In [ ]:

```
import pandas as pd
import numpy as np
import nltk
from nltk.corpus import stopwords
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.preprocessing import LabelEncoder
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout
from tensorflow.keras.callbacks import EarlyStopping
from tensorflow.keras.utils import to_categorical
from sklearn.metrics import accuracy_score, classification_report
import re

nltk.download('stopwords')
stop_words = set(stopwords.words('english'))

def load_dataset(file_path):
    try:
        df = pd.read_csv(file_path, encoding='latin1')
        return df
    except FileNotFoundError:
        print(f"Error: File '{file_path}' not found.")
        raise
    except pd.errors.EmptyDataError:
        print("Error: File is empty.")
        raise
    except pd.errors.ParserError:
        print("Error: File could not be parsed.")
        raise

def clean_text(text):
    text = text.lower()
    text = re.sub(r'\s+', ' ', text)
    text = re.sub(r'[^\w\s]', '', text)
    return ' '.join([word for word in text.split() if word not in stop_words])

def preprocess_data(df):
    # Print column names to check for mismatches
    print("Columns in the dataset:", df.columns.tolist())

    # Ensure correct column names
    if 'content' not in df.columns or 'sentiment' not in df.columns:
        raise KeyError("Required columns are missing from the dataset.")

    df['content'] = df['content'].fillna('')
    df['sentiment'] = df['sentiment'].fillna('neutral')  # Assuming 'neutral' is a
    df['sentiment'] = df['sentiment'].astype(str)
    df['content'] = df['content'].apply(clean_text)
    return df

def train_model(X_train, y_train):
    vectorizer = TfidfVectorizer(max_features=5000)
    X_train_vec = vectorizer.fit_transform(X_train).toarray()

    label_encoder = LabelEncoder()
    y_train_encoded = label_encoder.fit_transform(y_train)
    y_train_one_hot = to_categorical(y_train_encoded)

    model = Sequential()
    model.add(Dense(512, input_dim=X_train_vec.shape[1], activation='relu'))
    model.add(Dropout(0.5))
```

```
    model.add(Dense(256, activation='relu'))
    model.add(Dropout(0.5))
    model.add(Dense(y_train_one_hot.shape[1], activation='softmax'))

    model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accu

    # Adding EarlyStopping to avoid overfitting
    early_stopping = EarlyStopping(monitor='val_loss', patience=3, restore_best_wei

    model.fit(X_train_vec, y_train_one_hot, epochs=10, batch_size=32, validation_sp

    return model, vectorizer, label_encoder

def evaluate_model(model, vectorizer, label_encoder, X_test, y_test):
    X_test_vec = vectorizer.transform(X_test).toarray()

    y_test_encoded = label_encoder.transform(y_test)
    y_test_one_hot = to_categorical(y_test_encoded)

    y_pred_prob = model.predict(X_test_vec)
    y_pred = np.argmax(y_pred_prob, axis=1)

    accuracy = accuracy_score(np.argmax(y_test_one_hot, axis=1), y_pred)
    report = classification_report(np.argmax(y_test_one_hot, axis=1), y_pred, targe

    return accuracy, report

def predict_sentiment(model, vectorizer, label_encoder, text):
    text = clean_text(text)
    text_vec = vectorizer.transform([text]).toarray()
    text_pred_prob = model.predict(text_vec)
    text_pred = np.argmax(text_pred_prob, axis=1)
    sentiment = label_encoder.inverse_transform(text_pred)
    return sentiment[0]

def main(file_path):
    df = load_dataset(file_path)
    df = preprocess_data(df)

    X = df['content']
    y = df['sentiment']

    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random

    print("Training data shape:", X_train.shape)
    print("Testing data shape:", X_test.shape)

    model, vectorizer, label_encoder = train_model(X_train, y_train)
    accuracy, report = evaluate_model(model, vectorizer, label_encoder, X_test, y_t

    print(f"Accuracy: {accuracy}")
    print(f"Classification Report:\n{report}")

    print("\nLabel encoding mapping:")
    for index, label in enumerate(label_encoder.classes_):
        print(f"{index}: {label}")

    while True:
        user_input = input("Enter a message to analyze sentiment (or type 'exit' to
        if user_input.lower() == 'exit':
            break
        sentiment = predict_sentiment(model, vectorizer, label_encoder, user_input)
        print(f"The sentiment of the message is: {sentiment.capitalize()}")
```

```
if __name__ == "__main__":
    main('Sentiment_Analysis.csv')
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\jalum\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
Columns in the dataset: ['tweet_id', 'sentiment', 'author', 'content']
Training data shape: (32000,)
Testing data shape: (8000,)
C:\Users\jalum\anaconda3\Lib\site-packages\keras\src\layers\core\dense.py:87: User
Warning: Do not pass an `input_shape`/`input_dim` argument to a layer. When using
Sequential models, prefer using an `Input(shape)` object as the first layer in the
model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
Epoch 1/10
900/900 ━━━━━━━━━━━━━━━━━━━━ 21s 23ms/step - accuracy: 0.2573 - loss: 2.1228 - val
_accuracy: 0.3237 - val_loss: 1.9329
Epoch 2/10
900/900 ━━━━━━━━━━━━━━━━━━━━ 22s 24ms/step - accuracy: 0.4001 - loss: 1.7779 - val
_accuracy: 0.3338 - val_loss: 1.9293
Epoch 3/10
900/900 ━━━━━━━━━━━━━━━━━━━━ 21s 24ms/step - accuracy: 0.4897 - loss: 1.5351 - val
_accuracy: 0.3231 - val_loss: 2.0177
Epoch 4/10
900/900 ━━━━━━━━━━━━━━━━━━━━ 20s 23ms/step - accuracy: 0.5801 - loss: 1.2968 - val
_accuracy: 0.3091 - val_loss: 2.1738
Epoch 5/10
900/900 ━━━━━━━━━━━━━━━━━━━━ 20s 23ms/step - accuracy: 0.6674 - loss: 1.0293 - val
_accuracy: 0.3103 - val_loss: 2.3752
250/250 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step
C:\Users\jalum\anaconda3\Lib\site-packages\sklearn\metrics\_classification.py:146
9: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to
0.0 in labels with no predicted samples. Use `zero_division` parameter to control
this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\jalum\anaconda3\Lib\site-packages\sklearn\metrics\_classification.py:146
9: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to
0.0 in labels with no predicted samples. Use `zero_division` parameter to control
this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\jalum\anaconda3\Lib\site-packages\sklearn\metrics\_classification.py:146
9: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to
0.0 in labels with no predicted samples. Use `zero_division` parameter to control
this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

```
Accuracy: 0.33525
Classification Report:
              precision    recall  f1-score   support

       anger       0.00      0.00      0.00        19
     boredom       0.00      0.00      0.00        31
       empty       0.00      0.00      0.00       162
  enthusiasm       0.00      0.00      0.00       163
         fun       0.07      0.01      0.01       338
   happiness       0.30      0.47      0.36      1028
        hate       0.44      0.20      0.28       268
        love       0.44      0.39      0.42       762
     neutral       0.35      0.53      0.42      1740
      relief       0.00      0.00      0.00       352
     sadness       0.28      0.43      0.34      1046
    surprise       0.41      0.03      0.05       425
       worry       0.36      0.28      0.31      1666

    accuracy                           0.34      8000
   macro avg       0.20      0.18      0.17      8000
weighted avg       0.31      0.34      0.30      8000


Label encoding mapping:
0: anger
1: boredom
2: empty
3: enthusiasm
4: fun
5: happiness
6: hate
7: love
8: neutral
9: relief
10: sadness
11: surprise
12: worry
Enter a message to analyze sentiment (or type 'exit' to quit): i am having fever
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 37ms/step
The sentiment of the message is: Neutral
```

In [ ]: