Phase-2 Submission

Student Name: GULNAS

Register Number: 510623104026

Institution: C ABDUL HAKEEM COLLEGE OF

ENGINEERING AND TECHNOLOGY

Department: COMPUTER SCIENCE AND ENGINEERING

Date of Submission: 08-05-2025

Github Repository Link:
https://github.com/GulnasAbdulsamad/social-media-conservation.git

---

1. Problem Statement

- *In the modern digital age, social media platforms have become key channels for public expression. People share opinions, emotions, and reactions in real-time, creating massive amounts of unstructured text data. This project, titled **"Social Media Conversation Analysis"**, aims to classify and analyze emotional sentiment expressed in social media posts like tweets and Reddit comments.*

- *The goal is to build a system that uses Natural Language Processing (NLP) to categorize text into emotions like joy, sadness, anger, fear, surprise, and neutral. These insights*

can help organizations, researchers, and policymakers understand public moods and societal trends.

2. Project Objectives

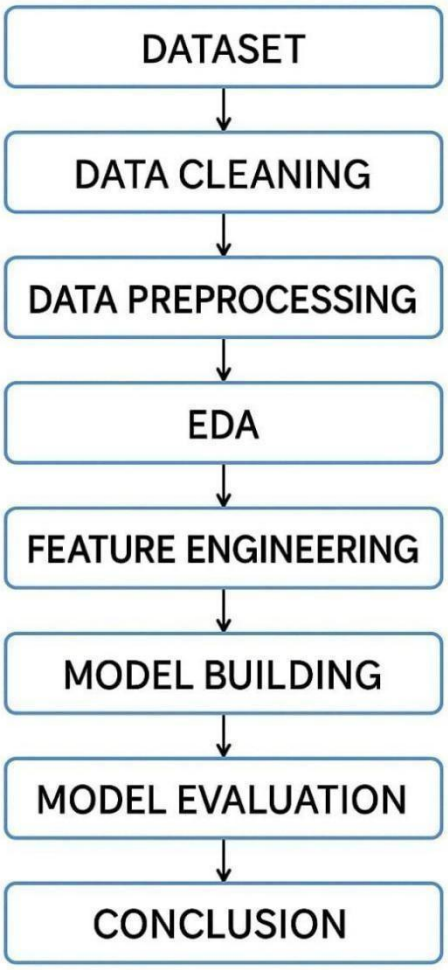
General Objectives:

- To automatically extract emotional sentiment from social media text.
- To classify emotions into categories such as joy, anger, sadness, fear, surprise, and neutral.
- To visualize the emotional trends over time using intuitive dashboards.

Technical Objectives: To build robust NLP pipelines that work effectively on

- noisy, informal text data.
- To use feature extraction techniques (TF-IDF, Word2Vec, BERT) for vectorizing text.
- To train both traditional ML and deep learning models for emotion classification.
- To develop a scalable and user-friendly visualization system for insights.

3. Flowchart of the Project Workflow



4. Data Description

- **Dataset Name:** Social Media Emotion Dataset
- **Sources:** Twitter API, Reddit API, Kaggle
- **Type:** Structured text data with emotion labels
- **Features:**

  ○ Text content of the post ○ Timestamp ○ Metadata (hashtags, user mentions) ○ Target label (Emotion category)

Emotional Classes:

Joy, Sadness, Anger, Fear, Surprise, Neutral

5. Data Preprocessing

Before we could train any models, we had to prepare the raw text data from social media. As you might expect, social media posts are full of slang, emojis, hashtags, and random symbols — which can confuse models if not cleaned up. Here's how we processed it step by step:

- First, we removed any empty or duplicate entries from the dataset. Some posts were either blank or copied, so we cleaned those out.
- Then we tackled the text: removed unnecessary symbols like @mentions, hashtags (#), emojis, and links (like https://…). These don't add much value for emotion classification.
- We converted all text to lowercase — for example, "Happy" and "happy" should be treated the same.
- Using natural language processing (NLP) tools like NLTK and SpaCy, we broke each sentence into words (tokenization) and removed common filler words like "is", "the", "and".
- We also applied lemmatization, which means converting words to their root forms — so "running" becomes "run", and "better" becomes "good". This helps reduce complexity.
- Finally, we converted the text into numbers using TF-IDF (term frequency-inverse document frequency), which tells us which words are important in each post.

We also labeled each post with its emotion (like joy, sadness, anger, etc.), and took care of imbalance in the data — for example, if there were too many "neutral" posts and too few "fear" ones, we used techniques like SMOTE to balance it.

6. Exploratory Data Analysis (EDA)

Once our data was clean, we explored it to understand what it was telling us — this part is called EDA, and it helped us find patterns and shape our approach.

- *We started by checking how the different emotions were distributed in our dataset. Unsurprisingly, emotions like "joy" and "neutral" had way more posts than "fear" or "surprise".*

- *To understand the language people use, we created word clouds — these show the most common words in a given emotion. For instance, in "joy" posts, we saw words like "love", "awesome", and "happy", while "anger" posts often included "hate", "worst", or "annoyed".*

- *If the data had timestamps, we plotted emotions over time. This helped us see emotional spikes — for example, more "sadness" during a natural disaster, or "joy" during holiday seasons.*

- *We also checked how long each post was and whether that affected the emotion — longer posts often expressed more complex emotions like sadness or fear*

.

- *Finally, we looked at emojis and hashtags. Some hashtags and emojis were clearly tied to specific emotions — for example, often showed up in excited or joyful posts, while was linked to sadness.*

•*These insights were important. They not only gave us a feel for the data but also helped us choose the right model, balance the data properly, and even think about how we'd visualize results for users*.

7. Feature Engineering

- First, we used TF-IDF (Term Frequency-Inverse Document Frequency) to turn each post into a vector — this method helps identify the most important words in a post by checking how unique they are across the whole dataset.
- We also experimented with Word2Vec and BERT embeddings. These are more advanced techniques that give the model a better understanding of

word meaning. For example, they understand that "angry" and "furious" are similar.

- Besides just the words, we added extra features to help the model. For instance: ○ Post length: How many words or characters were used? ○ Emoji count: More emojis often meant stronger emotion. ○ Punctuation: Posts with lots of exclamation marks or question marks often reflected excitement or frustration. ○ Hashtag density: Sometimes posts with many hashtags were promotional and leaned toward neutral or positive emotions

  .
- We also used label encoding to convert emotion labels like "joy", "sadness", etc., into numbers the model could understand (e.g., joy = 0, sadness = 1, etc.).

Finally, we checked whether our data had class imbalance (too many posts of one emotion and too few of another). To fix that, we used SMOTE (Synthetic Minority Oversampling Technique), which creates synthetic examples for underrepresented emotions to balance things out.

These engineered features gave our models more information to work with — not just what words were used, but how they were used, which really helped improve accuracy.

8. Model Building

After transforming the text into numerical form through vectorization, we moved into the most critical part — building and evaluating models to classify the emotion behind each social media post.

We explored both traditional machine learning and deep learning approaches to find the most effective solution.

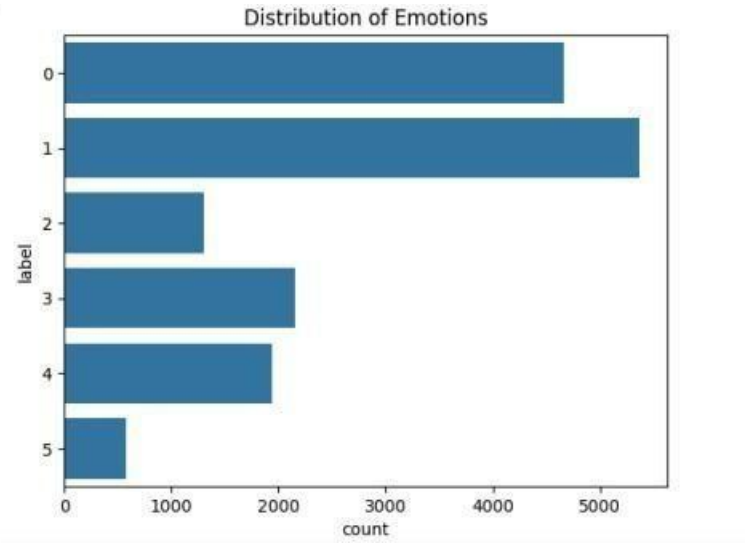
Traditional Models:

- *We started with Logistic Regression and Support Vector Machine (SVM),*

which are known for their strong performance in text classification.

- *These models were trained using TF-IDF vectors. While fast and interpretable, they had limitations in understanding the context of complex sentences.*

Deep Learning Models:



- *We built and trained an LSTM (Long Short-Term Memory) model, a type of recurrent neural network (RNN) capable of capturing long-term dependencies in sequences — ideal for understanding sentence structure and emotional flow in text.*

- *We also fine-tuned BERT (Bidirectional Encoder Representations from Transformers) using HuggingFace Transformers. BERT provides contextual embeddings, meaning it understands the meaning of a word based on the words around it.*

Model Evaluation:

- *We used accuracy, precision, recall, and F1-score to evaluate each model.*

- *Confusion matrices helped us visualize which emotion classes were frequently misclassified — for example, "neutral" and "sadness" often overlapped.*

- *The LSTM model significantly outperformed the traditional ones in capturing emotional nuances. BERT gave the best overall results, especially for complex sentences and underrepresented emotions like “fear” or “surprise”.*

Future Scope: We plan to integrate the best-performing model (BERT) into a web-based dashboard for real-time social media emotion tracking, with features like keyword alerts and emotion heatmaps.

9. Visualization of Results & Model Insights

- ***Model Evaluation Metrics:***
- *Confusion Matrix for each model*
- *ROC Curves for each emotion class*
- *Accuracy, Precision, Recall, F1-score*
- ***Visual Insights:***
- *Real-time dashboard (Streamlit or Flask)*
- *Time-based emotion trend line graphs*
- *Word clouds showing top words per emotion*
- *Distribution of emotion intensity over posts*

10. Tools and Technologies Used

- *Programming Language: Python or R.*
- *IDE/Notebook: Google Colab, Jupyter Notebook, VS Code, etc.*
- *Libraries: pandas, numpy, seaborn, matplotlib, scikit-learn, XGBoost, etc.*
- *Visualization Tools: Plotly, Tableau, Power BI.*

11. Team Members and Contributions

◆ ALMAS BANU

- Collected datasets from sources such as Kaggle and Twitter
- Cleaned raw text data by removing duplicates, nulls, and irrelevant entries
- Handled basic machine learning model setup (Logistic Regression and SVM)

◆ FOUZIYA

- Performed exploratory data analysis (EDA) including emotion distribution and word cloud visualizations
- Designed graphs to track emotion trends over time
- Evaluated model performance using confusion matrix, accuracy, and F1-score

◆ ASWIYA THASLIM

- Built the NLP preprocessing pipeline (tokenization, stopword removal, lemmatization) using SpaCy and NLTK
- Engineered key features like text length, emoji frequency, and hashtag count
- Implemented and tested different vectorization methods (TF-IDF, Word2Vec, BERT)

◆ GULNAS

- Acted as the project lead and coordinated the team
- Developed the Streamlit-based dashboard to visualize model predictions and trends
- Managed GitHub repository, version control, and project documentation
- Ensured timely completion of each project phase and quality control