Phase-2

Student Name: Krishnakumar K

Register Number: 712523121011

Institution: PPG INSTITUTE OF TECHNOLOGY

Department: BIOMEDICAL ENGINEERING

Date of Submission:02-05-2025

Decoding emotions through sentiment analysis of social media conversations

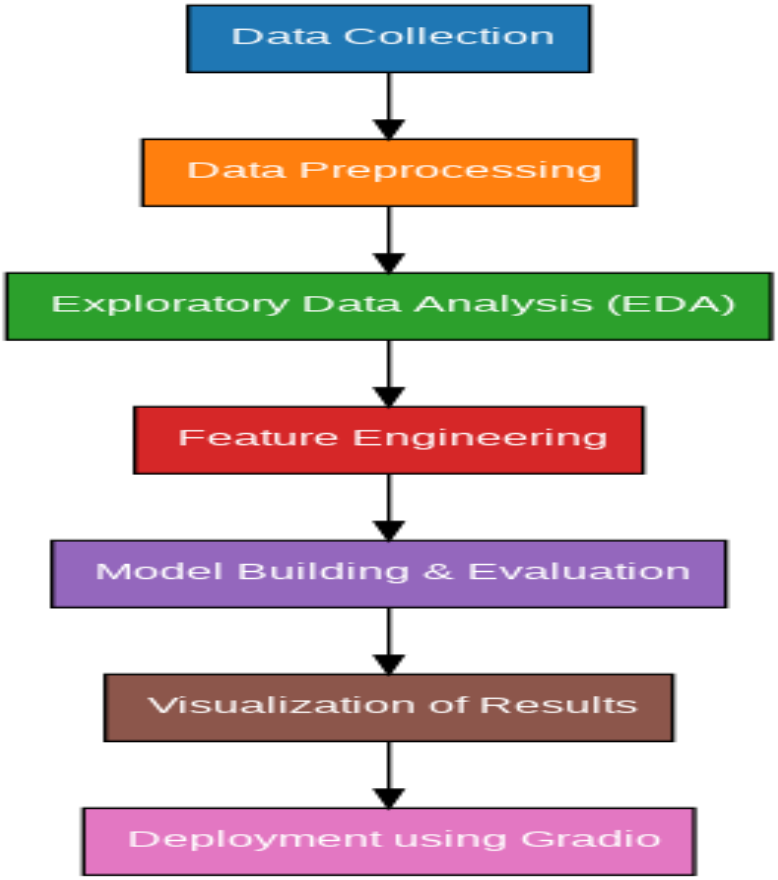
1.Problem Statement:

In the digital age, social media platforms have become powerful mirrors of human emotion, capturing raw, real-time sentiments expressed across the globe. This project explores how sentiment analysis—a natural language processing (NLP) technique—can decode these emotions by analyzing conversations on platforms like Twitter, Facebook, Instagram, and whatsapp.

2.Objectives of the Project:

- To identify emotional patterns and trends in social media conversations.
- To classify sentiments (positive, negative, neutral) and deeper emotions (joy, anger, fear, etc.) using machine learning and deep learning models.
- To assess the impact of external events (e.g., pandemics, elections, celebrity news) on collective emotional expression.
- To understand the role of language nuances, slang, emojis, and sarcasm in sentiment detection.

3. Flowchart of the Project Workflow



4. Data Description:

- Dataset Name: analysis of social media conversations.

Source: Manually created examples

- Type of Data: Structured (Text + Labels).
- Number of Records:
- Static/Dynamic Dataset
- Target Variable:

5. Data Preprocessing:

1. Data Cleaning:

- Lowercasing: Convert all text to lowercase to reduce dimensionality.
- Removal of Noise:
    - URLs, hashtags (#topic), mentions (@user), and HTML tags are removed or replaced with placeholders.
    - Emojis are either removed or translated into text (e.g., 😊 → "smiling face") using emoji libraries, since they often carry strong emotional meaning.
- Special Characters: Remove punctuation, special symbols, and extra whitespace.
- Spelling Correction: Correct common typos or use normalization to handle informal language (e.g., "luv" → "love").

2. Language Detection and Filtering:

- If multilingual data is used, detect and separate posts by language or apply multilingual models like XLM-RoBERTa.
- Optionally translate non-English posts to English for consistency.

3. Annotated Data Alignment

- Ensure that the preprocessed text is correctly aligned with the sentiment/emotion labels.
- Handle multilabel cases appropriately (e.g., a post labeled with both *anger* and *disgust*).

4. Train-Test Split

- Stratified splitting to maintain emotion distribution across training, validation, and test sets.
- Typical split: 70% training, 15% validation, 15% test.

6. Exploratory Data Analysis (EDA):

1. Dataset Overview

- **Total Records**: Number of posts/comments.

- **Unique Users**: Count of distinct user IDs (if available).
- **Platforms Represented**: Breakdown of data by source (e.g., Twitter: 60%, Reddit: 30%, Facebook: 10%).

2. Label Distribution

- **Emotion Categories**:
  - Count and percentage of each emotion (e.g., joy: 25%, sadness: 18%, anger: 15%, etc.).
  - Visualizations:
    - **Bar plots** to show frequency of emotions.
    - **Pie charts** for proportion comparison.
    - **Word clouds** for each emotion to visualize commonly used terms.

3. Emoji and Hashtag Analysis

- **Emoji Usage**:
  - Most common emojis and their associated emotions.
  - Visualize frequency using bar charts or emoji clouds.
- **Hashtag Trends**:
  - Frequent hashtags and their emotional context.
  - Useful for thematic clustering of posts.

7. Feature Engineering:

1. Textual Features

These are features directly extracted from the text content:

- **Bag of Words (BoW)**:
  - Represents text as a vector of word frequencies or presence/absence.
  - Simple but often effective for traditional models.
- **TF-IDF (Term Frequency-Inverse Document Frequency)**:

- Gives importance to words that are frequent in a document but rare across the corpus.
- Applied on unigrams, bigrams, or trigrams.

N-grams:

- Captures contextual word sequences.
- Example: "very happy", "not good".

- **Lexicon-based Sentiment Scores**:
  - Use sentiment/emotion lexicons like VADER, NRC Emotion Lexicon, or SentiWordNet to extract:
    - Polarity scores (positive/negative/neutral)
    - Emotion intensities (joy, anger, etc.)
- **Part-of-Speech (POS) Tags**:
  - Frequency of nouns, verbs, adjectives, etc.
  - Adjectives and adverbs often carry emotional weight.

Feature Selection Techniques

- **Univariate Feature Selection**: Chi-square, ANOVA for classification relevance.
- **Recursive Feature Elimination (RFE)**: For reducing feature set while maintaining performance.
- **PCA or LDA**: Dimensionality reduction techniques for visualization or input size reduction.

8. Model Building:

1. Problem Framing

- Emotion Classification:
  - Multi-Class: One dominant emotion per post.
  - Multi-Label: Multiple emotions can co-occur (e.g., *sadness* + *anger*).
- Sentiment Classification:
  - Three-Class: Positive, Negative, Neutral.

2. Model Types

A. Traditional Machine Learning Models (Baseline)

- Algorithms:
    - Logistic Regression
    - Support Vector Machine (SVM)
    - Random Forest
    - Naive Bayes
- Input:
    - TF-IDF vectors, N-grams, and hand-crafted features (punctuation, emoji count, etc.).
- Pros:
    - Fast to train and interpret.
    - Good as baselines.
- Cons:
    - Limited ability to capture deep semantic context.

B. Deep Learning Models

- RNN-based Models:
    - LSTM and GRU networks for capturing sequential context.
    - Often combined with GloVe or Word2Vec embeddings.
- CNN for Text:
    - Convolutional layers over word embeddings to extract local emotional cues.
- Attention Mechanisms:
    - Add attention layers to LSTM/GRU to focus on emotionally significant words.

C. Transformer-Based Models

- Pretrained Language Models:
    - BERT, RoBERTa, DistilBERT, XLM-R (for multilingual data).
    - Fine-tuned for classification using the [CLS] token representation.
- Implementation Strategy:
    - Tokenize using model-specific tokenizer.
    - Add classification head (dense + softmax or sigmoid layer).
    - Fine-tune on emotion and sentiment labels.
- Pros:
    - State-of-the-art performance on NLP tasks.
    - Captures complex emotional nuances from context.
- Cons:
    - Requires more compute.
    - Needs careful hyperparameter tuning to avoid overfitting.

3. Handling Imbalanced Data

- Class Weights: Adjust loss functions to penalize underrepresented classes.
- Oversampling/Undersampling: Balance the dataset during training.
- Focal Loss: Improve learning on hard-to-classify samples in multi-label tasks.

4. Model Training

- Training-Validation-Test Split: Typical split is 70-15-15 or 80-10-10.
- Loss Functions:
    - CrossEntropyLoss for multi-class.
    - Binary Cross-Entropy or Focal Loss for multi-label.
- Optimizers: AdamW (commonly used with transformers).
- Learning Rate Schedulers: Linear warmup and decay for transformers.

5. Evaluation Metrics

- Accuracy (for balanced multi-class problems).
- Precision, Recall, F1-Score (for imbalanced and multi-label settings).
- Macro vs. Micro Averaging: Use macro-F1 to ensure performance across all emotion categories.
- Confusion Matrix: Understand misclassifications between similar emotions (e.g., sadness vs. fear).
- ROC-AUC (for multi-label): Assess classification threshold quality.

6. Model Selection and Tuning

- Perform Grid Search or Bayesian Optimization for:
    - Learning rate
    - Dropout rate
    - Batch size
    - Number of hidden units (for custom networks)
- Use cross-validation on traditional models.
- Track metrics using TensorBoard, Weights & Biases, or MLflow.

7. Deployment Readiness (Optional)

- Save best-performing model as .pkl, .h5, or .pt (depending on framework).
- Build an API wrapper (Flask/FastAPI) for real-time sentiment/emotion detection.
- Consider latency, especially for transformer-based models.

9. Visualization of Results & Model Insights:

1. Performance Metrics Visualization

- **Confusion Matrix**:
    - Displays correct and incorrect predictions for each emotion/sentiment class.
    - Useful for identifying confusion between similar emotions (e.g., *anger* vs. *disgust*).
    - Visualized as a heatmap using Seaborn or Matplotlib.
- **Precision, Recall, F1-Score (per class)**:
    - Bar plots showing metrics for each emotion.
    - Helps assess whether the model favors certain classes.
- **ROC and AUC Curves (for multi-label tasks)**:
    - Plot ROC curves for each label to evaluate classification thresholds.
    - Higher AUC indicates better separability.

2. Emotion and Sentiment Distribution (Predicted vs. Actual)

- **Bar Charts or Stacked Bars**:
    - Compare the predicted vs. actual distribution of emotions and sentiment.
    - Reveals any bias toward overpredicting dominant classes (e.g., neutral sentiment).
- **Pie Charts**:
    - Show the proportion of predicted emotions in the test dataset.
    - Good for summarizing results visually in presentations.

3. Model Confidence Visualization

- **Prediction Probability Histograms**:

  - Show how confident the model is for each prediction.
  - Identify if the model is overconfident on wrong predictions.
- **Calibration Plots**:
  - Analyze if predicted probabilities match true outcomes.
  - Well-calibrated models yield more reliable predictions.

4. Attention and Interpretability

- **Attention Heatmaps (Transformer Models)**:
  - Visualize which words the model "attended to" when predicting a specific emotion.
  - Helps understand why a particular emotion was predicted.
- **SHAP Values / LIME Explanations**:
  - Show word-level contribution to the final prediction.
  - Useful for identifying emotionally charged keywords influencing the output.

5. Word Cloud Visualization

- **Emotion-wise Word Clouds**:
  - Create separate word clouds for each predicted emotion class.
  - Highlight the most frequent or most influential words associated with emotions like *joy*, *fear*, *surprise*, etc.

6. Temporal Trends of Emotions

- **Time Series Plots**:
  - Track the frequency of each predicted emotion over time.
  - Useful for analyzing how public emotion shifts in response to major events or news.

7. Embedding Space Visualization

- **t-SNE or PCA Plot**:

- Reduce word or sentence embeddings to 2D.
- Plot them with color-coded emotion labels to observe clustering.
- Indicates how well the model differentiates between emotional tones in vector space.

8. Error Analysis Dashboard (Optional Advanced Feature)

- **Interactive Tables/Charts**:
  - Display misclassified samples with original text, true label, predicted label, and confidence score.
  - Helps in identifying patterns in incorrect predictions.

10.Tools for Visualization

- **Libraries**:
  - Matplotlib, Seaborn, Plotly, Altair
  - SHAP, LIME for explainable AI
  - WordCloud
  - TensorBoard or Weights & Biases for training visualizations

1. Programming Language

- Python
  - Chosen for its rich ecosystem of NLP, machine learning, and data science libraries.

2. Data Collection and Storage

- Tweepy / snscrape – For extracting tweets and public posts from Twitter.
- PRAW (Python Reddit API Wrapper) – For collecting Reddit comments and threads.
- Facebook Graph API – (if used) for collecting Facebook posts.
- MongoDB / SQLite / CSV / JSON – For storing and managing raw social media data.

3. Text Preprocessing

- NLTK (Natural Language Toolkit) – Tokenization, stopword removal, stemming, and basic text cleaning.

- spaCy – Advanced tokenization, lemmatization, and POS tagging.
- re (Regex Library) – For pattern-based text cleaning.
- emoji / emot – For extracting and interpreting emojis.

4. Feature Engineering

- Scikit-learn – For TF-IDF vectorization, feature selection, and traditional ML pipelines.
- TextBlob / VADER – For extracting lexicon-based sentiment features.
- WordCloud – To visualize prominent terms in text grouped by emotion.

5. Machine Learning & Deep Learning

- Scikit-learn – Traditional models (Logistic Regression, SVM, Random Forest).
- Keras / TensorFlow – For building and training deep learning models (LSTM, CNN).
- PyTorch – For custom model architectures and flexibility in experimentation.

6. Transformer Models and NLP Pipelines

- Hugging Face Transformers – Pre-trained language models like BERT, RoBERTa, DistilBERT for emotion classification.
- Tokenizers Library (Hugging Face) – Efficient tokenization compatible with transformer models.

7. Model Evaluation and Visualization

- Matplotlib / Seaborn – For plotting metrics, confusion matrices, and distribution graphs.
- Plotly / Altair – Interactive visualizations.
- SHAP / LIME – Model explainability and interpretability.
- t-SNE / PCA (via Scikit-learn) – For visualizing high-dimensional embeddings.

8. Model Tracking and Experimentation

- TensorBoard – For tracking model training and visualizing learning curves.
- Weights & Biases (wandb) – For experiment logging and hyperparameter tuning (optional).

9. Deployment Tools (Optional)

- Flask / FastAPI – To build an API for real-time emotion prediction.
- Docker – For containerizing the application.

Cloud/Hardware Resources (If applicable)

- Google Colab / Kaggle Kernels – For model training and experimentation with GPU support.
- AWS / Azure / GCP – For scalable deployment and storage (optional).

11.Team Members and Roles:

TEAM MEMBERS	ROLES
Krishnakumar K	**Data cleaning , EDA**
Justin Jerold A	**Feature engineering**
Manu M	**Model development**
Dhinesh kumar M	**Documentation and reporting**