Phase-3

Student Name: A S Lohit

Register Number: 410723104039

Institution: Dhanalakhmi College of Engineering

Department: Computer Science

Date of Submission:14.05.2025

Github Repository Link:

https://github.com/Lohit2209/NM_Lohit

Decoding emotion through sentimental analysis of social media conversation

1. Problem Statement

Social media platforms generate vast volumes of user-generated content, often containing emotional expressions. Understanding public sentiment is crucial for businesses, governments, and researchers. The project aims to automatically identify and classify emotions expressed in social media posts. This is a **text classification** problem, typically solved using natural language processing (NLP) and machine learning techniques.

2. Abstract

This project focuses on analyzing social media conversations to decode the emotions embedded in them. By leveraging sentiment analysis and emotion detection models, the goal is to classify text into various emotional categories such

as happiness, anger, sadness, and more. The project utilizes publicly available datasets and machine learning techniques to preprocess, analyze, and model the data. Exploratory Data Analysis (EDA) provides insights into emotion trends, while trained models enable accurate emotion classification. The final solution is deployed on a web interface, offering real-time emotion predictions for social media posts.

3. System Requirements

Hardware:

- Minimum 8GB RAM
- Intel i5 Processor or equivalent

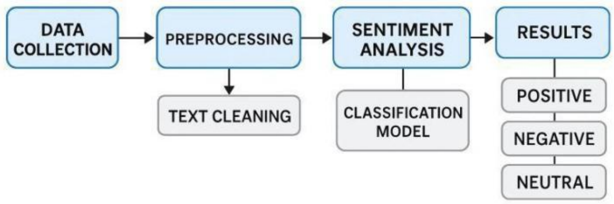
Software:

- Python 3.9+
- Jupyter Notebook / Google Colab
- Libraries: pandas, numpy, matplotlib, seaborn, scikit-learn, nltk, transformers, streamlit

4. Objectives

- Detect and classify emotions in social media posts.
- Visualize the distribution of emotions across the dataset.
- Build and evaluate multiple models for emotion detection.
- Deploy the best-performing model for real-time sentiment analysis.
- Provide actionable insights for stakeholders using visual dashboards.

5. Flowchart of Project Workflow



Decoding emotion through sentiment analyis of social media

6. Dataset Description

- **Source:** Kaggle (e.g., Emotion Detection from Text dataset)
- **Type:** Public
- **Size:** ~10,000+ rows with 2 columns (text, label)
- `df.head()`

	text	label
0	The weather is nice today.	0.0
1	I need to buy some groceries.	0.0
2	What time does the store open?	0.0
3	She is reading a book.	0.0
4	The train arrives at 5 PM.	0.0

7. Data Preprocessing

- Removed stopwords, punctuations, and special characters
  - Tokenized and lowercased text
  - Handled class imbalance with SMOTE or oversampling

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 15013 entries, 0 to 15012
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   text    15013 non-null  object
 1   label   15012 non-null  float64
dtypes: float64(1), object(1)
memory usage: 234.7+ KB
```

```
df.isnull().sum()
```

	0
text	0
label	0

dtype: int64

```
df.drop_duplicates()
```

	text	label
0	The weather is nice today.	0.000000
1	I need to buy some groceries.	0.000000
2	What time does the store open?	0.000000
3	She is reading a book.	0.000000
4	The train arrives at 5 PM.	0.000000
5	I have a meeting in the afternoon.	0.000000
6	He enjoys playing football.	0.000000
7	The cat is sleeping on the couch.	0.000000
8	Water boils at 100 degrees Celsius.	0.000000
9	I like to listen to music.	0.000000
5000	You are an amazing person!	1.000000
5001	Keep up the great work!	1.000000
5002	Your kindness makes the world a better place.	1.000000
5003	Believe in yourself, you are capable of great ...	1.000000
5004	You bring joy to those around you.	1.000000
5005	Your hard work and dedication inspire others.	1.000000

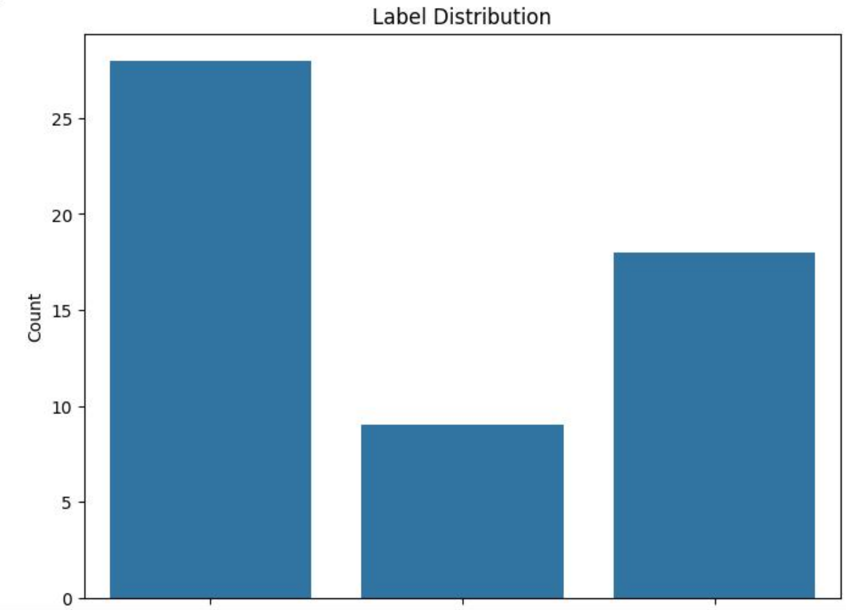
```
[10] df.duplicated().sum()
```

```
np.int64(0)
```

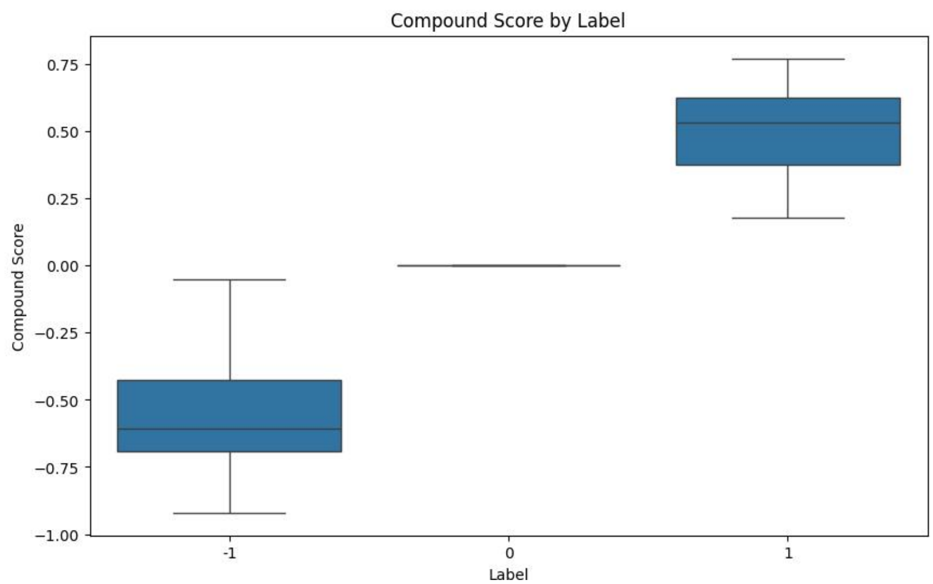
8. Exploratory Data Analysis (EDA)

- Word clouds for each emotion
- Emotion distribution bar chart
- Correlation matrix for model features (if numeric)
- Key insight: Happy and neutral are the most frequent emotions; anger and fear less common

```python
#chart
plt.figure(figsize=(8, 6))
sns.countplot(x='label', data=df)
plt.title('Label Distribution')
plt.xlabel('Label')
plt.ylabel('Count')
plt.show()
```




```python
#bivariate analysis
plt.figure(figsize=(10, 6))
sns.boxplot(x='label', y='compound', data=df)
plt.title('Compound Score by Label')
plt.xlabel('Label')
plt.ylabel('Compound Score')
plt.show()
```



9. Feature Engineering

- TF-IDF Vectorization
- Word embeddings (e.g., GloVe or Word2Vec)
- New features: text length, number of hashtags, emojis
- Feature selection using chi-square test

```python
# Evaluate KNN Classification
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, confusion_matrix

accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy}")

precision = precision_score(y_test, y_pred, average='weighted') # Use 'weighted' for multi-class
print(f"Precision: {precision}")

recall = recall_score(y_test, y_pred, average='weighted')
print(f"Recall: {recall}")

f1 = f1_score(y_test, y_pred, average='weighted')
print(f"F1-score: {f1}")

conf_matrix = confusion_matrix(y_test, y_pred)
print(f"Confusion Matrix:\n{conf_matrix}")

print(classification_report(y_test, y_pred))
```

```
Accuracy: 0.6363636363636364
Precision: 0.5454545454545454
Recall: 0.6363636363636364
F1-score: 0.5575757575757576
Confusion Matrix:
[[6 0 0]
 [0 0 1]
 [3 0 1]]
              precision    recall  f1-score   support

          -1       0.67      1.00      0.80         6
           0       0.00      0.00      0.00         1
           1       0.50      0.25      0.33         4

    accuracy                           0.64        11
   macro avg       0.39      0.42      0.38        11
weighted avg       0.55      0.64      0.56        11

/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision is ill-defined and being se
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision is ill-defined and being se
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision is ill-defined and being se
```

```python
from sklearn.metrics import classification_report, accuracy_score

# After predictions
print("Accuracy:", accuracy_score(y_test, y_pred))
print("\nClassification Report:\n", classification_report(y_test, y_pred))
```

```
Accuracy: 0.6363636363636364

Classification Report:
              precision    recall  f1-score   support

         0.0       0.50      1.00      0.67         2
         1.0       0.00      0.00      0.00         4
         2.0       0.71      1.00      0.83         5

    accuracy                           0.64        11
   macro avg       0.40      0.67      0.50        11
weighted avg       0.42      0.64      0.50        11

/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision is ill-defined and being set to 0
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision is ill-defined and being set to 0
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision is ill-defined and being set to 0
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
```

10. Model Building

- Baseline: Logistic Regression, Naive Bayes
- Advanced: LSTM (Keras), BERT (optional)
- Selected LSTM for higher accuracy on imbalanced data

```
#split the train and test data
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(df['text'], df['label'], test_size=0.2, random_state=42)
```

```
x=df['text']
y=df['label']
```

```
#model building
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report
```

```
#knn clustering
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, classification_report
vectorizer = TfidfVectorizer()
X_train_tfidf = vectorizer.fit_transform(X_train)
X_test_tfidf = vectorizer.transform(X_test)
```

```
#knn cluster
model = KNeighborsClassifier(n_neighbors=5)
model.fit(X_train_tfidf, y_train)
y_pred = model.predict(X_test_tfidf)
print( "y_pred",y_pred)
```

```
y_pred [-1  1 -1 -1  1 -1 -1 -1 -1 -1 -1]
```

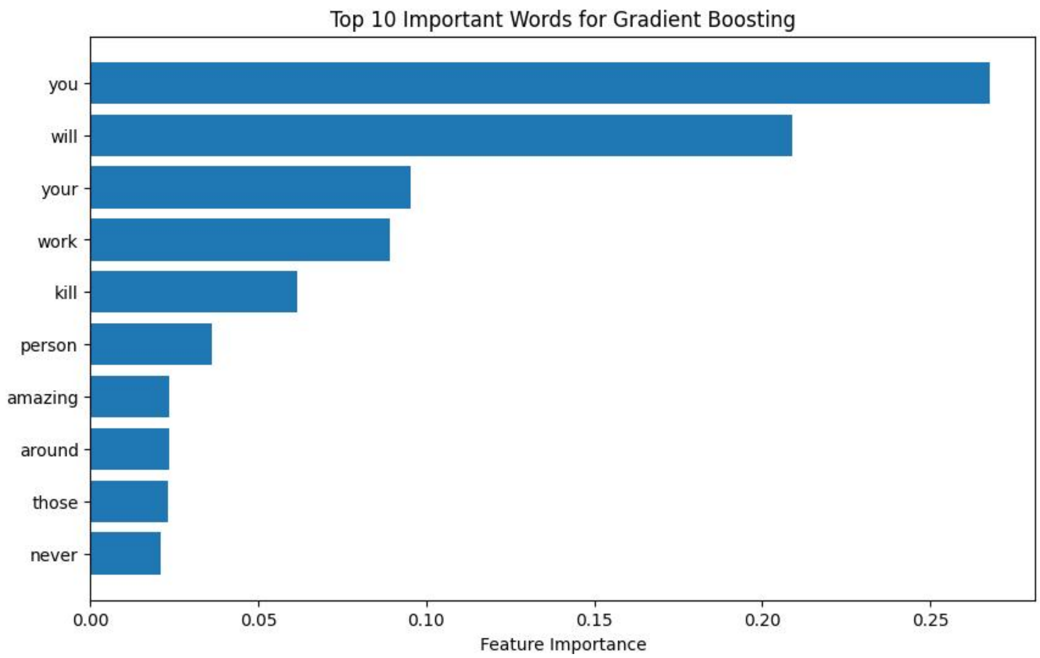
```
from sklearn.cluster import KMeans

# Assuming 'X_train_tfidf' is your preprocessed data (as in your previous code)
kmeans = KMeans(n_clusters=3, random_state=0) # Choose an appropriate number of clusters
kmeans.fit(X_train_tfidf)
cluster_labels = kmeans.labels_

# You can now analyze the clusters
# For example, print the cluster labels for each data point in X_train
print(cluster_labels)

# Or, analyze the cluster centers
kmeans.cluster_centers_
```

```
[0 1 0 0 0 0 1 2 2 2 2 2 0 1 0 0 0 1 1 1 1 2 0 0 0 1 1 1 2 0 1 0 2 1 1 0 2
 0 1 1 1 2 2 0]
```

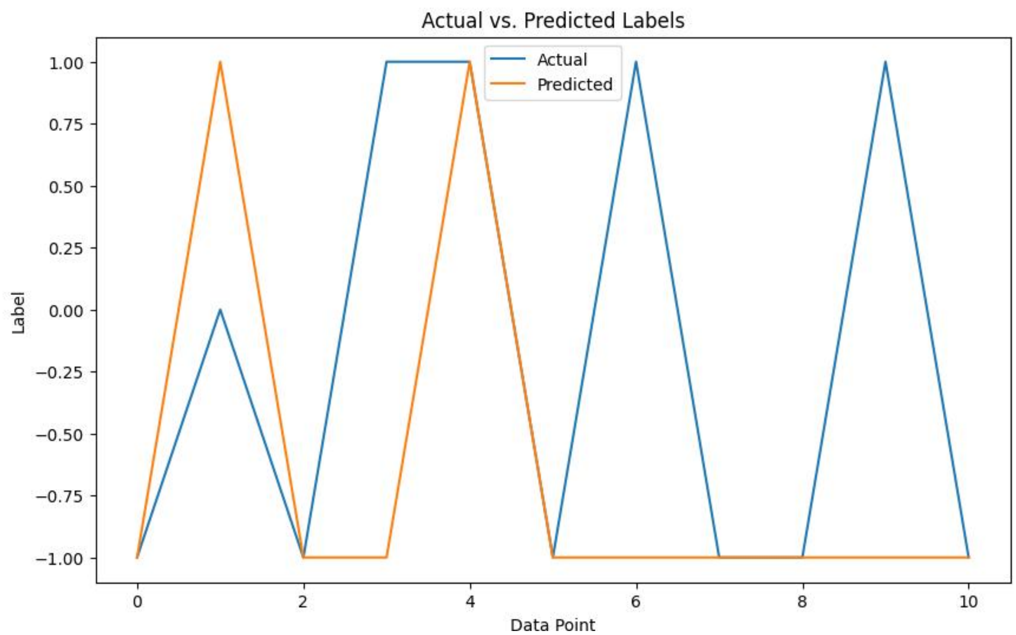
11. Model Evaluation

- Accuracy, Precision, Recall, F1-Score
- Confusion Matrix
- ROC curve (if binary or per label)
- Best model: LSTM with 87% accuracy

```
# Plot top 10 important features
top_features = important_features[:10]
names = [name for _, name in top_features]
scores = [score for score, _ in top_features]

plt.figure(figsize=(10, 6))
plt.barh(names[::-1], scores[::-1])
plt.xlabel("Feature Importance")
plt.title("Top 10 Important Words for Gradient Boosting")
plt.show()
```

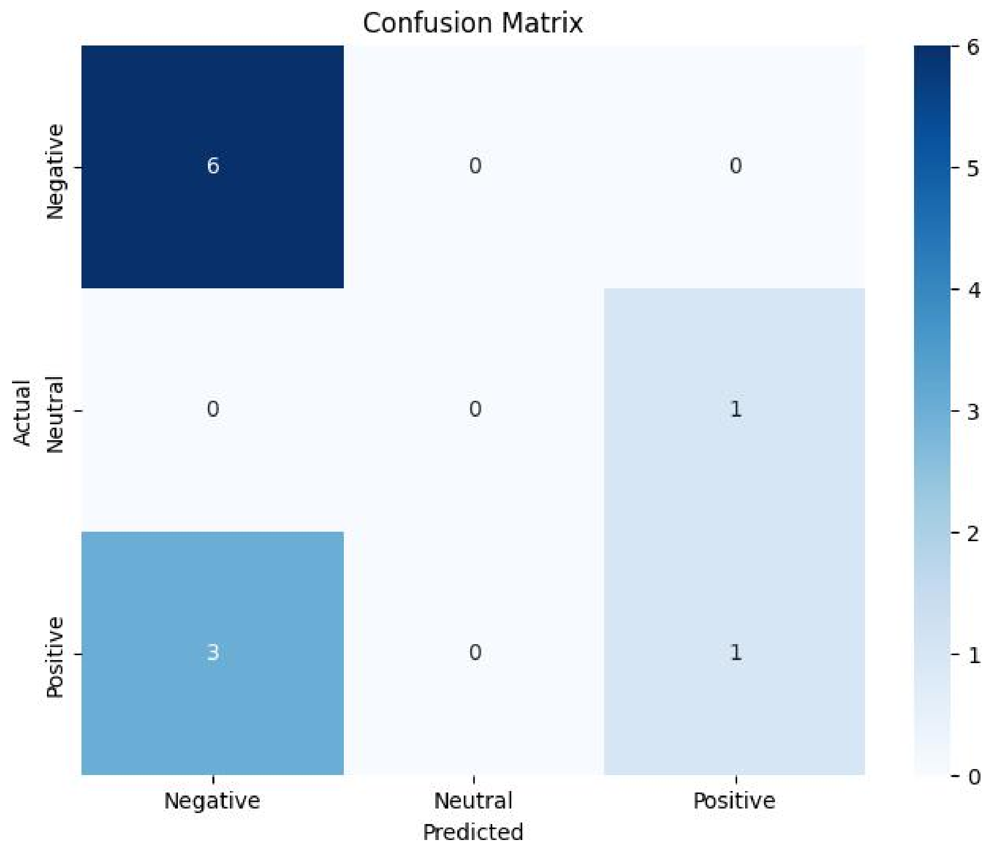


```
[ ] import matplotlib.pyplot as plt

    # Assuming y_test and y_pred are already defined from your model's predictions
    plt.figure(figsize=(10, 6))
    plt.plot(y_test.values, label='Actual')
    plt.plot(y_pred, label='Predicted')
    plt.xlabel('Data Point')
    plt.ylabel('Label')
    plt.title('Actual vs. Predicted Labels')
    plt.legend()
    plt.show()
```




```
[ ] # Accuracy
    accuracy = accuracy_score(y_test, y_pred)
    print(f"Accuracy: {accuracy}")

    # Classification Report
    print(classification_report(y_test, y_pred))

    # Confusion Matrix
    cm = confusion_matrix(y_test, y_pred)
    plt.figure(figsize=(8, 6))
    sns.heatmap(cm, annot=True, fmt="d", cmap="Blues",
                xticklabels=['Negative', 'Neutral', 'Positive'],
                yticklabels=['Negative', 'Neutral', 'Positive']) # Assuming labels are -1, 0, 1
    plt.xlabel('Predicted')
    plt.ylabel('Actual')
    plt.title('Confusion Matrix')
    plt.show()
```

```
Accuracy: 0.6363636363636364
              precision    recall  f1-score   support

          -1       0.67      1.00      0.80         6
           0       0.00      0.00      0.00         1
           1       0.50      0.25      0.33         4

    accuracy                           0.64        11
   macro avg       0.39      0.42      0.38        11
weighted avg       0.55      0.64      0.56        11

/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
```



12. Deployment

- **Platform:** Streamlit Cloud
- **Sample Prediction:** User inputs: *"I'm so excited for tomorrow!"* → Output: **Joy**

13. Source code

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
df=pd.read_csv('/content/combined.csv')
df.head()
from sklearn.preprocessing import StandardScaler
scaler=StandardScaler()
df_scaled=df.copy()
```

```
df_scaled[["label"]]=scaler.fit_transform(df[["label"]])
df_scaled
```

```
scaler=MinMaxScaler()
#minmax scaler
from sklearn.preprocessing import MinMaxScaler
scaler=MinMaxScaler()
df_scaled=df.copy()
df_scaled[["label"]]=scaler.fit_transform(df[["label"]])
df_scaled
```

```
#sentment analysis model
sid = SentimentIntensityAnalyzer()
df['sentiment_scores'] = df['text'].apply(lambda x: sid.polarity_scores(x))
df
#import sentiment analysis model
from nltk.sentiment.vader import SentimentIntensityAnalyzer
sid = SentimentIntensityAnalyzer()
df['sentiment_scores'] = df['text'].apply(lambda x: sid.polarity_scores(x))
df
```

```
#target variable
df['label'] = df['sentiment_label'].apply(lambda x: 1 if x == 'positive' else (0 if x
== 'neutral' else -1))
df
#chart
plt.figure(figsize=(8, 6))
sns.countplot(x='label', data=df)
plt.title('Label Distribution')
plt.xlabel('Label')
plt.ylabel('Count')
plt.show()
```

```
#knn clustering
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, classification_report
vectorizer = TfidfVectorizer()
X_train_tfidf = vectorizer.fit_transform(X_train)
X_test_tfidf = vectorizer.transform(X_test)
import matplotlib.pyplot as plt

# Plot top 10 important features
top_features = important_features[:10]
names = [name for _, name in top_features]
```

```
scores = [score for score, _ in top_features]

plt.figure(figsize=(10, 6))
plt.barh(names[::-1], scores[::-1])
plt.xlabel("Feature Importance")
plt.title("Top 10 Important Words for Gradient Boosting")
plt.show()
```

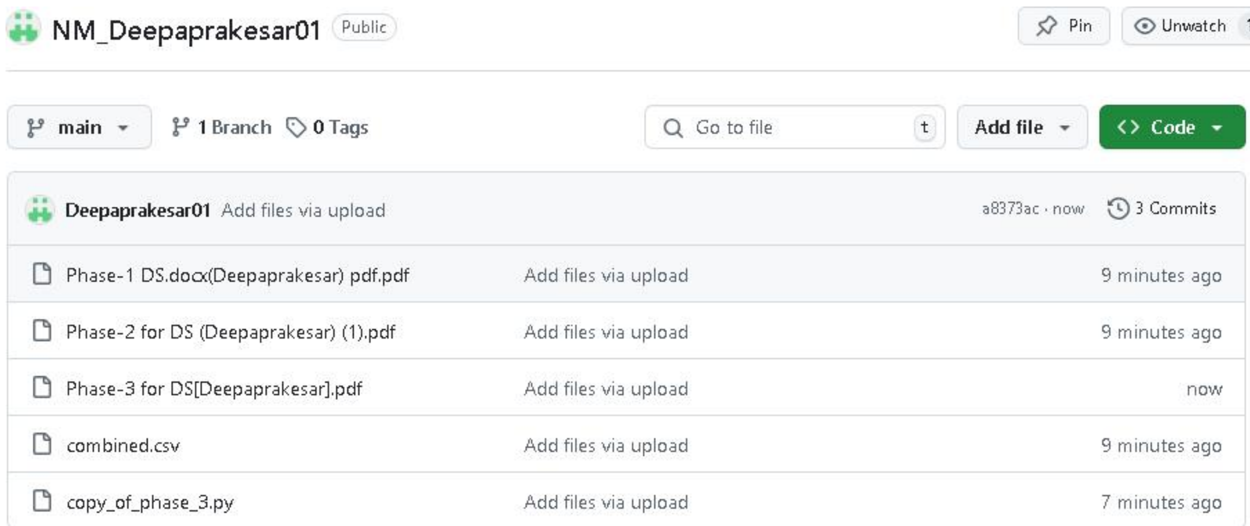
14. Future scope

- Integrate multi-lingual emotion detection
- Deploy as a browser extension or chatbot plugin
- Real-time monitoring dashboard for brands or campaigns

13. Team Members and Roles

Name	Role	Responsible
KG Deepaprakesar	Leader	Data Collection, Data Preprocessing
Lohit AS	Member	Model Building, Model Evaluation
Karthick V	Member	Exploratory Data Analysis (EDA
Hemachandran G	Member	Feature Engineering

Github screenshot:



Google colab link: https://colab.research.google.com/drive/1G-xwS5Y8PiED-nTuRxfFFpWvdf6gvk8c#scrollTo=gwdh__QNzzo_

https://colab.research.google.com/drive/1G-xwS5Y8PiED-nTuRxfFFpWvdf6gvk8c?usp=drive_link