Phase-3 Submission Template

Student Name: Eswari.S

Register Number: 422023104011

Institution:Sri Rangapoopathi College of engineering

Department: BE-computer science engineering

Date of Submission: 15-05-2025

Github Repository Link: https://github.com/Eswari2006/NM-project-.git

---

1. Problem Statement

With the rise of social media,people frequently share opinions on environmental conservation. However, understanding the emotions behind these posts is challenging due to the volume and complexity of language. Traditional sentiment analysis often misses nuanced emotional cues, highlighting the need for advanced methods to accurately interpret public sentiment and support for conservation efforts.The exponential growth of user-generated content on social media platforms has made them rich sources for understanding public opinion, behavior, and emotional states. However, decoding emotion through sentiment analysis of social media conversations remains a complex and unresolved challenge. Unlike structured textual data, social media content is highly informal, ungrammatical, and rife with slang, abbreviations, emojis, code-switching, and sarcasm—features that complicate natural language processing tasks. Furthermore, social media posts

2. Abstract

In the digital age, social media platforms have become vital spaces for individuals to express thoughts, emotions, and opinions. The vast and unstructured nature of user-generated content presents both an opportunity and a challenge for understanding public sentiment. This study explores the application of sentiment analysis techniques to decode emotions embedded in social media conversations. By leveraging natural language processing (NLP), machine learning algorithms, and emotion classification models, we analyze textual data from platforms such as Twitter, Facebook, and Reddit. The research focuses on identifying key emotional states—such as joy, anger, sadness, and fear—and understanding their contextual emergence in online discourse. Additionally, we evaluate the accuracy of various sentiment analysis tools and frameworks in recognizing nuanced emotional expressions. The findings provide insights into how real-time emotional trends can be tracked and interpreted, offering implications for fields such as marketing, mental health monitoring, political forecasting, and crisis management.

3. System Requirements

➢ **Hardware Requirements**

- Processor: Intel Core i5 or higher (or AMD equivalent)
- RAM: Minimum 8 GB (16 GB recommended for large datasets)
- Storage: At least 100 GB of free disk space
- GPU: NVIDIA GPU with CUDA support (for deep learning models) – optional but recommended
- Internet Connection: Required for accessing social media APIs and online libraries

➢ **Software Requirements**

- Operating System
- Windows 10 or later / Linux **(Ubuntu 18.04 or**

➢ **Programming Language & Tools**

- Python 3.8+ – Primary programming language
- Jupyter Notebook / VS Code / PyCharm – Development Environment

➢ **Libraries and Frameworks** ● Natural Language Processing:

- NLTK (Natural Language Toolkit)
- SpaCy
- TextBlob
- Transformers (HuggingFace)
- Machine Learning:
- Scikit-learn
- TensorFlow / PyTorch (for deep learning models)
- Data Handling:
- Pandas
- NumPy

- Visualization:
- Matplotlib
- Seaborn
- Plotly (for interactive graphs)

➢ **APIs and External Tools**

- Twitter API / Facebook Graph API / Reddit API – For data extraction
- Tweepy / PRAW (Python Reddit API Wrapper) – Python libraries for accessing social media platforms
- Google Cloud / AWS / Azure – Optional for cloud-based processing and scalability

➢ **Database (optional)**

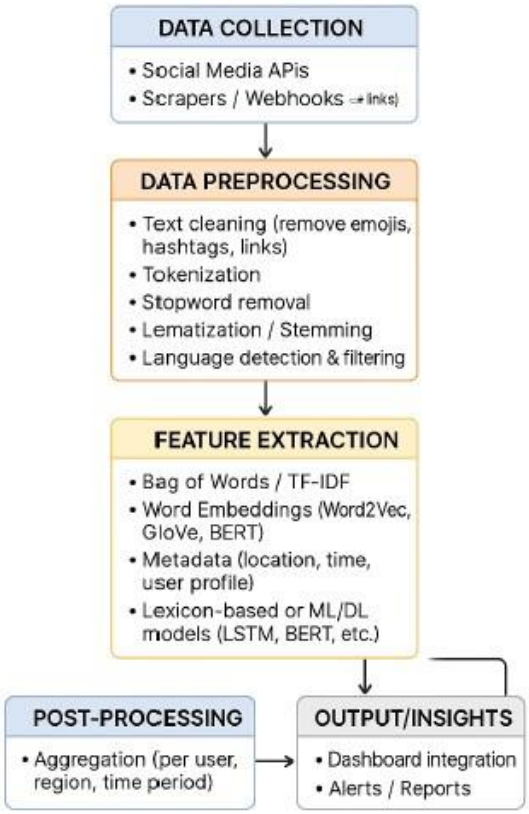
- MongoDB / MySQL / PostgreSQL – For storing large volumes of collected and processed data

4. Objectives

The primary objective of this project is to analyze and interpret emotional expressions embedded within social media conversations using sentiment analysis techniques. The study aims to collect and preprocess textual data from platforms such as Twitter, Facebook, and Reddit, transforming raw user content into a format suitable for analysis. By applying natural language processing (NLP) methods, the

project seeks to detect and classify emotions—such as joy, anger, sadness, fear, and surprise—within posts and comments. Another key objective is to evaluate and compare the performance of different sentiment analysis tools and models, including lexicon-based methods, machine learning algorithms, and deep learning frameworks. The project also focuses on visualizing emotional trends through graphs, word clouds, and timelines to better understand how public sentiment evolves over time or in response to specific events. Additionally, it aims to analyze the polarity of sentiments (positive, negative, neutral) and their contextual relevance within conversations.

5. Flowchart of Project Workflow



6. Dataset Description

➢ Sources:

- Textual data collected from two major social media platforms: Twitter and Reddit.

➢ Data Collection Tools:

- Twitter data retrieved using the Twitter API via the Tweepy library.
- Reddit data collected using PRAW (Python Reddit API Wrapper).

➢ Data Focus:

- Tweets and posts related to specific hashtags, keywords, or notable events.
- Subreddits chosen for their focus on emotional or opinion-based discussions.

➢ Data Components:

- Includes user posts, comments, timestamps, and optionally metadata .

➢ Privacy and Ethics:

- All personally identifiable information (PII) is excluded or anonymized.
- Data handling complies with platform policies and ethical research standards.

7. Data Preprocessing

➢ Data Cleaning:

- Remove URLs, user mentions (@username), hashtags, HTML tags, and special characters.

- Eliminate emojis and emoticons (or convert them to text if relevant to emotions).

- Strip out numbers and punctuation not essential for meaning.

➢ Lowercasing:

- Convert all text to lowercase to ensure uniformity.

➢ Tokenization:

- Split sentences into individual words or tokens for easier analysis.

➢ Stop Words Removal:

- Remove common words (e.g., "the", "is", "in") that do not contribute significantly to sentiment.

➢ Stemming and Lemmatization:

- Reduce words to their root or base form (e.g., "running" → "run") to group similar terms.

➢ Contraction Handling:

- Expand contractions (e.g., "don't" → "do not") to improve token clarity.

➢ Language Detection and Filtering:

Detect and retain only text in the desired language (e.g., English).

➢ Text Normalization:

- Standardize informal language, slang, and abbreviations (e.g., "u" → "you").

➢ Vectorization (Text to Numbers):

- Convert processed text into numerical form using:

8. Exploratory Data Analysis (EDA)

➢ Dataset Overview:

- Inspect the structure of the dataset (number of records, features/columns).
- Check for missing values or anomalies in the data.

➢ Class Distribution:

- Analyze the distribution of sentiment labels (positive, negative, neutral) and emotional categories (e.g., joy, anger, sadness, fear).
- Identify class imbalance, if any, to guide model selection or balancing techniques.

➢ Most Frequent Words:

- Generate frequency distributions of words across the dataset.
- Identify commonly used words within each sentiment/emotion category.

➢ Word Cloud Visualization:

- Create word clouds to visualize dominant terms associated with different emotions or sentiments.

➢ Hashtag and Mention Analysis:

- Examine the most frequent hashtags and mentions to understand popular topics and influencers.

➢ Time-Based Analysis (if timestamp is available):

- Track how sentiment or emotion trends change over time (daily, weekly, event-based).
- Detect peaks in emotional expression that may correlate with external events (e.g., political issues, product launches).
- Text Length Distribution:
- Analyze the length of posts (in words or characters) to understand the nature of content being analyzed.
- Sentiment Polarity Scores:
- Use tools like VADER or TextBlob to assign polarity scores to posts and visualize the sentiment strength.
- Co-occurrence Analysis:
- Identify which words or emotions tend to appear together in posts.
- Correlation and Heatmaps:

Visualize relationships between variables, such as sentiment scores and engagement metrics (likes, retweets).

9. Feature Engineering

➢ Text Vectorization:

- Bag of Words (BoW): Represents text as a vector of word frequencies.
- TF-IDF (Term Frequency–Inverse Document Frequency): Weighs words based on their importance across documents.
- Word Embeddings: Use pre-trained models like Word2Vec, GloVe, or BERT to capture semantic meaning and context.

➢ Sentiment Scores:

- Generate polarity and subjectivity scores using tools like TextBlob, VADER, or SentiWordNet.
- These scores can be used as numerical features for sentiment intensity.

➢ Emotion Lexicon Features:

- Use emotion dictionaries (e.g., NRC Emotion Lexicon) to create features indicating the presence of specific emotional terms (e.g., joy, fear, anger).

➢ Part-of-Speech (POS) Tags:

- Count of nouns, verbs, adjectives, and adverbs, which can reflect writing style and emotional tone.

➢ Syntactic Features:

- Number of exclamation marks, question marks, all caps words, or repeated characters (e.g., “soooo happy”)—often linked to emotional expression.

➢ Text Length Features:

- Total number of characters, words, and average word length—useful for understanding verbosity and tone.

➢ Hashtag and Mention Counts:

- Number of hashtags and @mentions in each post, which may relate to topic or user engagement.

➢ Negation Handling:

- Identify presence of negation words (e.g., “not”, “never”) to adjust sentiment interpretation.

➢ Emoji/Emoticon Indicators (if preserved):

- Count or presence of emojis can be encoded as categorical or binary features to reflect emotional states.

➢ Topic Modeling Features (Optional):

Use LDA (Latent Dirichlet Allocation) or BERTopic to extract dominant topics and include topic identifiers as features.

10. Model Building

➢ Data Splitting:

- Split the dataset into training and testing sets (e.g., 80% training, 20% testing) to evaluate model performance.
- Optionally, use cross-validation (e.g., K-fold cross-validation) to ensure robust model evaluation.

➢ Traditional Machine Learning Models:

- Logistic Regression: A simple but effective model for binary sentiment classification.
- Support Vector Machine (SVM): Often effective for text classification tasks.
- Random Forests / Decision Trees: For handling non-linear relationships and interactions between features.
- Naive Bayes: Especially useful for text classification with probabilistic models.

➢ Feature Engineering Integration:

- Use engineered features like TF-IDF, word embeddings, sentiment scores, and emotion lexicons as input for the models.

Combine numeric features (e.g., text length, POS tags) and categorical features (e.g., presence of hashtags, emojis) for more comprehensive learning.

➢ Hyperparameter Tuning:

- Perform hyperparameter optimization using Grid Search or Randomized Search to find the best model settings (e.g., regularization, number of trees, learning rate).

- For neural networks, adjust parameters like learning rate, batch size, number of epochs, and hidden layers

11. Model Evaluation

➢ Evaluation Metrics:

- Accuracy: Measures the percentage of correctly predicted labels across all classes.

- Precision: Indicates how many predicted positive/emotion labels are actually correct.

- Recall (Sensitivity): Measures how well the model identifies all relevant instances of a class.

- F1-Score: Harmonic mean of precision and recall; useful when the dataset is imbalanced.

- Confusion Matrix: Visualizes the number of true positives, false positives, false negatives, and true negatives for each emotion/sentiment class.

  ROC-AUC Score (for binary classification): Evaluates the trade-off between true positive rate and false positive rate.

➢ Cross-Validation:

- Use K-Fold Cross-Validation (e.g., 5-fold or 10-fold) to ensure model robustness and prevent overfitting.
- Average the results across folds to get a reliable performance estimate.

➢ Class Imbalance Analysis:

- Examine performance for each emotion/sentiment category separately to detect if the model favors majority classes.
- Apply macro or weighted averaging for F1-scores to ensure balanced evaluation.

➢ Error Analysis:

- Manually review misclassified samples to understand where and why the model is making mistakes.
- Identify confusion between similar emotions (e.g., anger vs. frustration, joy vs. excitement).

➢ Comparison of Models:

- Compare traditional machine learning models (e.g., SVM, Logistic Regression) with deep learning models (e.g., LSTM, BERT) using the same evaluation metrics.
- Select the best-performing model based on overall accuracy, F1-score, and generalizability.

➢ Model Interpretability (optional):

- Use tools like LIME or SHAP to understand which features contribute most to the model's predictions.

➢ Real-World Testing (if applicable):

- Test the model on unseen, real-time social media data to evaluate practical performance.

12. Deployment

➢ Objective of Deployment:

- To make the emotion prediction model publicly accessible through a simple, interactive web application.

➢ Chosen Deployment Platform:

- Gradio + Hugging Face Spaces – A free, cloud-based platform ideal for deploying and sharing machine learning models.

➢ Deployment Method:

- Develop a Python script that:
- Loads the pre-trained sentiment/emotion analysis model.
- Defines a `predict_emotion(text)` function to process user input.
- Uses Gradio to build an intuitive user interface.
- Push all files (model, script, and requirements.txt) to a GitHub repository.

- Create a Hugging Face Space and connect it to the GitHub repo for automatic deployment.

```
!pip install gradio

def predict_emotion(text):
    # Replace with your model prediction logic
    return "Predicted emotion: Joy"

import gradio as gr

def predict_emotion(text):
    # Replace with your model prediction logic
    return "Predicted emotion: Joy"

iface = gr.Interface(fn=predict_emotion, inputs="text", outputs="text")
iface.launch()
# gunicorn app:app  This line is a shell command and should not be part of the python code.  To run
!gunicorn app:app
```

13. Source code import gradio as gr import nltk import

joblib import string from nltk.corpus import stopwords

from sklearn.feature_extraction.text import

TfidfVectorizer

# Download stopwords if not already
nltk.download('stopwords')

```
# Load stop words stop_words =

set(stopwords.words('english'))



# Sample emotions (for a real model, train using a labeled dataset)

emotion_labels = ['joy', 'sadness', 'anger', 'fear', 'surprise', 'love']



# Load trained model and vectorizer (Assuming they are saved as .pkl

files) model = joblib.load("emotion_model.pkl") vectorizer =

joblib.load("vectorizer.pkl")



# Text preprocessing function def

preprocess_text(text):

    text = text.lower()     text = ''.join([char for char in text if char not

in string.punctuation])     tokens = text.split()     tokens = [word for

word in tokens if word not in stop_words]     return " ".join(tokens)
```

```
# Prediction function def

predict_emotion(text):

    processed = preprocess_text(text)      vectorized =

vectorizer.transform([processed])      prediction =

model.predict(vectorized)[0]      return f"Predicted

Emotion: {prediction.capitalize()}"


# Gradio UI interface = gr.Interface(      fn=predict_emotion,

inputs=gr.Textbox(lines=3, placeholder="Enter a tweet or comment..."),

outputs="text",      title="Emotion Detection from Social Media Text",

    description="Enter social media text to detect the expressed emotion (joy, anger,

sadness, etc.)."

)


# Launch the app if

__name__ == "__main__":

    interface.launch()
```

14. Future scope

➢ Textual Features:

- Raw user-generated content (tweets, Reddit posts, comments).
- Cleaned and preprocessed using tokenization, lowercasing, and stopword removal.

➢ Linguistic Features:

- Part-of-Speech (POS) Tags: Counts of nouns, verbs, adjectives, adverbs.
- Negation Terms: Words like "not", "never", "don't" that influence sentiment.

  Punctuation Usage: Frequency of exclamations, question marks, all caps (e.g., “I LOVE this!”).

➢ Lexical Features:

- TF-IDF (Term Frequency–Inverse Document Frequency): Reflects word importance in context.
- Bag of Words (BoW): Word frequency vectors for traditional ML models.

➢ Sentiment & Emotion Scores:

- Scores from tools like TextBlob, VADER, or NRC Emotion Lexicon.

- Emotion label probabilities or polarity scores (positive, neutral, negative).

➢ Embedding-Based Features:

- Word2Vec, GloVe, or BERT embeddings to capture contextual meaning of words.

- Vector representation of text for input into deep learning models.

➢ Metadata Features (if available):

- Timestamps: To detect temporal emotion trends.

- Likes, retweets, upvotes: Indicators of engagement and influence.

➢ Social Media-Specific Features:

- Hashtag Analysis: Frequency and relevance to emotional content.

- Emoji Interpretation: Conversion of emojis into emotion tags or text (e.g., 😊 → joy).
- Mention Counts (@user): Indicators of directed conversation or conflict.

➢ Text Structure Features:

- Sentence/Word Count: To assess verbosity.
- Average Word Length: Indicative of complexity or tone.

13. Team Members and Roles

Eswari.S- (overall project manager,Data Collection,Model Development)

Dhivyadharshini.A-(EDA, Feature Engineering,Interpretation)

Dhinakaran.T&Arunkumar.A-(Documentation And Reporting,preprocessing overseas visualization)