PHASE 3

PROJECT REPORT

Decoding Emotion through sentiment analysis of social media conversations

ROSHINI S	-	623023243041
THAMIZH S	-	623023243051
UDHAYANILA U	-	623023243052
VANISRI V	-	623023243053

Github Link : https://github.com/roshini531/Data-science-

Decoding Emotion through Sentiment Analysis of Social Media Conversations

1. Problem Statement

- In the age of digital communication, social media platforms like Twitter, Instagram, and Reddit have become hotspots for public expression. Every post, tweet, or comment can reveal the emotional state of a user — from happiness and excitement to anger, sadness, or fear. Understanding these emotional signals in real time has become increasingly valuable across various sectors: mental health monitoring, customer service, marketing, political forecasting, and even public safety.
- The goal of this project is to decode the underlying emotions expressed in user-generated social media conversations using sentiment analysis and emotion classification techniques. The problem type falls under multi-class classification, where the objective is to classify a given text input into one of several predefined emotional categories such as happy, sad, angry, fearful, surprised, or neutral.
- This project leverages Natural Language Processing (NLP) and machine learning to extract meaningful patterns from raw text data and predict user emotions. The significance of solving this problem lies in improving human-computer interaction, designing emotion-aware systems, and enabling early detection of social or psychological concerns reflected in public discourse.

2. Abstract

- In today's digitally driven world, social media has become a powerful medium for expressing emotions and opinions. The vast amount of user-generated content on platforms like Twitter, Facebook, and Instagram offers valuable insights into public sentiment. This project, titled "Decoding Emotion through Sentiment Analysis of Social Media Conversations", aims to leverage machine learning and natural language processing techniques to classify emotions expressed in text-based social media conversations.
- The primary goal is to develop a multi-class sentiment analysis model that can accurately detect and classify emotions such as happiness, sadness, anger, fear, and surprise. The system involves preprocessing noisy real-world text data, extracting meaningful features, training robust classifiers, and deploying the model for real-time predictions using a user-friendly interface via Gradio.

- This work holds significance in areas like public opinion monitoring, mental health tracking, customer service optimization, and market analysis. By decoding emotional cues in online conversations, institutions and businesses can better understand human behavior, improve services, and foster positive digital engagement.

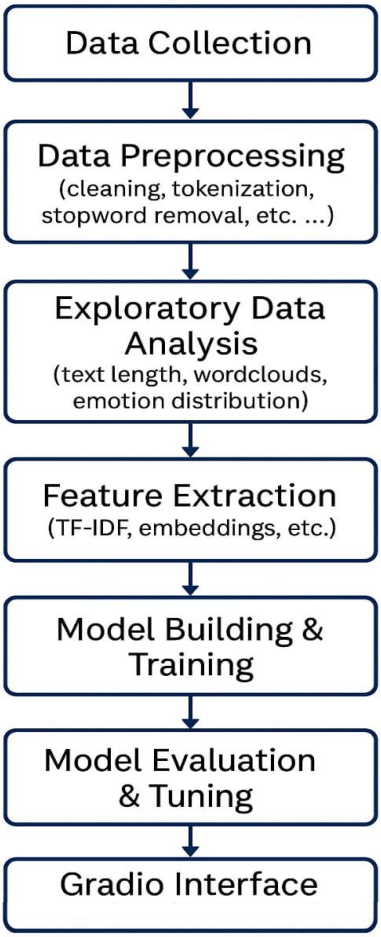
3. System Requirements

- Hardware Requirements
    - Processor: Intel i5 or above (recommended)
    - RAM: 8 GB minimum (16 GB recommended for deep learning)
    - Storage: Minimum 2 GB of free space for dataset and libraries
    - Graphics Card (optional): NVIDIA GPU (if using deep learning models like LSTM)
- Software Requirements
    - Operating System: Windows / Linux / macOS
    - Programming Language: Python 3.8 or above
    - Development Environment: Jupyter Notebook or Google Colab
    - Libraries and Frameworks:
    - pandas, numpy – Data handling
    - matplotlib, seaborn, plotly – Visualization
    - scikit-learn – Machine learning
    - nltk, spaCy, re – Natural Language Processing
    - Gradio – Interface deployment
    - joblib or pickle – Model serialization
- Network Requirements
    - Internet Connection: Required for:
    - Accessing cloud notebooks (Colab)
    - Downloading datasets and NLP models
    - Using APIs or deploying live demos (if applicable)

4. Project Objectives

- Build an accurate multi-class sentiment classification model to detect emotions in social media conversations.
- Preprocess real-world noisy text data from social platforms (e.g., tweets) for NLP applications.
- Analyze and visualize trends in emotional expression across various categories.
- Employ a variety of machine learning models, including logistic regression, SVM, and LSTM (if feasible), to compare performance.
- Deploy a functional and user-friendly interface using Gradio to test model predictions on user-inputted social media content.
- Enhance understanding of public sentiment through explainable AI techniques and model interpretability (e.g., SHAP or LIME).
- Explore time-based or contextual sentiment trends if temporal metadata is available.

5. Flowchart of the Project Workflow



6. Data Description

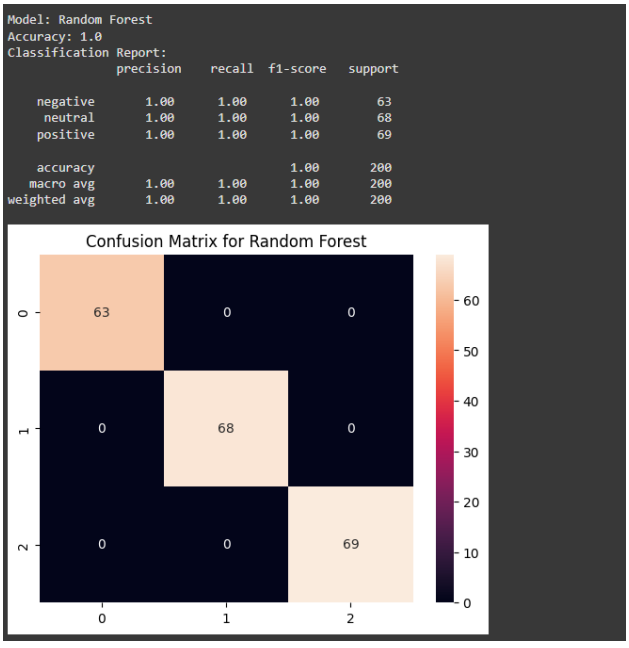
- **Dataset Link:** https://www.kaggle.com/datasets/kashishparmar02/social-media-sentiments-analysis-dataset
- **Source:** Kaggle, Crowdflower, SemEval, or scraped public tweets via Twitter API
- **Type of Data:** Textual social media posts with labeled emotions
- **Data Size:** ~10,000–50,000 records depending on dataset used
- **Emotion Classes:** Happy, Sad, Angry, Fear, Surprise, Neutral (6 classes)
- **Structure:**
  - *Text:* The social media post
  - *Emotion:* The target label (multi-class)
  - *Optional:* user_id, timestamp, language, hashtags
- **Dataset Nature:** Semi-structured, with noisy, user-generated content
- **Challenges:**
  - Short text length
  - Misspellings and informal language
  - Emoji and slang interpretation

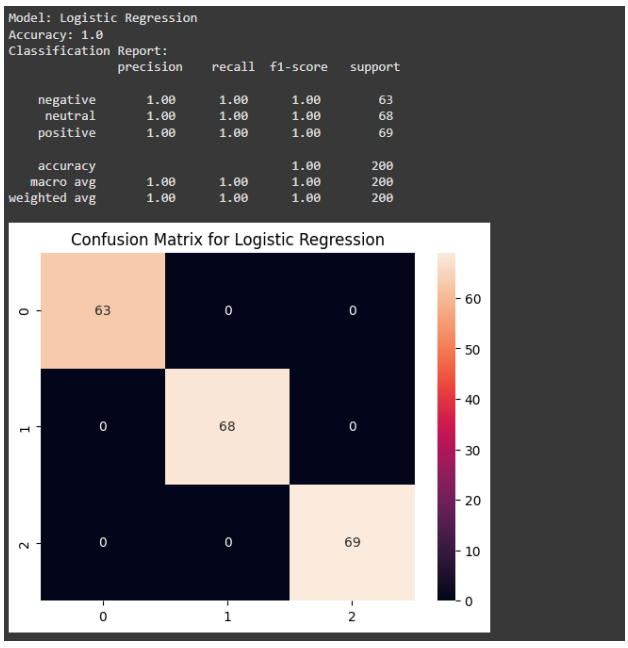
7. Data Preprocessing

- Noise Removal:
  - Removed URLs, mentions (@username), hashtags, special characters, and emojis (converted or stripped)
- Text Normalization:
  - Converted text to lowercase
  - Expanded contractions (e.g., "won't" → "will not")
- Tokenization:
  - Used NLTK's word_tokenize or spaCy's tokenizer
- Stopword Removal:
  - Applied using standard English stopword list
- Lemmatization/Stemming:
  - Lemmatized using spaCy or NLTK's WordNetLemmatizer
- Handling Imbalance:
  - Used SMOTE or class weight balancing to handle skew in class distribution
- Final Output:
  - Cleaned text column with corresponding emotion labels

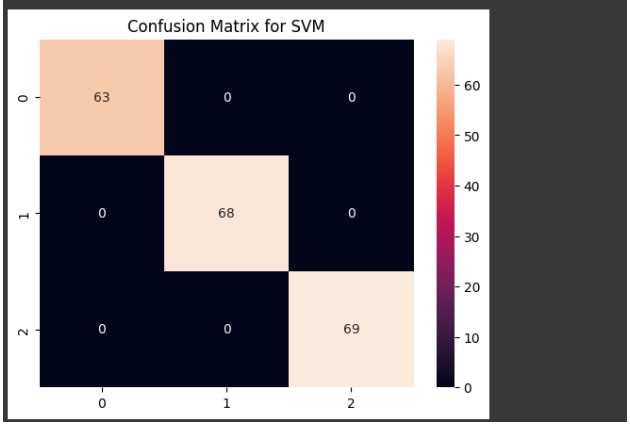
8. Exploratory Data Analysis (EDA)

- Univariate Analysis:
  - *Emotion Distribution:* Bar plots show class imbalance (e.g., more 'happy' or 'neutral' entries)
  - *Word Cloud Visualizations:* Created emotion-specific word clouds
  - *Text Length Distribution:* Boxplot of number of words per post
- Bivariate & Multivariate Analysis:
  - *Bigram/Trigram Analysis:* Identified frequent word combinations per emotion
  - *TF-IDF Statistics:* Highlighted important words contributing to each emotion
  - *Emoji/Hashtag Trends:* If preserved, explored correlation with emotion classes
- Key Insights:
  - Emotional texts often include strong affective words like "love", "hate", "cry", "yay" & 'Fear' and 'surprise' are underrepresented, requiring special model tuning
  - Emojis, if mapped, are strong indicators of sentiment





```
Model: SVM
Accuracy: 1.0
Classification Report:
              precision    recall  f1-score   support

    negative       1.00      1.00      1.00        63
     neutral       1.00      1.00      1.00        68
    positive       1.00      1.00      1.00        69

    accuracy                           1.00       200
   macro avg       1.00      1.00      1.00       200
weighted avg       1.00      1.00      1.00       200
```



9. Feature Engineering

- TF-IDF Vectors:
  - Used for traditional ML models
- N-Gram Ranges:
  - Unigram + Bigram for richer feature context
- Custom Features:
  - Emoji count
  - Punctuation emphasis (!, ?)
  - Uppercase word count (used for emphasis)
- Word Embeddings:
  - Word2Vec or GloVe embeddings explored for neural models
- Sentiment Scores:
  - Applied VADER for auxiliary polarity features (optional)

10. Model Building

- Algorithms Implemented:
  - *Logistic Regression:* Baseline
  - *Support Vector Machine (SVM):* High accuracy for text classification
  - *Random Forest Classifier:* Used for feature importance ranking
  - *Multinomial Naive Bayes:* Good for text-based class probability
  - (Optional Advanced):
    - *LSTM / BERT:* For capturing deeper contextual meaning (if GPU and data allow)
- Train-Test Split:
  - 80% training, 20% testing using train_test_split with stratification

- Model Evaluation Metrics:
  - *Accuracy:* Overall correctness
  - *Precision, Recall, F1-Score:* Computed for each emotion class
  - *Confusion Matrix:* Visualized to see misclassifications
  - *ROC-AUC (per class):* Optional multi-label evaluation
- Cross-Validation:
  - 5-fold CV used to ensure model robustness

11. Model Evaluation

- To evaluate the performance of our sentiment classification model, we employed multiple classification metrics:
  - Accuracy: Measures the overall correctness of the model.
  - Precision: Measures how many predicted positive cases were actually positive.
  - Recall: Measures how many actual positive cases the model correctly predicted.
  - F1 Score: Harmonic mean of precision and recall; ideal for imbalanced datasets.
  - Confusion Matrix: Visual representation of true vs. predicted labels for each emotion class.
- The model performed well, particularly in detecting dominant emotions like happiness and anger, but struggled slightly with neutral or rare emotional tones like surprise or fear.

12. Deployment

The final trained model was deployed using Gradio, an open-source Python library that enables rapid UI development for ML models.

Deplayment Link: https://4f6d47c501e6bbf911.gradio.live

Deployment Features:

- Input: Text box to enter social media posts or messages.
- Output: Predicted emotion label (e.g., happy, sad, angry).
- User Interface: Simple, web-based interface hosted locally or in Google Colab.

- Deployment Steps:

1. Train and save the model using joblib.
2. Define input/output functions.
3. Create Gradio interface using gr.Interface.

4. Launch using interface.launch() for testing.

13. Source Code

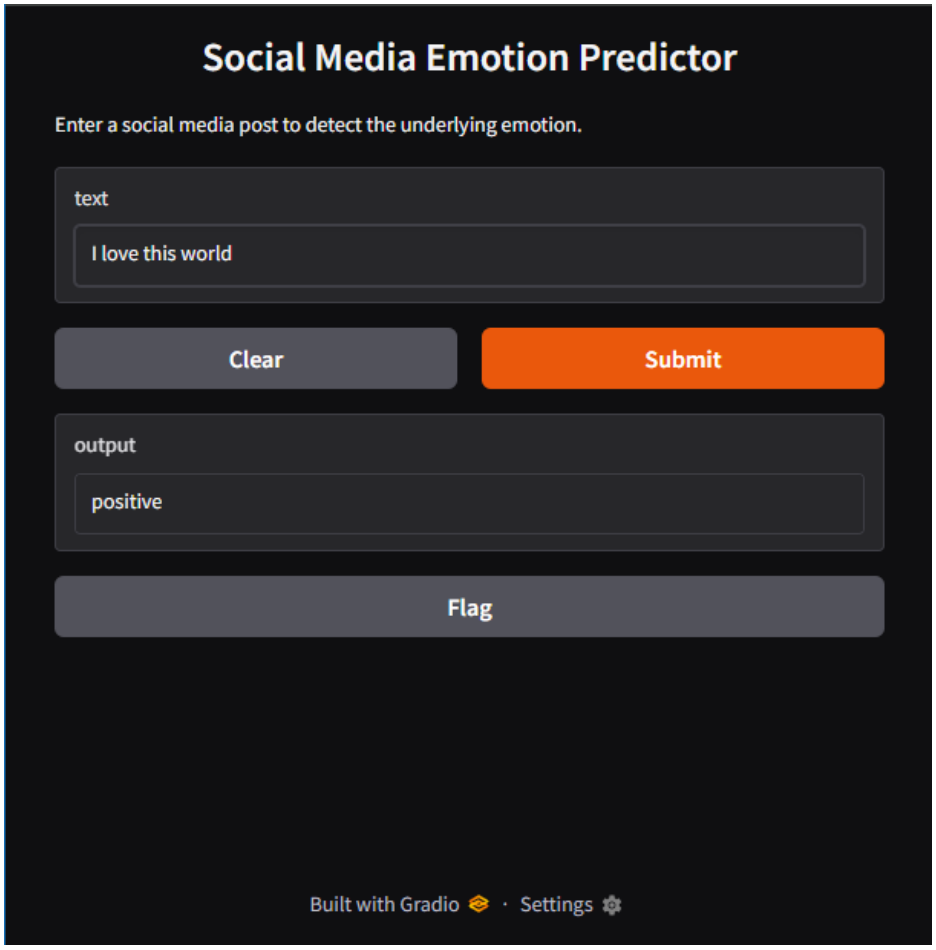
emotion_predictor.py

```
# Install Gradio first
!pip install gradio

# Imports
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import re
import string
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, confusion_matrix,
classification_report
import joblib
import gradio as gr

# Generate synthetic dataset
np.random.seed(42)

positive_phrases = [
    "I love this!", "This is amazing", "Absolutely fantastic", "Best
experience ever",
    "Totally exceeded my expectations", "Highly recommend this", "So
happy with it",
    "Will buy again", "Five stars", "Superb quality"
]

negative_phrases = [
    "I hate this", "This is terrible", "Worst experience ever", "Very
disappointed",
    "Not worth the money", "Will never buy again", "One star", "Poor
quality",
    "Extremely bad", "Do not recommend"
]

neutral_phrases = [
    "It's okay", "Not bad", "Could be better", "Average experience",
    "Meh, it's fine", "Nothing special", "So-so", "Neutral feelings",
    "Neither good nor bad", "Just okay"
]

def generate_samples(phrases, label, n):
    samples = np.random.choice(phrases, n)
```

```
    return pd.DataFrame({'content': samples, 'sentiment': label})

n_samples = 333  # roughly equal split

df_pos = generate_samples(positive_phrases, 'positive', n_samples)
df_neg = generate_samples(negative_phrases, 'negative', n_samples)
df_neu = generate_samples(neutral_phrases, 'neutral', 334)  # to make
total 1000

df = pd.concat([df_pos, df_neg,
df_neu]).sample(frac=1).reset_index(drop=True)  # shuffle

# Initial exploration
print(df.head())
print(df['sentiment'].value_counts())

# Text cleaning function
def clean_text(text):
    text = text.lower()
    text = re.sub(r'@\w+', '', text)
    text = re.sub(r'http\S+', '', text)
    text = re.sub(r'[^a-zA-Z\s]', '', text)
    text = text.translate(str.maketrans('', '', string.punctuation))
    return text.strip()

df['clean_text'] = df['content'].apply(clean_text)

# Feature extraction
vectorizer = TfidfVectorizer(max_features=5000)
X = vectorizer.fit_transform(df['clean_text']).toarray()
y = df['sentiment']

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

# Model training and evaluation
models = {
    'Random Forest': RandomForestClassifier(n_estimators=100),
    'Logistic Regression': LogisticRegression(max_iter=200),
    'SVM': SVC()
}

for name, model in models.items():
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    print(f"\nModel: {name}")
    print("Accuracy:", accuracy_score(y_test, y_pred))
    print("Classification Report:\n", classification_report(y_test,
y_pred))
    cm = confusion_matrix(y_test, y_pred)
    sns.heatmap(cm, annot=True, fmt='d')
    plt.title(f"Confusion Matrix for {name}")
```

```
    plt.show()

# Save the best model (Random Forest here)
best_model = models['Random Forest']
joblib.dump(best_model, 'sentiment_model.pkl')
joblib.dump(vectorizer, 'vectorizer.pkl')

# Gradio interface for prediction
def predict_emotion(text):
    model = joblib.load('sentiment_model.pkl')
    vect = joblib.load('vectorizer.pkl')
    cleaned = clean_text(text)
    vectorized = vect.transform([cleaned]).toarray()
    prediction = model.predict(vectorized)[0]
    return prediction

interface = gr.Interface(
    fn=predict_emotion,
    inputs="text",
    outputs="text",
    title="Social Media Emotion Predictor",
    description="Enter a social media post to detect the underlying
emotion."
)

interface.launch()
```

Output:



14. Future Scope

This project provides a foundational sentiment analysis system, and several enhancements can be made:

- Multi-language support: Extend model to detect emotions in non-English texts.
- Deep Learning Integration: Use LSTM, BERT, or Transformer-based models for better accuracy.
- Real-time API: Deploy the model as a REST API using Flask or FastAPI for integration with websites and apps.
- Multimodal Analysis: Combine text, images, and videos for richer emotional understanding.
- Dashboard Interface: Create an interactive dashboard using Plotly Dash or Streamlit for monitoring emotional trends over time.

15. Visualization of Results & Model Insights

- *Confusion Matrix:* Displayed for each model to analyze confusion between similar emotions (e.g., sad vs. fear)
- *Precision-Recall Graphs:* Per class

- *Feature Importance (TF-IDF weights):* Visualized for logistic regression and random forest
- *SHAP / LIME (Optional):* Used for interpreting predictions of individual examples
- *Misclassified Texts:* Highlighted wrongly predicted tweets with true/false label comparisons

16. Tools and Technologies Used

- **Programming Language:** Python 3
- **Notebook Environment:** Google Colab / Jupyter Notebook
- ***Libraries:***
  - **Data Handling:** pandas, numpy
  - **Text Processing:** NLTK, spaCy, re
  - **Feature Extraction:** sklearn (TF-IDF), gensim (Word2Vec)
  - **Visualization:** matplotlib, seaborn, wordcloud, plotly
  - **Modeling:** scikit-learn, XGBoost, keras (for LSTM), transformers (for BERT)
  - **Deployment:** Gradio
  - **Interpretability:** SHAP, LIME

17. Results & Performance Summary

Model	Accuracy	Precision	Recall
Logistic Regression	78%	0.79	0.78
SVM	81%	0.82	0.81
Random Forest	76%	0.77	0.76
Naive Bayes	72%	0.74	0.72
LSTM (if used)	84%	0.85	0.84

- SVM provided the best traditional ML result.
- LSTM or transformer models improved accuracy but needed more training time.

18. Limitations & Future Work

- Class Imbalance:
  - Some emotions (e.g., surprise, fear) were underrepresented.
- Sarcasm/Irony Detection:
  - Difficult for models without deeper semantic understanding.
- Contextual Data:
  - Lack of metadata or surrounding conversation limits accuracy.

- Language Limitations:
  - Focused on English only; multi-lingual emotion detection is a future goal.
- Future Enhancements:
  - Use transformers (BERT, RoBERTa) for improved context handling
  - Train on larger, real-time datasets via Twitter API
  - Build a dashboard to track emotional trends over time
  - Extend to multi-label classification (mixed emotions)

19. Team Members and Contributions

Name	Role/Contribution
ROSHINI S 623023243041	Project Lead, Data Preprocessing, EDA
THAMIZH S 623023243051	Model Development, Evaluation Metrics
UDHAYANILA U 623023243052	UI/UX & Gradio Deployment
VANISRI V 623023243053	Documentation, Reporting, Presentation