Student Name: M VIGNESH

Register Number: 422323205065

Institution: Thiruvalluvar college of Engineering and Technology

Department: Information Technology

Date of Submission: 10-05-2025

Git:hub Repository
Link:https://github.com/vignesh17704/Myproject.git

Title: Decoding emotions through sentiment analysis of social media conversations

1. Problem Statement

In the digital age, social media platforms have become a primary outlet for people to express their opinions, emotions, and sentiments. Analyzing these conversations can offer valuable insights into public mood, brand perception, political opinions, and societal trends.

This project aims to build a machine learning-based sentiment analysis model to classify emotions and sentiments embedded in social media text data. The problem is framed as a classification task, where each post or tweet is labeled as expressing a specific emotion or polarity (e.g., positive, negative, neutral, joy, anger, etc.).

The potential applications of this project span mental health monitoring, brand reputation management, and social research.

2. Project Objectives

- Build a robust sentiment classification model using social media conversation data.

- Distinguish between multiple sentiment classes (e.g., joy, sadness, anger, neutral).

- Leverage Natural Language Processing (NLP) techniques for feature extraction.

- Visualize key patterns and emotional trends over time.

- Integrate a Gradio-based interface for real-time sentiment detection.

3. Flowchart of the Project Workflow

4. Data Description

- Dataset Name: Twitter Sentiment/Emotion Dataset (e.g., Sentiment140, Emotion Dataset, or Kaggle dataset)

- Source: Kaggle / Twitter API / Other open-source datasets

- Type of Data: Unstructured textual data

- Records and Features: ~50,000+ text samples with labeled sentiments/emotions

- Target Variable: Sentiment class (e.g., positive, negative, neutral or emotion labels)

- Static or Dynamic: Static (can be updated via API)

- Attributes Covered: Tweet/post content, user metadata (optional), timestamp

5. Data Preprocessing

- Removed URLs, mentions, hashtags, and emojis

- Converted text to lowercase and removed stop words

- Performed stemming and lemmatization

- Handled class imbalance using oversampling (e.g., SMOTE)

- Tokenized text and padded sequences for input into ML models

- Encoded labels using one-hot or ordinal encoding

6. Exploratory Data Analysis (EDA)

- Univariate Analysis:

  - Bar charts showing sentiment distribution

  - Word clouds for each sentiment class

- Bivariate Analysis:

  - Sentiment trends across time or geography (if timestamp/geolocation available)

  - Co-occurrence of certain keywords with sentiment labels

Key Insights:

- Positive sentiments were more frequent than negative in the sampled dataset.

- Specific words (e.g., “happy”, “love”, “hate”) strongly correlated with sentiment classes.

- Neutral texts often contain factual or promotional content.

7. Feature Engineering

- TF-IDF and Count Vectorization for initial experiments

- Used pre-trained word embeddings (e.g., GloVe, BERT) for advanced modeling

- Created sentiment polarity and subjectivity scores using TextBlob

- Extracted n-grams and POS tags for additional context

8. Model Building

Algorithms Used:

- Logistic Regression and Naive Bayes (baseline)

- Support Vector Machine (SVM)

- Random Forest Classifier

- Deep Learning (LSTM, BERT)

Model Selection Rationale:

- Naive Bayes and Logistic Regression for their speed and baseline benchmarking

- SVM for handling high-dimensional text data

- LSTM/BERT for context-aware deep learning performance

Train-Test Split:

- 80% training, 20% testing

- Stratified split to preserve class distribution

Evaluation Metrics:

- Accuracy
- Precision, Recall, F1-Score
- Confusion Matrix
- ROC-AUC for binary classification

9. Visualization of Results & Model Insights

- Confusion matrix heatmaps for performance analysis
- Precision-Recall curves comparing multiple models

- Feature importance visualized for classical ML models
- Word embeddings visualization using t-SNE or PCA
- Real-time prediction interface developed using Gradio

10. Tools and Technologies Used

- Programming Language: Python 3
- Notebook Environment: Google Colab / Jupyter Notebook
- Key Libraries:
  - pandas, numpy – data handling
  - matplotlib, seaborn, wordcloud – visualizations
  - scikit-learn, nltk, TextBlob, spaCy – NLP and ML

- tensorflow, transformers – Deep learning models

- Gradio – Interface deployment