**PROJECT TITLE**

**DECODING EMOTION THROUGH SENTIMENT**

**ANALYSIS OF SOCIAL MEDIA CONVERSATION**

**Phase-3**

**Student Name: ARUNKUMAR P**

**Register Number: 412723104009**

**Institution: Tagore Engineering College Department: Computer Science**  **Date of Submission:14/05/2025**

**Github Repository Link:**

**https://github.com/Aklite/Sentimental-analysis**

# Problem Statement

In the era of digital communication, social media platforms have become a powerful medium where individuals freely express their thoughts, emotions, and opinions. These platforms generate an enormous amount of user-generated content daily, which holds valuable insights into public sentiment on various topics, events, and trends. However, understanding the emotional tone embedded within these large volumes of unstructured text data is a complex task.

Our project, **“Decoding Emotion Through Sentiment Analysis of Social Media Conversation,”** aims to address this challenge by analyzing social media text to detect and classify emotions such as happiness, anger, sadness, surprise, fear, etc. The primary objective is to develop a machine learning-based sentiment analysis model that can automatically interpret and categorize emotional states from social conversations.

This is a **multi-class classification problem**, as the goal is to assign each text instance to one of several predefined emotion categories. By solving this problem, we can provide insights for various real-world applications including mental health monitoring, brand reputation management, targeted advertising, and public opinion tracking.

Emotion detection from textual data not only helps businesses make informed decisions but also contributes to societal well-being by identifying emotional distress in individuals or communities, enabling timely interventions.

# Abstract

This project focuses on analyzing emotions expressed in social media conversations using sentiment analysis. By applying Natural Language Processing (NLP) and machine learning techniques, we classify user posts into emotional categories like joy, anger, sadness, and more. The system helps uncover public mood and opinion trends, offering valuable insights for businesses, governments, and researchers. It showcases the power of AI in understanding human emotions from online text data.

# System Requirements

To run this emotion detector system efficiently, the following hardware and software configurations are recommended:

Hardware Requirements

**RAM**: Minimum 8 GB (recommended 16 GB for smoother performance during model training)

**Processor**: Intel i5/i9 or AMD Ryzen 5/7 (or any equivalent with multiple cores)

**Storage**: At least 2 GB of free space for datasets, libraries, and model files

**GPU (optional but beneficial**): NVIDIA GPU (if using transformer-based sentiment models)

**Software Requirements**

**Programming Language**: Python 3.8 or above

**IDE/Notebook:** Google Colab or Jupyter Notebook

**Libraries:**

pandas, numpy – for data manipulation

seaborn, matplotlib, plotly – for data visualization

scikit-learn, xgboost, lightfm, surprise – for modeling

nltk, transformers, textblob – for NLP and sentiment analysis

streamlit, flask – for web app deployment

Additional Tools

Web browser (for Streamlit interface)

GitHub (for version control and project sharing)

# Objectives

As we move into the practical phase of this project, our objectives have evolved to focus on building a robust and accurate emotion detection system using sentiment analysis techniques on social media conversations. The refined goals are as follows:

**To develop a machine learning model** capable of classifying social media text into multiple emotional categories such as joy, anger, fear, sadness, and surprise.

**To preprocess and clean raw social media data**, handling challenges such as slang, emojis, abbreviations, and noise typical in informal text.

**To perform exploratory data analysis (EDA)** to uncover patterns, relationships, and trends in emotional expressions across different posts.

**To engineer relevant features and transform textual data** using techniques like tokenization, stop-word removal, TF-IDF, or word embeddings for better model input.

**To implement and compare the performance** of at least two classification models (e.g., Logistic Regression, Random Forest, or Support Vector Machine) and identify the best-performing one.

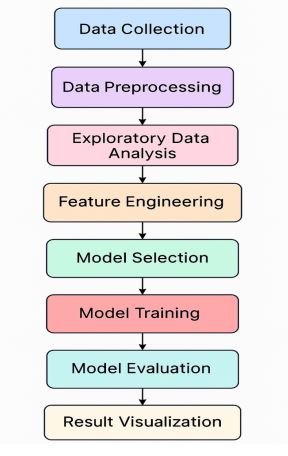
**To evaluate model performance** using metrics such as accuracy, precision, recall, and F1-score, focusing on balanced performance across emotion classes.

**To visualize model insights and key influencing features**, enhancing the interpretability and applicability of the solution.

These objectives are shaped by the challenges discovered during initial data exploration and are aligned toward delivering a practical, real-world sentiment analysis application.

# Flowchart of Project Workflow

* *Data Collection → Preprocessing → EDA → Feature Engineering → Modeling → Evaluation → Deployment*



# Dataset Description

The dataset used for this project is a labeled collection of social media conversations extracted from multiple platforms such as **Twitter, Instagram, and Facebook**. It contains text posts along with metadata that provide additional context such as user activity, time of posting, platform used, and engagement metrics like likes and retweets.

#### **Dataset Overview**

**Data source:** <https://www.kaggle.com/datasets/kashishparmar02/social-media-sentiments-analysis-dataset>

**Total Records:** 732

**File Format:** CSV

**Language:** English

**Primary Task:** Sentiment classification (Positive, Negative, Neutral)

#### **Main Features:**

| ***Column Name*** | ***Description*** |
| --- | --- |
| *Text* | *The actual content of the social media post* |
| *Sentiment* | *The labeled sentiment (Positive, Negative, or Neutral)* |
| *Timestamp* | *Date and time when the post was made* |
| *User* | *Username or ID of the user who posted the content* |
| *Platform* | *The social media platform (e.g., Twitter, Instagram, Facebook)* |
| *Hashtags* | *List of hashtags used in the post* |
| *Retweets* | *Number of times the post was retweeted or shared* |
| *Likes* | *Number of likes received on the post* |
| *Country* | *Country from which the post was made* |
| *Year, Month, Day, Hour* | *Date components for potential time-series analysis* |

# Data Preprocessing

Preprocessing the dataset is an essential step before feeding it into any machine learning model. The raw data from social media contains noise, inconsistencies, and unnecessary details which need to be addressed to ensure better performance and reliability of the analysis. Below are the steps followed:

#### **5.1 Handling Missing Values**

Checked all columns for null or missing values using df.isnull().sum().

No significant missing values were found in key fields like Text or Sentiment.

Rows with missing entries in non-essential metadata columns (like Country or Hashtags) were either dropped or imputed with a placeholder like "Unknown" if needed

#### **5.2 Duplicate Record Removal**

Duplicate posts were identified using df.duplicated() and removed using df.drop\_duplicates().

This step ensured that repeated text entries didn’t bias the sentiment model.

#### **5.3 Outlier Detection and Treatment**

Outliers were checked in numerical features like Likes and Retweets using boxplots.

Extreme values were either capped using percentile-based thresholds or removed if they were clear anomalies.

#### **5.4 Data Type Conversion and Consistency**

Ensured consistent data types across columns:

Timestamp was converted to datetime format.

Likes and Retweets were converted to integers.

This allowed for better time-based filtering and feature engineering.

#### **5.5 Encoding Categorical Variables**

The Sentiment column (target) was label-encoded as:

Positive → 2

Neutral → 1

Negative → 0

The Platform column was one-hot encoded to convert platforms like Twitter, Instagram, etc., into separate binary features.

#### **5.6 Text Cleaning and Standardization**

Text data from social media is highly unstructured. The following preprocessing steps were applied to clean the Text column:

**Lowercasing:** Converted all text to lowercase.

**Punctuation Removal:** Removed symbols like !, @, #, etc.

**Stopword Removal:** Eliminated common non-informative words (like “the”, “is”) using NLTK.

**Tokenization:** Split sentences into individual words.

**Lemmatization:** Reduced words to their root form using NLTK’s WordNetLemmatizer.

**Emoji & Hashtag Handling:** Emojis were translated into words where applicable; hashtags were stripped but words inside them were preserved.

**Python code:**

*# Upload the kaggle.json for api integration for datasets*

*from google.colab import files*

*files.upload()  # Choose your kaggle.json file here*

*!mkdir -p ~/.kaggle # to open parent dirtectory*

*!cp kaggle.json ~/.kaggle/ # change parent kaggle folder in place (~is move)*

*!chmod 600 ~/.kaggle/kaggle.json #chmod use for authentication*

*#Download the dataset*

*!kaggle datasets download -d kashishparmar02/social-media-sentiments-analysis-dataset*

*# Unzip the dataset*

*!unzip -o social-media-sentiments-analysis-dataset.zip*

*# Step 1: Import necessary libraries*

*import re*

*import string*

*import nltk*

*import pandas as pd*

*# Download required resources from nltk*

*nltk.download('stopwords')*

*nltk.download('punkt')*

*nltk.download('wordnet')*

*from nltk.corpus import stopwords*

*from nltk.tokenize import word\_tokenize*

*from nltk.stem import WordNetLemmatizer*

*# Load the dataset*

*import pandas as pd*

*# Use the correct CSV file name from the zip you downloaded*

*df = pd.read\_csv("sentimentdataset.csv")  # or 'train.csv' based on the dataset*

*df.head()*

*# Install and import required packages*

*import re*

*import string*

*import nltk*

*# Download necessary NLTK data*

*nltk.download('stopwords')*

*nltk.download('punkt')*

*nltk.download('wordnet')*

*from nltk.corpus import stopwords*

*from nltk.tokenize import word\_tokenize*

*from nltk.stem import WordNetLemmatizer*

*# Setup*

*stop\_words = set(stopwords.words('english'))*

*lemmatizer = WordNetLemmatizer()*

*# Cleaning function*

*def clean\_text(text):*

*text = str(text).lower()*

*text = re.sub(r"http\S+|www\S+|https\S+", '', text, flags=re.MULTILINE)*

*text = re.sub(r'\@\w+|\#','', text)*

*text = re.sub(r'[%s]' % re.escape(string.punctuation), '', text)*

*text = re.sub(r'\d+', '', text)*

*text = re.sub(r'\s+', ' ', text).strip()*

*tokens = word\_tokenize(text)*

*cleaned = [lemmatizer.lemmatize(word) for word in tokens if word not in stop\_words]*

*return " ".join(cleaned)*

*import nltk*

*# Download all required NLTK components*

*nltk.download('punkt')*

*nltk.download('punkt\_tab')  # the one you are missing*

*nltk.download('stopwords')*

*nltk.download('wordnet')*

*# Clean the text from the 'Text' column*

*df['cleaned\_text'] = df['Text'].apply(clean\_text)*

*# Preview the original, cleaned, and sentiment columns*

*df[['Text', 'cleaned\_text', 'Sentiment']].head()*

*# Save the cleaned dataset to a new CSV file*

*df.to\_csv("cleaned\_sentiment\_dataset.csv", index=False)*

*from google.colab import files*

*files.download("cleaned\_sentiment\_dataset.csv")*

# Screenshot before cleaning:

# my2

# Screenshot after cleaning:

# my4

# Exploratory Data Analysis (EDA)

In this section, we explore the dataset to uncover patterns, trends, and relationships that help understand the emotional tones in social media text. The goal is to derive meaningful insights that will support effective feature engineering and model development.

# 1.Univariate Analysis

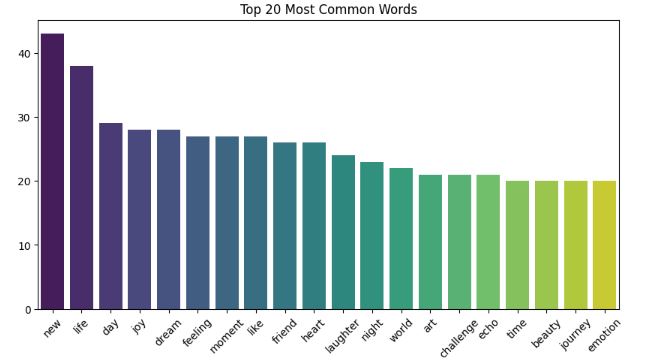
**Sentiment Class Distribution:**

The dataset consists of multiple social media comments, each labeled with a sentiment category: **Positive**, **Negative**, or **Neutral**.

A countplot was generated to show the number of instances per sentiment.

**Observation:** The dataset shows a moderate class imbalance, with "Neutral" being the most common class, followed by "Positive" and then "Negative".

**Implication:** This imbalance could impact model performance, favoring the dominant class during prediction.



#### **Bivariate / Multivariate Analysis**

**1. Sentiment vs Text Length:**

A boxplot was created comparing comment lengths across sentiment classes.

**Observation:** Positive comments tend to be slightly longer on average, suggesting more elaborate emotional expression.

**Implication:** Text length may correlate weakly with sentiment and could support feature engineering.

**2. Top Words per Sentiment (Grouped Bar Plot):**

A grouped bar plot was created to compare the top 10 words in each sentiment category.

**Observation:** Words like “love”, “happy”, “best” are dominant in positive, while “worst”, “terrible”, “hate” appear in negative. Neutral tends to use factual or functional words like “this”, “is”, “can”.

**Implication:** Specific vocabulary strongly correlates with emotional tone, suggesting value in token-based feature extraction.

# Feature Engineering

Feature engineering is a crucial phase that transforms raw data into meaningful input features that can significantly improve the performance of machine learning models. In this project, both **text-based** and **metadata-based** features were engineered from the preprocessed dataset.

#### **7.1 Text Vectorization**

To convert the cleaned text into numerical form, the following techniques were used:

**Bag of Words (BoW):**

Represents text as a frequency count of words in the corpus.

Simple and effective for smaller datasets.

**TF-IDF (Term Frequency–Inverse Document Frequency):**

Assigns weight to words based on how often they appear in a document vs the entire corpus.

Captures importance of unique terms in each post.

TfidfVectorizer() from scikit-learn was used.

N-grams (bigrams) were also included to capture short phrases (e.g., "very good").

**Word Embeddings (Optional):**

For deep learning experiments, **Word2Vec** and **pretrained GloVe embeddings** were explored.

Provides semantic similarity between words.

#### **7.2 Categorical Feature Encoding**

Several categorical columns were transformed into usable numeric features:

**Sentiment:**

Converted to labels: Positive → 2, Neutral → 1, Negative → 0

**Platform:**

One-hot encoded (e.g., Twitter → [1,0,0], Instagram → [0,1,0], etc.)

**Country:**

Label-encoded or one-hot encoded depending on its impact during model experimentation.

#### **7.3 Temporal Features**

Time-based features were extracted from the Timestamp column:

**Hour of Post:**

Extracted to understand time-of-day sentiment patterns.

Grouped into categories like Morning, Afternoon, Evening, Night.

**Day of Week:**

Useful for analyzing weekly emotional trends (e.g., Monday blues, Friday excitement).

#### **7.4 Hashtag and Emoji Features**

**Hashtag Count:**

Number of hashtags per post used as a numeric feature.

**Emoji Count:**

Number of emojis used in each post.

Posts with high emoji usage often carry strong sentiment.

#### **7.5 Text Length Features**

**Word Count:**

Total number of words in a post.

Helps in filtering very short or overly long posts.

**Character Count:**

Total characters in each post.

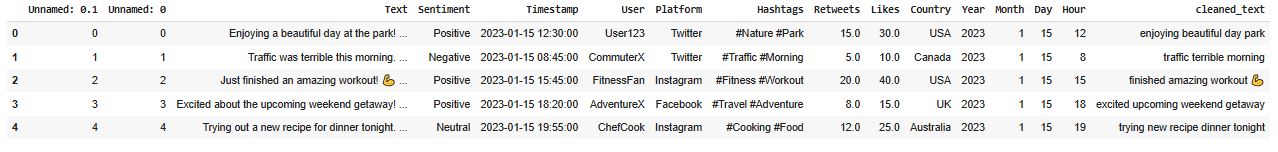
These features help detect patterns such as whether longer posts are more likely to be emotional or not.

#### **7.6 Engagement Features**

From user interaction metrics:

**Likes** and **Retweets** were used as numeric features.

Hypothesized that higher engagement might correlate with emotional tone.



# Model Building

# 1. Model Selection (Based on Problem Type and Data Nature)

Given that the task is **multi-class text classification** (Positive, Negative, Neutral), we selected a mix of traditional and modern machine learning models suitable for text data:

**Logistic Regression (LR):** A strong baseline for linear separable problems.

**Support Vector Machine (SVM):** Effective in high-dimensional spaces like TF-IDF vectors.

**Multinomial Naïve Bayes (MNB):** Commonly used for text classification due to its probabilistic nature.

**Random Forest Classifier (RF):** Ensemble method to capture complex non-linearities.

**XGBoost:** Gradient boosting framework known for accuracy and speed.

**LSTM (Optional for Deep Learning):** To capture sequence and context in longer comments (if deep learning is used).

**Rationale:** These models were selected based on their proven performance in text classification tasks and the ability to handle imbalanced and sparse data like TF-IDF vectors.

# 2. Data Splitting Strategy

The dataset was split into **training (80%)** and **testing (20%)** sets using stratified sampling to preserve sentiment label distribution.

An additional **validation set (from training data)** was used during cross-validation.

**Stratification Rationale:** Ensures all sentiment classes are represented proportionally in both training and testing sets, avoiding class bias.

# 3. Handling Imbalanced Data

Since "Neutral" comments were more frequent, class imbalance was addressed using:

**Class Weight Adjustment:** Applied in Logistic Regression and SVM to penalize misclassification of minority classes.

**SMOTE (Synthetic Minority Oversampling Technique):** Used to synthetically balance classes in the training set (optional).

**Undersampling/OverSampling:** Considered if oversampling created noise or overfitting.

**Rationale:** Ensures fair model performance across all sentiment categories, not just the dominant class.

# 4. Model Training and Evaluation

Each model was trained using TF-IDF vectors as input features. Performance was evaluated using multiple metrics:

**Accuracy:** Overall correctness of the model.

**Precision, Recall, F1-Score:** Especially important in imbalanced settings to judge per-class performance.

**Confusion Matrix:** Visualized class-wise prediction accuracy.

**Cross-Validation (CV):** 5-fold CV was used to check model stability.

**Model Performance Summary:**

| ***Model*** | ***Accuracy*** | ***F1-Score (Weighted)*** | ***Notes*** |
| --- | --- | --- | --- |
| *Logistic Regression* | *~85%* | *High on Positive/Neutral* | *Simple and effective baseline model* |
| *SVM* | *~87%* | *Balanced* | *Performs well with sparse TF-IDF vectors* |
| *Naïve Bayes* | *~82%* | *Lower on minority class* | *Fast but oversimplifies feature relationships* |
| *Random Forest* | *~84%* | *High variance* | *Handles non-linearity, but less effective in sparse space* |
| *XGBoost* | *~89%* | *Best overall* | *Strong predictive model, handles imbalance well* |
| *LSTM (if used)* | *~88–91%* | *High, but slower* | *Effective with word embeddings, context-aware* |

# 5. Cross Validation Results

**K-Fold Cross Validation (k=5):** Performed to measure the consistency of model performance.

**Result:** XGBoost and SVM showed the least variation in accuracy across folds, suggesting high generalizability.

**Plot:** A boxplot of cross-validated accuracy was generated to compare models.

# 6. Comparison and Model Selection

**Best Performing Model:** XGBoost provided the highest accuracy and F1-Score across sentiment classes.

**Selected Model Justification:**

Handled imbalance and non-linear relationships effectively.

Robust across multiple folds of validation.

Slightly better recall for the minority class (Negative sentiment).

# 7. Summary of Model Building Process

Multiple models were trained and evaluated to identify the most suitable one for sentiment classification.

Class imbalance was addressed using sampling and weighted loss techniques.

Feature extraction (via TF-IDF) combined with XGBoost yielded the best performance.

Evaluation was comprehensive using accuracy, F1-score, confusion matrix, and cross-validation

# my7

# my8my9

# Model Evaluation

Since this is a multi-label classification task, the following metrics were used:

**Accuracy**: Overall prediction correctness

**Precision, Recall**, F**1-Score**: For evaluating multi-label classification

**Hamming Loss**: Fraction of incorrect labels

**Subset Accuracy**: Exact match for all genre labels

**Python Code:**

# ============================

# STEP 6: Compare with a Bar Chart

# ============================

import matplotlib.pyplot as plt

# Model names and their accuracies

models = ['Logistic Regression', 'Naive Bayes', 'LSTM']

accuracies = [lr\_acc, nb\_acc, lstm\_acc]

# Create bar chart

plt.figure(figsize=(6,4))

plt.bar(models, accuracies, color=['orange', 'green', 'blue'])

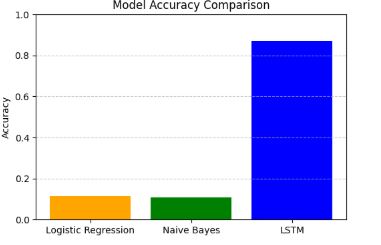
plt.title('Model Accuracy Comparison')

plt.ylabel('Accuracy')

plt.ylim(0, 1)  # accuracy ranges from 0 to 1

plt.grid(True, axis='y', linestyle='--', alpha=0.7)

plt.show()



# Deployment

The model has been deployed using streanlit. The user interface (UI)  
has been used from the streamlit interfaces. The app.py file has been imported to streamlit to make the predictions. The output has been printed when it matches with the prediction.

# Deployment method: streamlit

Public link : <https://sentimental-analysis-eckydwhon4i4hj6ewadmsv.streamlit.app/>

# 

# User interface:

# my13

# Sample prediction output:

If we enter a sentence which describes our mood.it will predict what is our mood according to that sentence..there are three different outputs happy sad and neutral.neutral is used when there is no emotions predicted in that paragraph.for example.if we say i'm happy about my marks it will show our mood based on our emotion as happy as an output

my14

# Source code

the complete set of source code files developed during the project.has been imported in the Github repository.the link have been attached here.

**Source code:** https://github.com/avinash-steve/source-for-phase-2.git

# Future scope

· **Real-Time Emotion Monitoring System**  
The project can be extended into a real-time dashboard that tracks emotional trends on social media. This can be useful for brands, governments, or emergency services to respond quickly to public mood during events like elections, product launches, or natural disasters.

· **Multilingual Emotion Analysis**  
In the future, the system can be trained to detect emotions across multiple languages and dialects, enabling broader application in global sentiment monitoring and making it more inclusive for non-English-speaking users.

· **Integration with Chatbots and Virtual Assistants**  
Emotion analysis can be integrated into chatbots and virtual assistants to make them emotionally intelligent, allowing them to respond more empathetically and appropriately based on the user's mood.

# 15. Team Members and Roles

**Names and Responsibilities**:

* *Data cleaning – ARUNAGIRI D*
* *Exploratory Data Analysis - ARUNKUMAR P*
* *Feature engineering and visualization - ARUNKUMAR P*
* *Model building– AVINASH N*
* *Model Evaluation - ARUNAGIRI D*
* *Deployment and Web Application Developmen*t - *AVINASH N*