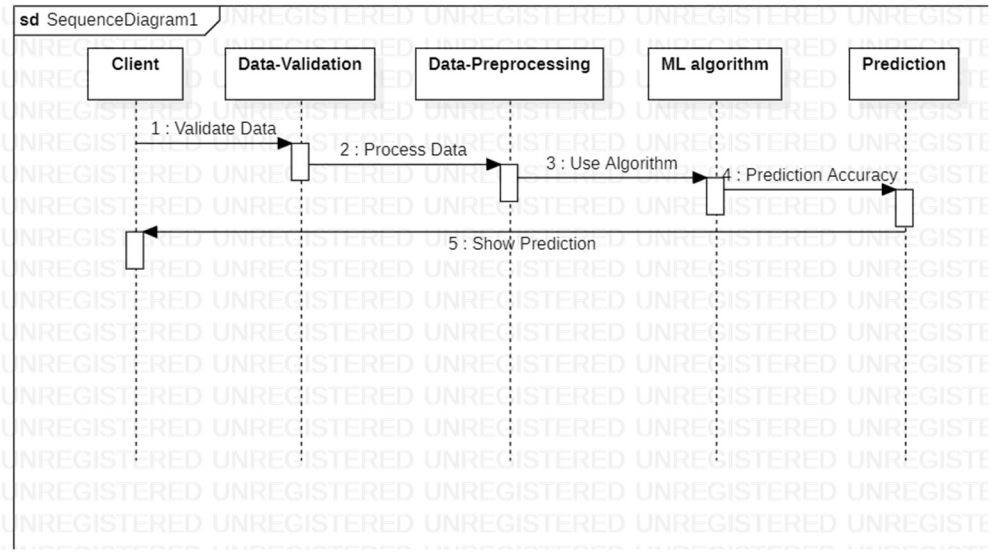


Fig 4.1.2.1: Sequence Diagram for Stress Detection Model

In the above sequence diagram, the lifelines are:

- Client
- Data-Validation
- Data-Preprocessing
- ML Algorithm
- Prediction

The sequence starts from the client sending the data required for the prediction process that includes data-validation, data-preprocessing.

4.1.3 ACTIVITY DIAGRAM

An activity diagram is a flowchart that displays the movement of information from one action to the next. A system operation can be used to describe the activity.

- Admin must gather data required to choose which algorithm to be used
- Then the admin will be processing the data and choose the algorithm based on accuracy
- User will be using the efficient model for prediction on the data

Use Cases:

- Collect Data
- Data Validation and Data Set Creation
- Data Processing
- Machine Learning Algorithm
- Use the Model for Prediction

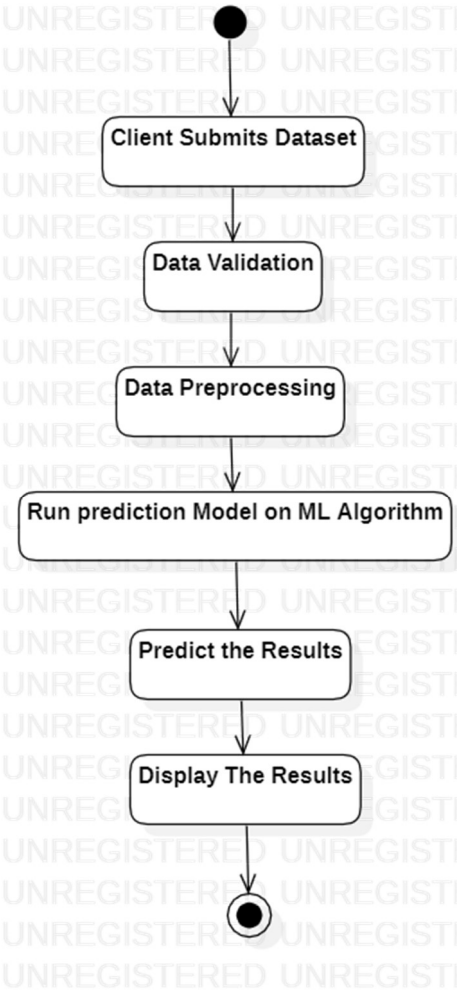


Fig 4.1.3.1: Activity Diagram for Stress Detection Model

This activity diagram shows the whole activity of the system. The Activity starts with the client submitting the data required for prediction and data validation, data preprocessing followed by predicting the results.

4.1.4 CLASS DIAGRAM

A static diagram is also referred to as a class diagram. It depicts the static view of an application. A class diagram can be used to visualize, describe, and document various parts of a system, as well as to create executable code for a software programmer.

The traits and activities of a class, as well as the constraints, are described in a class diagram.

- Admin must gather data required to choose which algorithm to be used.
- Then the admin will be processing the data and choose the algorithm based on accuracy.
- User will be using the efficient model for prediction on the data.

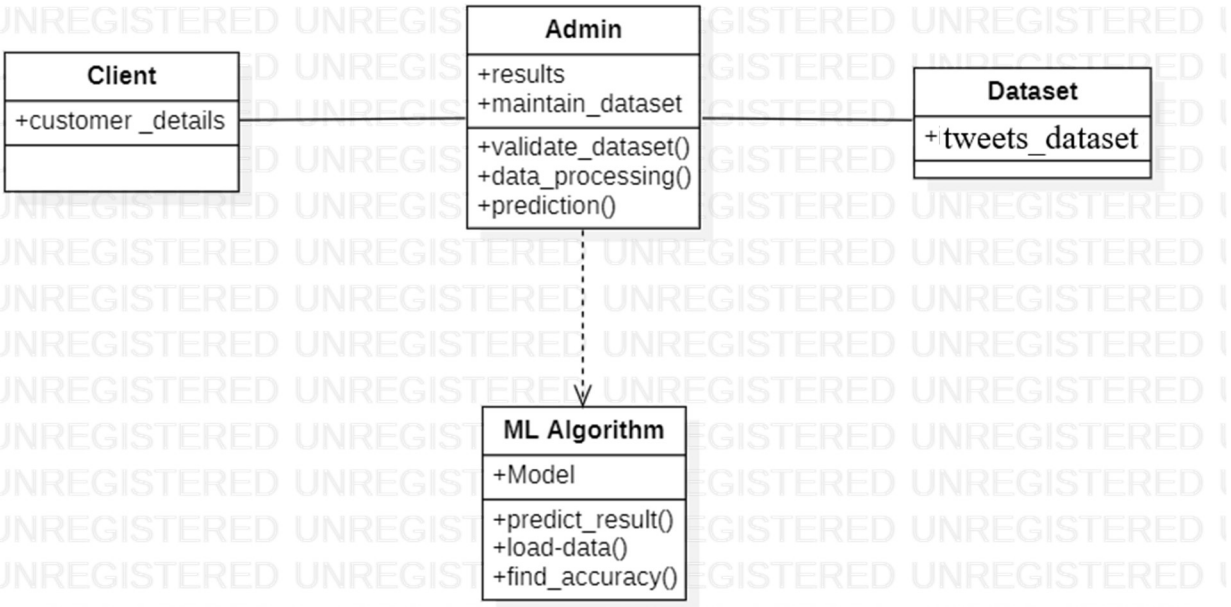


Fig 4.1.4.1: Class Diagram for Stress Detection Model.

The main class in this class diagram is responsible for coordinating the stress detection process. It holds text data from social media posts and the label class labels the result as stress and no stress.

5 PROPOSED SYSTEM

5.1 METHODOLOGY

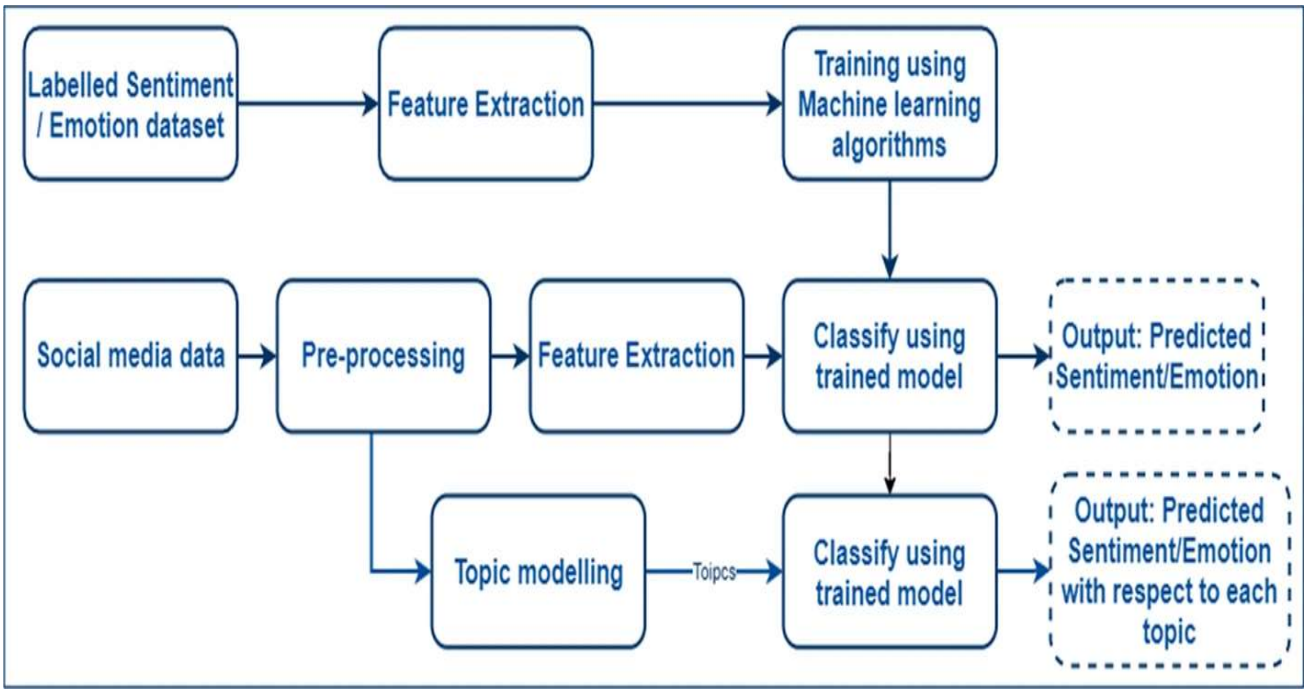


Fig:5.1.1 Methodology of the proposed system

Data Set:

Data is crucial in finding patterns and using that pattern to predict the outcome in our case finding the drowsiness of the driver. For our project we have collected a data set from kaggle and trained the model.

Cleaning:

We employed Natural Language Processing (NLP) techniques for data cleaning. The NLP tools automatically clean and preprocess the data when uploaded to our system.

Models:

Machine Learning is a fundamental component of our stress detection system. It involves training artificial neural networks, inspired by the human brain, to analyze language patterns and identify stress indicators in social media posts.

In our project, we explored different machine learning technologies including:

- Logistic Regression
- Linear SVC
- Random Forest
- Bernoulli Naïve Bayes

Train and Test:

Machine Learning is one of the booming technologies across the world that enables computers/machines to turn a huge amount of data into predictions. However, these predictions highly depend on the quality of the data, and if we are not using the right data for our model, then it will not generate the expected result. In machine learning projects, we generally divide the original dataset into training data and test data. We train our model over a subset of the original dataset, i.e., the training dataset, and then evaluate whether it can generalize well to the new or unseen dataset or test set. Therefore, train and test datasets are the two key concepts of machine learning, where the training dataset is used to fit the model, and the test dataset is used to evaluate the model.

5.2 FUNCTIONALITIES

5.2.1 Collecting Data:

- We collected tweets dataset from kaggle.

5.2.2 Preprocessing data:

- Removing inconsistencies in data like null values and outliers
- dealing with categorical data

5.2.3 Choosing Model:

- Test each model and choose best model according to accuracies

5.2.4 Detect drowsiness of a driver:

- return 0 if No Stress,1 if the person is in Stress.

5.3 ADVANTAGES OF PROPOSED SYSTEM:

- Easy to maintain large tweets datasets.
- Takes less time as compared to traditional methods
- Less human effort.
- More Accurate.

6. CODING AND IMPLEMENTATIONS

6.1 Dataset:

It is used to train the model. The model learns from this data set and gives the outputs based on the learning during the testing. We use a dataset on Kaggle with 116 columns. We only need to use the text and label column for this task. The dataset contains data posted on tweets related to mental health. This dataset contains various mental health problems shared by people about their life. Fortunately, this dataset is labelled as 0 and 1, where 0 indicates no stress and 1 indicates stress.

6.2 Understanding Data

Initially we imported all libraries which are required like pandas, matplotlib, seaborn etc. Data set is in the CSV format. Data consists of both categorical and numerical data. Some columns have null values which as to be preprocessed.

```
import pandas as pd
import numpy as np
data = pd.read_csv("/content/dreaddit-train.csv")
print(data.head())
```

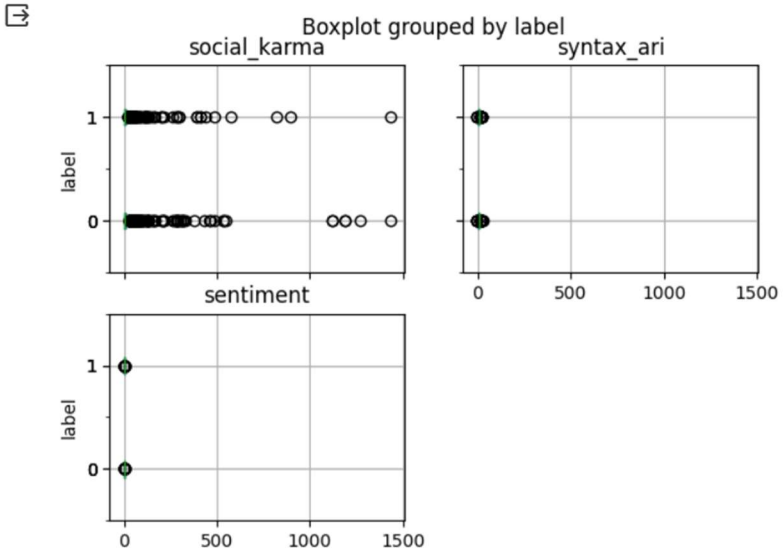
```
          subreddit post_id sentence_range  \
0             ptsd  8601tu       (15, 20)
1       assistance  81brx9         (0, 5)
2             ptsd  9ch1zh       (15, 20)
3    relationships  7rorpp        [5, 10]
4  survivorsofabuse  9p2gbc         [0, 5]

                                                text     id  label  \
0  He said he had not felt that way before, sugge...  33181      1
1  Hey there r/assistance, Not sure if this is th...   2606      0
2  My mom then hit me with the newspaper and it s...  38816      1
3  until i met my new boyfriend, he is amazing, h...    239      1
4  October is Domestic Violence Awareness Month a...   1421      1

   confidence  social_timestamp  social_karma  syntax_ari  ...  \
0         0.8        1521614353             5    1.806818  ...
1         1.0        1527009817             4    9.429737  ...
2         0.8        1535935605             2    7.769821  ...
3         0.6        1516429555             0    2.667798  ...
4         0.8        1539809005            24    7.554238  ...

   lex_dal_min_pleasantness  lex_dal_min_activation  lex_dal_min_imagery  \
0                     1.000                  1.1250                  1.0
1                     1.125                  1.0000                  1.0
2                     1.000                  1.1429                  1.0
3                     1.000                  1.1250                  1.0
4                     1.000                  1.1250                  1.0

   lex_dal_avg_activation  lex_dal_avg_imagery  lex_dal_avg_pleasantness  \
0                 1.77000              1.52211                   1.89556
1                 1.69586              1.62045                   1.88919
2                 1.83088              1.58108                   1.85828
3                 1.75356              1.52114                   1.98848
4                 1.77644              1.64872                   1.81456

   social_upvote_ratio  social_num_comments  syntax_fk_grade  sentiment
0                 0.86                    1         3.253573  -0.002742
1                 0.65                    2         8.828316   0.292857
2                 0.67                    0         7.841667   0.011894
3                 0.50                    5         4.104027   0.141671
4                 1.00                    1         7.910952  -0.204167

[5 rows x 116 columns]
```

```
print(data.isnull().sum())
```

```
subreddit                    0
post_id                      0
sentence_range               0
text                         0
id                           0
                            ..
lex_dal_avg_pleasantness     0
social_upvote_ratio          0
social_num_comments          0
syntax_fk_grade              0
sentiment                    0
Length: 116, dtype: int64
```

So this dataset does not have any null values.

```
data.boxplot(column=['social_karma', 'syntax_ari', 'sentiment'], by='label', vert=False)
plt.title('Box Plot of Numeric Features by Label')
plt.show()
```



A boxplot is a graphical representation that displays the distribution of a dataset. It provides a visual summary of the minimum, first quartile, median, third quartile, and maximum of a set of data. The resulting plot provides insights into the distribution of the specified numeric features across different labels, allowing for visual comparisons of the central tendency and spread of each feature within different categories. This type of visualization is useful for identifying patterns, variations, and potential outliers in the dataset concerning the specified numeric features and labels.

6.3 Data Preprocessing:

Data preprocessing is a process of preparing the raw data and making it suitable for a machine learning model. It is the first and crucial step while creating a machine learning model. When creating a machine learning project, it is not always a case that we come across the clean and formatted data. And while doing any operation with data, it is mandatory to clean it and put in a formatted way.

Some preprocessing tasks we are going to do are:

1. removing unwanted features
2. replacing a feature with more meaning full feature
3. removing or replacing null(?) values
4. Dealing with Categorical data

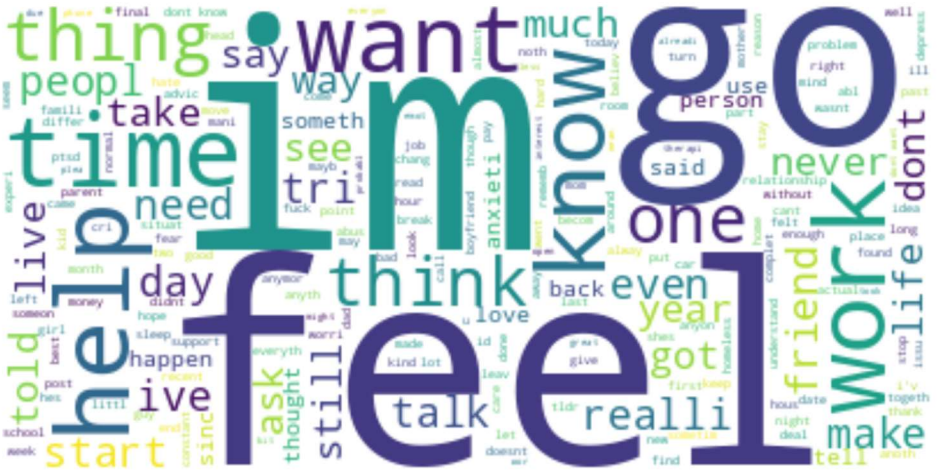
Let's prepare the text column of this dataset to clean the text column with stopwords, links, special symbols and language errors

```
import nltk
import re
from nltk.corpus import stopwords
from nltk import FreqDist
import matplotlib.pyplot as plt

nltk.download("stopwords")
stemmer = nltk.SnowballStemmer("english")
from nltk.corpus import stopwords
import string
stopword=set(stopwords.words('english'))

def clean(text):
    text = str(text).lower()
    text = re.sub('\[.*?\]', '', text)
    text = re.sub('https?://\S+|www\.\S+', '', text)
    text = re.sub('<.*?>+', '', text)
    text = re.sub('[%s]' % re.escape(string.punctuation), '', text)
    text = re.sub('\n', '', text)
    text = re.sub('\w*\d\w*', '', text)
    text = [word for word in text.split(' ') if word not in stopword]
    text=" ".join(text)
    text = [stemmer.stem(word) for word in text.split(' ')]
    text=" ".join(text)
    return text
data["text"] = data["text"].apply(clean)
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
```

Now let's have a look at the most used words by the people sharing about their life problems on social media by visualizing a word cloud of the text column

```
!pip install wordcloud
import matplotlib.pyplot as plt
from wordcloud import WordCloud, STOPWORDS, ImageColorGenerator
text = " ".join(i for i in data.text)
stopwords = set(STOPWORDS)
wordcloud = WordCloud(stopwords=stopwords,
                      background_color="white").generate(text)
plt.figure( figsize=(15,10))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.show()
```

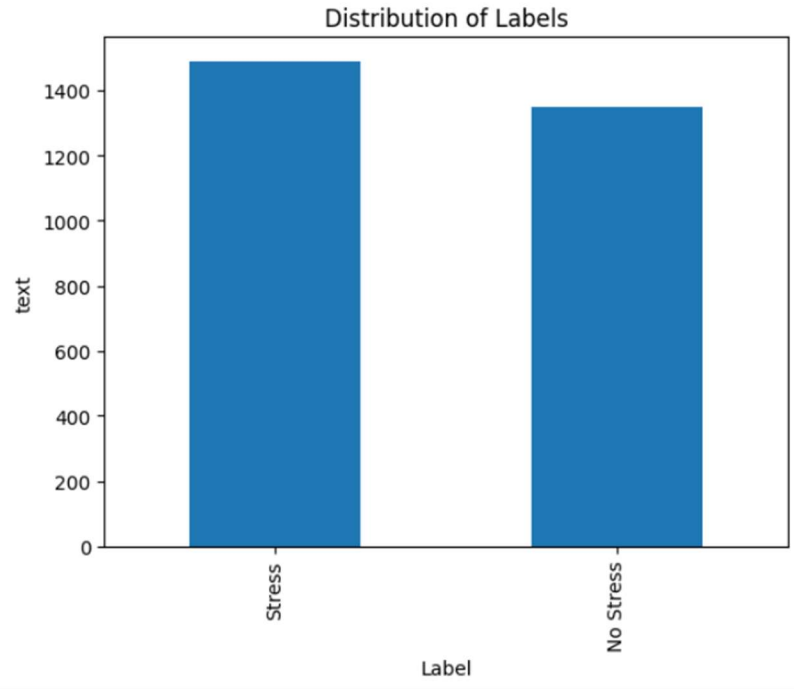


The label column in this dataset contains labels as 0 and 1. 0 means no stress, and 1 means stress. I will use Stress and No stress labels instead of 1 and 0. So let's prepare this column accordingly and select the text and label columns for the process of training a machine learning model.

```
import numpy as np
#dataset is labelled as 0 and 1, where 0 indicates no stress and 1 indicates stress.
data["label"] = np.select([data["label"] == 0, data["label"] == 1], ["No Stress", "Stress"])
data = data[["text", "label"]]
print(data.head())
```

```
                                                text      label
0  said felt way sugget go rest trigger ahead you...     Stress
1  hey rassist sure right place post goe  im curr...  No Stress
2  mom hit newspap shock would know dont like pla...     Stress
3  met new boyfriend amaz kind sweet good student...     Stress
4  octob domest violenc awar month domest violenc...     Stress
```

```
data['label'].value_counts().plot(kind='bar')
plt.title('Distribution of Labels')
plt.xlabel('Label')
plt.ylabel('text')
plt.show()
```



The plot(kind='bar') function is used to create a bar plot, which is a common visualization for displaying the count or frequency of categorical data.

As this is based on the problem of binary classification, I will be using the Bernoulli Naive Bayes algorithm, which is one of the best algorithms for binary classification problems. So let's train the stress detection model.

6.4 Choosing model:

```
[ ] from sklearn.feature_extraction.text import CountVectorizer
    from sklearn.model_selection import train_test_split

    x = np.array(data["text"])
    y = np.array(data["label"])

    cv = CountVectorizer()
    X = cv.fit_transform(x)
    xtrain, xtest, ytrain, ytest = train_test_split(X, y, test_size=0.33, random_state=42)
```

```
[ ] from sklearn.naive_bayes import BernoulliNB
    model = BernoulliNB()
    model.fit(xtrain, ytrain)
```

```
▾ BernoulliNB
BernoulliNB()
```

```
user = input("Enter a Text: ")
data = cv.transform([user]).toarray()
output = model.predict(data)
print(output)
```

```
Enter a Text: i am feeling stress alot
['Stress']
```

Count Vectorizer: Converts a collection of text documents into a matrix of token counts. Represents each document as a vector of word frequencies.

- Captures the bag-of-words representation, emphasizing word occurrence.
- Handles high-dimensional and sparse text data efficiently

```
[ ] from sklearn import svm
    SVCmodel = svm.LinearSVC()
    SVCmodel.fit(xtrain, ytrain)
```

```
▾ LinearSVC
LinearSVC()
```

```
[ ] user = input("Enter a Text: ")
    data1 = cv.transform([user]).toarray()
    output1 = SVCmodel.predict(data1)
    print(output1)
```

```
Enter a Text: i am not feeling stress i am happy
['No Stress']
```

```
from sklearn.svm import SVC

SVCmodel = SVC(kernel='linear', probability=True)
SVCmodel.fit(xtrain, ytrain)
output2 = SVCmodel.predict_proba(xtest)
print(output2)
```

```
[[0.70824647 0.29175353]
 [0.7268869  0.2731131 ]
 [0.38139055 0.61860945]
 ...
 [0.50721519 0.49278481]
 [0.82342637 0.17657363]
 [0.62100918 0.37899082]]
```

6.5 Training and Testing data

```
from sklearn.metrics import accuracy_score, classification_report

# Naive Bayes evaluation
nb_predictions = model.predict(xtest)
print("Naive Bayes Classification Report:\n", classification_report(ytest, nb_predictions))

# Evaluate the model
accuracy = accuracy_score(ytest, nb_predictions)
print(f"NB Accuracy: {accuracy:.2f}\n")

# LinearSVC evaluation
svc_predictions = SVCmodel.predict(xtest)
print("LinearSVC Classification Report:\n", classification_report(ytest, svc_predictions))
# Evaluate the model
accuracy = accuracy_score(ytest, svc_predictions)
print(f"SVM Accuracy: {accuracy:.2f}")
```

```
Naive Bayes Classification Report:
               precision    recall  f1-score   support

   No Stress       0.79      0.64      0.70       444
      Stress       0.72      0.85      0.78       493

    accuracy                           0.75       937
   macro avg       0.75      0.74      0.74       937
weighted avg       0.75      0.75      0.74       937

NB Accuracy: 0.75

LinearSVC Classification Report:
               precision    recall  f1-score   support

   No Stress       0.68      0.65      0.66       444
      Stress       0.69      0.72      0.71       493

    accuracy                           0.69       937
   macro avg       0.69      0.68      0.68       937
weighted avg       0.69      0.69      0.69       937

SVM Accuracy: 0.69
```

For Naive Bayes:
Accuracy: Achieved an accuracy of 75%.
Precision-Recall: Balanced precision and recall for both classes.
Advantages:

- Efficient, especially for high-dimensional text data.
- Simple and computationally inexpensive.

For Linear Support Vector Classifier (Linear SVC):

Accuracy: Achieved an accuracy of 69%.
Precision-Recall: Balanced precision and recall, slightly lower than Naive Bayes.
Advantages:

- Effective in high-dimensional spaces.
- Handles non-linear relationships through the kernel trick (not explicitly used here).

```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report
# Train a Logistic Regression model
LRmodel = LogisticRegression()
LRmodel.fit(xtrain, ytrain)

# Make predictions on the test set
predictions = LRmodel.predict(xtest)



# Evaluate the model
accuracy = accuracy_score(ytest, predictions)
print(f"LR Accuracy: {accuracy:.2f}")

# Print classification report for more detailed evaluation
print("Classification Report:")
print(classification_report(ytest, predictions))
```

```
LR Accuracy: 0.72
Classification Report:
              precision    recall  f1-score   support

   No Stress       0.71      0.68      0.69       444
      Stress       0.72      0.75      0.73       493

    accuracy                           0.72       937
   macro avg       0.71      0.71      0.71       937
weighted avg       0.71      0.72      0.71       937
```

Logistic Regression:

Accuracy: Achieved an accuracy of 72%.
Precision-Recall: Balanced precision and recall, competitive with Naive Bayes.
Advantages: Provides probabilities for predictions. Interpretable coefficients.

```
[ ] from sklearn.ensemble import RandomForestClassifier
    random=RandomForestClassifier()
```

```
random .fit(xtrain,ytrain)
# Make predictions on the test set
predictions = random.predict(xtest)



# Evaluate the model
accuracy = accuracy_score(ytest, predictions)
print(f"Randomforest Accuracy: {accuracy:.2f}")

# Print classification report for more detailed evaluation
print("Classification Report:")
print(classification_report(ytest, predictions))
```

```
Randomforest Accuracy: 0.71
Classification Report:
              precision    recall  f1-score   support

   No Stress       0.73      0.62      0.67       444
      Stress       0.70      0.79      0.74       493

    accuracy                           0.71       937
   macro avg       0.71      0.71      0.71       937
weighted avg       0.71      0.71      0.71       937
```

The best parameters found for the model are:
Alpha for Bernoulli Naive Bayes: 0.5
N-gram range for Count Vectorizer: (1, 2)
The best accuracy achieved with these parameters is approximately 75.93%.

7. TESTING

In machine learning, testing is mainly used to validate raw data and check the ML model's performance. The main objectives of testing machine learning models are:

- Quality Assurance
- Detect bugs and flaws

Once your machine learning model is built (with your training data), you need unseen data to test your model. This data is called testing data, and you can use it to evaluate the performance and progress of your algorithms' training and adjust or optimize it for improved results.

Testing data has two main criteria. It should:

- Represent the actual dataset
- Be large enough to generate meaningful predictions

7.1 TYPES OF TESTING

7.1.1 MANUAL TESTING

Manual Testing is a type of software testing in which test cases are executed manually by a tester without using any automated tools. The purpose of Manual Testing is to identify the bugs, issues, and defects in the software application. Manual software testing is the most primitive technique of all testing types and it helps to find critical bugs in the software application.

Any new application must be manually tested before its testing can be automated. Manual Software Testing requires more effort but is necessary to check automation feasibility. Manual Testing concepts does not require knowledge of any testing tool. One of the Software Testing Fundamental is "**100% Automation is not possible**". This makes Manual Testing imperative.

7.1.2 AUTOMATED TESTING

Automation Testing is a software testing technique that performs using special automated testing software tools to execute a test case suite. On the contrary, Manual Testing is performed by a human sitting in front of a computer carefully executing the test steps.

The automation testing software can also enter test data into the System Under Test, compare expected and actual results and generate detailed test reports. Software Test Automation demands considerable investments of money and resources.

7.2 TESTING LEVELS

7.2.1 NON-FUNCTIONAL TESTING

Non-functional testing is a type of software testing to test non-functional parameters such as reliability, load test, performance and accountability of the software. The primary purpose of non-functional testing is to test the reading speed of the software system as per non-functional parameters. The parameters of non-functional testing are never tested before the functional testing. Non-functional testing is also very important as functional testing because it plays a crucial role in customer satisfaction.

7.2.1.1 PERFORMANCE TESTING

Performance testing is a form of software testing that focuses on how a system running the system performs under a particular load. This is not about finding software bugs or defects. Different performance testing types measure according to benchmarks and standards. Performance testing gives developers the diagnostic information they need to eliminate bottlenecks.

7.2.1.2 STRESS TESTING

Stress Testing is a type of software testing that verifies stability & reliability of software application. The goal of Stress testing is measuring software on its robustness and error handling capabilities under extremely heavy load conditions and ensuring that software doesn't crash under crunch situations. It even tests beyond normal operating points and evaluates how software works under extreme conditions.

7.2.1.3 SECURITY TESTING

Security Testing is a type of Software Testing that uncovers vulnerabilities of the system and determines that the data and resources of the system are protected from possible intruders. It ensures that the software system and application are free from any threats or risks that can cause a loss. Security testing of any system is focused on finding all possible loopholes and weaknesses of the system which might result in the loss of information or repute of the organization.

7.2.1.4 PORTABILITY TESTING

Portability Testing is one of Software Testing which is carried out to determine the degree of ease or difficulty to which a software application can be effectively and efficiently transferred from one hardware, software or environment to another one. The results of portability testing are measurements of how easily the software component or application will be integrated into the environment and then these results will be compared to the non-functional requirement of portability of the software system.

7.2.1.5 USABILITY TESTING

Usability Testing, also known as User Experience (UX) Testing, is a testing method for measuring how easy and user-friendly a software application is. A small set of target end-users, use software applications to expose usability defects. Usability testing mainlyfocuses on the user's ease of using application, flexibility of application to handle controls and ability of application to meet its objectives. This testing is recommended during the initial design phase of SDLC, which gives more visibility on the expectations of the users.

7.2.2 FUNCTIONAL TESTING

It is a type of software testing which is used to verify the functionality of the software application, whether the function is working according to the requirement specification. In functional testing, each function is tested by giving the value, determining the output, and verifying the actual output with the expected value. Functional testing performed as black-box testing which is presented to confirm that the functionality of an application or system behaves as we are expecting. It is done to verify the functionality of the application. Functional testing is also called as black-box testing.

7.3 TEST CASES

Sl.no	Testcase	Expected Result	Actual Result	Pass/ Fail
1.	After Training testing with test sample data	most of the outcomes are right upto 76% accuracy	detect stress or no stress	PASS

8. RESULTS

In the context of stress detection systems, when the user is determined to be in a normal or unstressed state, the system's output signifies typical behavior and mental well-being. This might include consistent patterns in online communication, positive language use, and stable sentiment analysis. The system aims to establish a baseline for emotional well-being and remains passive during periods when the user is deemed normal and composed.

```
[ ]  user = input("Enter a Text: ")
     data1 = cv.transform([user]).toarray()
     output1 = SVCmodel.predict(data1)
     print(output1)

     Enter a Text: i am not feeling stress i am happy
     ['No Stress']
```

Fig.8.1 Output for No Stress

In stress detection systems, the output for detecting stressed states often involves alerts or warnings to the user. This can include visual cues, such as color changes or on-screen notifications, auditory alerts like alarms or notification sounds, and sometimes haptic feedback through device vibrations. The system aims to prompt the user to take corrective actions, such as engaging in stress-relief activities or seeking support, to enhance overall well-being.

```
[ ]  user = input("Enter a Text: ")
     data = cv.transform([user]).toarray()
     output = model.predict(data)
     print(output)

     Enter a Text: i am feeling stress alot
     ['Stress']
```

Fig.8.1 Output for Stress

we can see good results from our machine learning model. This is how you can train a stress detection model to detect stress from social media posts. This machine learning model can be improved by feeding it with more data.

9. CONCLUSION AND FURTHER WORK

In summary, the integration of NLP techniques and Bernoulli Naive Bayes algorithm for psychological stress detection from social media posts demonstrates a robust and effective solution, supported by the technical efficacy of the chosen algorithm and the flexibility of Python. The accuracy in analyzing language patterns indicative of stress, coupled with the efficient integration process, showcases the technical feasibility of this approach.

Future work in this domain could focus on enhancing the model's ability to detect stress patterns in diverse contexts and across different demographic groups. Exploring the integration of additional features or considering hybrid models involving multiple machine learning techniques may further improve detection accuracy. Collaboration with mental health professionals and experts in linguistics could provide valuable insights for refining the model and its applications.

Continued research might also address ethical considerations, ensuring responsible and unbiased deployment of stress detection systems. Privacy concerns and the potential impact on individuals' mental health should be carefully considered in the development and implementation of such technologies.

In conclusion, the combined use of NLP techniques and the Bernoulli Naive Bayes algorithm, implemented in Python, offers a strong foundation for stress detection from social media posts. Ongoing research and development efforts can contribute to the evolution of this technology for better mental health awareness and support.

REFERENCES

[1]. A Novel Text Mining Approach for Mental Health Prediction Using Bi-LSTM and BERT Model [(accessed on 03 March 2022)] [Google Scholar]
[2]. Stress detection using natural language processing and machine learning over social interactions [(accessed on 20 March 2022)] [Google Scholar]
[3]. Daily Stress and Mood Recognition System Using Deep Learning and Fuzzy Clustering for Promoting Better Well-Being [(accessed on 07 Mar 2019)]; Available on IEEE **DOI:** 10.1109/ICCE.2019.8661932
[4]. Investigations in Psychological Stress Detection from social media Text using Deep Architectures [IEEE] **DOI** : 10.1109/ICPR56361.2022.9956639
[5]. The Improvement of Stress Level Detection in Twitter: Imbalance Classification Using SMOTE [(accessed on 14 Nov 2022)]. Available on IEEE
DOI: 10.1109/ICOCO56118.2022.10031684