Phase-2

Student Name: Venkatesan Gagan.P

Register Number: 410723106022

Institution: Dhanalakshmi college of engineering

Department: Electronics and Communication Engineering

Date of Submission: 07/04/2025

Github Repository Link: https://github.com/gagan7115/Nm_gagan_ds

---

Decoding emotions through sentiment analysis of social media conversations

1. Problem Statement

- The proliferation of social media has given rise to massive volumes of usergenerated content reflecting public opinions and emotions. However, extracting meaningful emotional insights from this data remains a challenge due to informal language, sarcasm, and contextual subtleties.

- This project aims to decode emotions embedded in social media conversations using sentiment analysis techniques.

Type of Problem:

- Multi-class classification (detecting emotions like joy, anger, sadness, etc.) or sentiment classification (positive, negative, neutral).

Relevance:

- Understanding emotional trends can benefit mental health monitoring, brand reputation management, political analysis, and crisis detection.

2. Project Objectives

- To preprocess and clean raw social media text data.
- To classify emotional tone or sentiment using machine learning and/or deep learning models.
- To evaluate model performance using precision, recall, F1-score, and accuracy.
- To visualize emotional distributions and understand key drivers behind different sentiments.

Updated Insight:

- After data exploration, we may explore multi-label classification if posts express more than one emotion.

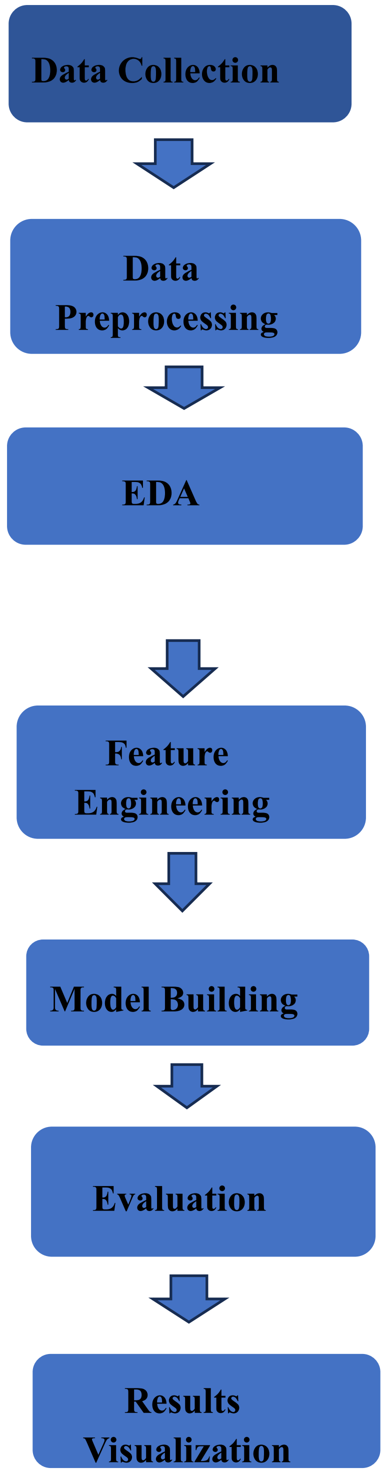
3. Flowchart of the Project Workflow :

The **Flowchart of the Project Workflow** provides a structured representation of how the sentiment analysis process progresses. It ensures that each step is logically connected and contributes to accurate emotion classification from social media conversations. Below is a detailed breakdown of the flowchart:

1. **Data Collection** – Gather social media posts, tweets, or relevant datasets.
2. **Data Cleaning** – Remove unnecessary elements like URLs, special characters, and duplicate values.
3. **Exploratory Data Analysis (EDA)** – Understand key patterns in text data using statistical methods.
4. **Feature Engineering** – Extract meaningful features such as word count, sentiment polarity, and emoji presence.
5. **Text Preprocessing** – Convert text to lowercase, tokenize sentences, remove stop words, and apply stemming/lemmatization.

6. **Vectorization** – Transform text into numerical representations using TF-IDF or word embeddings.
7. **Splitting Data** – Divide the dataset into training and testing sets using an 80-20 split.
8. **Model Selection** – Choose suitable machine learning or deep learning models (e.g., Logistic Regression, LSTM, BERT).
9. **Model Training** – Train models on labeled data to learn sentiment patterns.
10. **Hyperparameter Tuning** – Optimize model parameters for better performance.
11. **Performance Evaluation** – Use metrics like accuracy, F1-score, and confusion matrix to assess effectiveness.
12. **Error Analysis** – Identify misclassified emotions and refine the model accordingly.
13. **Visualization of Results** – Generate word clouds, class distribution plots, and correlation charts for insights.
14. **Prediction on New Data** – Apply trained models to unseen social media posts.
15. **Interpretability Analysis** – Understand which features contributed most to classification.
16. **Decision-Making** – Provide actionable insights based on sentiment trends.
17. **Report Generation** – Summarize findings into an understandable format.
18. **Deploying the Model** – Integrate the model into applications for real-time sentiment analysis.
19. **Monitoring & Updating** – Continuously refine the model as new data becomes available.
20. **Final Documentation** – Create a report detailing the project findings and improvements.

This structured approach ensures smooth execution and optimization of sentiment analysis results. Let me know if you need a more detailed explanation for any step!


Data Collection
Data Preprocessing
EDA
Feature Engineering
Model Building
Evaluation
Results Visualization

4. Data Description

- **Dataset Name**: e.g., Twitter Sentiment140, EmoReact, or Kaggle emotion datasets.
- **Source**: [Specify dataset source like Kaggle, UCI, etc.]
- **Data Type**: Unstructured text data (tweets, posts).
- **Records/Features**: [e.g., 100,000+ rows, columns include text, emotion label, timestamp].
- **Static or Dynamic**: Static snapshot.
- **Target Variable**: Emotion or sentiment label (e.g., joy, fear, anger, sadness).
- **Data Set Link**: https://www.kaggle.com/datasets/vidyapb/elon-musk-tweets-2015-to-2020?resource=download

5. Data Preprocessing

- Removed URLs, mentions, hashtags, emojis, and special characters.
- Handled missing and duplicate values.
- Converted text to lowercase.
- Tokenized and removed stop words.
- Performed stemming/lemmatization.
- Encoded labels (label encoding or one-hot). • Vectorized text (TF-IDF, CountVectorizer, or embeddings like BERT).

```
from sklearn.feature_extraction.text import TfidfVectorizer
tfidf = TfidfVectorizer(max_features=5000)  X =
tfidf.fit_transform(cleaned_text)
```

6. Exploratory Data Analysis (EDA)

Univariate:

- Word clouds, top frequent words by emotion, class distribution plots.

Bivariate:

- Word usage patterns by emotion class.

Multivariate: • Correlation between post length, time of posting, and sentiment.

Key Insights:

- Common positive emotions include "joy", often using words like "great", "love", "happy".
- Negative emotions showed spikes during specific events/dates.

7. Feature Engineering

Created new features:

- word count, sentiment polarity (TextBlob), emoji presence.
- Used n-gram features. • Experimented with dimensionality reduction (e.g., PCA on TF-IDF).

```
df['word_count'] = df['text'].apply(lambda x: len(x.split()))
df['sentiment_polarity'] = df['text'].apply(lambda x:
    TextBlob(x).sentiment.polarity)
def contains_emoji(s):
    return any(char in emoji.EMOJI_DATA for char in s) df['emoji_present'] =
    df['text'].apply(contains_emoji).astype(int)

    df['pca_1'] = tfidf_pca[:, 0]
df['pca_2'] = tfidf_pca[:, 1]
```

8. Model Building

Models Used:

- Logistic Regression for baseline.

- Random Forest for interpretability.
- LSTM/BERT for deep contextual understanding.

```
Ir_model = make_pi[eline(TfidVectorizer(),LogisticRegression())
Ir_model.fit(X_train, y_train)
Print("Logistic Regression Accuracy:",Ir_model.score(X_test, y_test))
```

Data Split: • 80-20 (train-test), with stratification.

Metrics Used: • F1-score (multi-class), accuracy, confusion matrix.

```
from sklearn.metrics import classification_report
print(classification_report(y_test, y_pred))
```

9. Visualization of Results & Model Insights •

Confusion Matrix to analyze misclassifications.

- ROC Curves for each class (multi-class).
- Feature importance (Random Forest) or attention scores (BERT).

Class distribution chart:

- Before vs After model.

10. Tools and Technologies Used

- **Language**: Python
- **Notebook**: Jupyter / Google Colab

- **Libraries**: pandas, numpy, nltk, sklearn, matplotlib, seaborn, transformers (HuggingFace), TensorFlow/Keras
- **Visualization**: seaborn, matplotlib, WordCloud

11. Team Members and Contributions

S.No	Names	Roles	Responsibility
1.	Karthik S	Leader	Data collections and Data cleaning
2.	P Venkatesan Gagan	Member	Visualization & Interpretation
3.	Joshua Judson J	Member	Exploratory Data Analaysis(EDA)
4.	SanthaKumar p	Member	Model Building & Model Evaluation