```python
# Importing required libraries
import nltk
import pandas as pd
from nltk.corpus import stopwords
from textblob import Word
from sklearn.preprocessing import LabelEncoder
from collections import Counter
import wordcloud
from sklearn.metrics import classification_report, confusion_matrix,
accuracy_score
from keras.models import Sequential
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
from keras.layers import Dense, Embedding, LSTM, SpatialDropout1D
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as pit
import numpy as np

# Load the dataset from a CSV file into a pandas DataFrame
data = pd.read_csv('sentimentdataset.csv')

# Remove any rows with missing values (NaN) to ensure clean data
data = data.dropna()

# Check for remaining missing values after cleaning
# This prints the count of null values per column (should be 0 after
dropna())
data.isnull().sum()
```

```
Unnamed: 0.1    0
Unnamed: 0      0
Text            0
Sentiment       0
Timestamp       0
User            0
Platform        0
Hashtags        0
Retweets        0
Likes           0
Country         0
Year            0
Month           0
Day             0
Hour            0
dtype: int64
```

```python
# Display the first 5 rows of the DataFrame to inspect the dataset
structure
# This helps verify columns, sample data, and ensure proper loading
data.head()
```

```
   Unnamed: 0.1  Unnamed: 0  \
0             0           0
1             1           1
2             2           2
3             3           3
4             4           4

                                                Text    Sentiment  \
0   Enjoying a beautiful day at the park!        ...   Positive
1   Traffic was terrible this morning.           ...   Negative
2   Just finished an amazing workout! □          ...   Positive
3   Excited about the upcoming weekend getaway!  ...   Positive
4   Trying out a new recipe for dinner tonight.  ...   Neutral

              Timestamp            User     Platform  \
0  2023-01-15 12:30:00   User123          Twitter
1  2023-01-15 08:45:00   CommuterX        Twitter
2  2023-01-15 15:45:00   FitnessFan      Instagram
3  2023-01-15 18:20:00   AdventureX       Facebook
4  2023-01-15 19:55:00   ChefCook        Instagram

                                    Hashtags  Retweets  Likes
Country  \
0   #Nature #Park                                 15.0   30.0     USA

1   #Traffic #Morning                              5.0   10.0
Canada
2   #Fitness #Workout                             20.0   40.0   USA

3   #Travel #Adventure                             8.0   15.0     UK

4   #Cooking #Food                                12.0   25.0
Australia

   Year  Month  Day  Hour
0  2023      1   15    12
1  2023      1   15     8
2  2023      1   15    15
3  2023      1   15    18
4  2023      1   15    19
```

```python
# Drop unnecessary columns from the DataFrame to clean the dataset
# Removes columns named 'Unnamed: 0.1' and 'Unnamed: 0' (often auto-generated by pandas when saving/loading CSV files)
data =data = data.drop(['Unnamed: 0.1', 'Unnamed: 0'], axis=1)

# Display concise summary of the DataFrame structure:
# - Number of entries (rows)
# - Column names and data types
# - Memory usage
```

```
# - Non-null counts (helps identify missing data)
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 732 entries, 0 to 731
Data columns (total 13 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   Text       732 non-null    object
 1   Sentiment  732 non-null    object
 2   Timestamp  732 non-null    object
 3   User       732 non-null    object
 4   Platform   732 non-null    object
 5   Hashtags   732 non-null    object
 6   Retweets   732 non-null    float64
 7   Likes      732 non-null    float64
 8   Country    732 non-null    object
 9   Year       732 non-null    int64
 10  Month      732 non-null    int64
 11  Day        732 non-null    int64
 12  Hour       732 non-null    int64
dtypes: float64(2), int64(4), object(7)
memory usage: 74.5+ KB
```

```
# Display the first 5 rows of the DataFrame to:
# - Verify data loaded correctly
# - Inspect column values
# - Check for obvious data quality issues
data.head()
```

```
                                                Text    Sentiment  \
0   Enjoying a beautiful day at the park!        ...   Positive
1   Traffic was terrible this morning.           ...   Negative
2   Just finished an amazing workout! □          ...   Positive
3   Excited about the upcoming weekend getaway!  ...   Positive
4   Trying out a new recipe for dinner tonight.  ...    Neutral

             Timestamp            User     Platform  \
0  2023-01-15 12:30:00   User123          Twitter
1  2023-01-15 08:45:00   CommuterX        Twitter
2  2023-01-15 15:45:00   FitnessFan      Instagram
3  2023-01-15 18:20:00   AdventureX       Facebook
4  2023-01-15 19:55:00   ChefCook        Instagram

                                    Hashtags  Retweets  Likes
Country  \
0   #Nature #Park                                  15.0   30.0     USA

1   #Traffic #Morning                               5.0   10.0
Canada
```

```
2   #Fitness #Workout                                  20.0   40.0   USA

3   #Travel #Adventure                                  8.0   15.0     UK

4   #Cooking #Food                                     12.0   25.0
Australia

   Year  Month  Day  Hour
0  2023      1   15    12
1  2023      1   15     8
2  2023      1   15    15
3  2023      1   15    18
4  2023      1   15    19
```

```
import re

# Get all unique special characters in the column
special_chars = set(''.join(data['Text'].astype(str).apply(lambda x:
''.join(re.findall(r'[^a-zA-Z0-9\s]', x)))))

print(special_chars)
```

```
{'□', '+', ',', '□', '–', '.', '#', '□', '–', '□', "'", '!', '□', ':',
'é', '', '□', '-', '?', '❤'}
```

```
# Create a new DataFrame containing only the 'Text' and 'Sentiment'
columns
# This selects specific columns for sentiment analysis while
discarding others
# - 'Text': Column containing the textual data to analyze
# - 'Sentiment': Column containing the ground truth labels (e.g.,
positive/negative)
data_sentiment_analysis = data[['Text']]
data_sentiment_analysis
```

```
                                                  Text
0     Enjoying a beautiful day at the park!        ...
1     Traffic was terrible this morning.           ...
2     Just finished an amazing workout! □          ...
3     Excited about the upcoming weekend getaway!  ...
4     Trying out a new recipe for dinner tonight.  ...
..                                                 ...
727  Collaborating on a science project that receiv...
728  Attending a surprise birthday party organized ...
729  Successfully fundraising for a school charity ...
730  Participating in a multicultural festival, cel...
731  Organizing a virtual talent show during challe...

[732 rows x 1 columns]
```

```
import re
from textblob import Word

def clean(text):
    # Removes all special characters and numericals leaving the
alphabets and numbers
    text = re.sub(r'[^a-zA-Z0-9\s]', '', str(text))
    # Convert to lowercase
    text = text.lower()
    return text

data_sentiment_analysis['Clean_Text'] =
data_sentiment_analysis['Text'].apply(clean)
data_sentiment_analysis
```

```
C:\Users\patil\AppData\Local\Temp\ipykernel_17180\4087181782.py:1:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation:
https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#
returning-a-view-versus-a-copy
  data_sentiment_analysis['Clean_Text'] =
data_sentiment_analysis['Text'].apply(clean)
```

```
                                                  Text  \
0     Enjoying a beautiful day at the park!         ...
1     Traffic was terrible this morning.            ...
2     Just finished an amazing workout! 💪          ...
3     Excited about the upcoming weekend getaway!   ...
4     Trying out a new recipe for dinner tonight.   ...
..                                                  ...
727  Collaborating on a science project that receiv...
728  Attending a surprise birthday party organized ...
729  Successfully fundraising for a school charity ...
730  Participating in a multicultural festival, cel...
731  Organizing a virtual talent show during challe...

                                             Clean_Text
0     enjoying a beautiful day at the park         ...
1     traffic was terrible this morning            ...
2     just finished an amazing workout
3     excited about the upcoming weekend getaway   ...
4     trying out a new recipe for dinner tonight   ...
..                                                 ...
727  collaborating on a science project that receiv...
728  attending a surprise birthday party organized ...
729  successfully fundraising for a school charity ...
730  participating in a multicultural festival cele...
```

```
731  organizing a virtual talent show during challe...

[732 rows x 2 columns]
```

```
import nltk

"""This punkt tokenizer divides a text into a list of sentences by
using an unsupervised algorithm to build a model for abbreviation
words,
collocations, and words that start sentences. """

nltk.download('punkt')
from nltk.tokenize import word_tokenize
from nltk import pos_tag
nltk.download('stopwords')
from nltk.corpus import stopwords
nltk.download('wordnet')
from nltk.corpus import wordnet
```

```
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\patil\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\patil\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package wordnet to
[nltk_data]     C:\Users\patil\AppData\Roaming\nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
```

```
nltk.download('omw-1.4')
nltk.download('averaged_perceptron_tagger')

# POS tagger dictionary
pos_dict = {'J':wordnet.ADJ, 'V':wordnet.VERB, 'N':wordnet.NOUN,
'R':wordnet.ADV}
def token_stop_pos(text):
    tags = pos_tag(word_tokenize(text))
    #print(tags)
    newlist = []
    for word, tag in tags:
        if word.lower() not in set(stopwords.words('english')):
            newlist.append(tuple([word, pos_dict.get(tag[0])]))
          #print(tag[0])
          #print(pos_dict.get(tag[0]))
    return newlist

data_sentiment_analysis['POS tagged'] =
data_sentiment_analysis['Clean_Text'].apply(token_stop_pos)
```

```
[nltk_data] Downloading package omw-1.4 to
[nltk_data]     C:\Users\patil\AppData\Roaming\nltk_data...
```

```
[nltk_data]   Package omw-1.4 is already up-to-date!
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     C:\Users\patil\AppData\Roaming\nltk_data...
[nltk_data]   Package averaged_perceptron_tagger is already up-to-
[nltk_data]       date!
C:\Users\patil\AppData\Local\Temp\ipykernel_17180\2602511732.py:17:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation:
https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#
returning-a-view-versus-a-copy
  data_sentiment_analysis['POS tagged'] =
data_sentiment_analysis['Clean_Text'].apply(token_stop_pos)
```

```
data_sentiment_analysis
```

```
                                                  Text  \
0     Enjoying a beautiful day at the park!        ...
1     Traffic was terrible this morning.           ...
2     Just finished an amazing workout! 💪         ...
3     Excited about the upcoming weekend getaway!  ...
4     Trying out a new recipe for dinner tonight.  ...
..                                                 ...
727  Collaborating on a science project that receiv...
728  Attending a surprise birthday party organized ...
729  Successfully fundraising for a school charity ...
730  Participating in a multicultural festival, cel...
731  Organizing a virtual talent show during challe...

                                             Clean_Text  \
0     enjoying a beautiful day at the park         ...
1     traffic was terrible this morning            ...
2     just finished an amazing workout
3     excited about the upcoming weekend getaway   ...
4     trying out a new recipe for dinner tonight   ...
..                                                 ...
727  collaborating on a science project that receiv...
728  attending a surprise birthday party organized ...
729  successfully fundraising for a school charity ...
730  participating in a multicultural festival cele...
731  organizing a virtual talent show during challe...

                                             POS tagged
0    [(enjoying, v), (beautiful, a), (day, n), (par...
1          [(traffic, n), (terrible, a), (morning, n)]
2          [(finished, v), (amazing, a), (workout, n)]
3    [(excited, v), (upcoming, a), (weekend, n), (g...
4    [(trying, v), (new, a), (recipe, n), (dinner, ...
```

```
..                                                   ...
727  [(collaborating, v), (science, n), (project, n...
728  [(attending, v), (surprise, n), (birthday, n),...
729  [(successfully, r), (fundraising, v), (school,...
730  [(participating, v), (multicultural, a), (fest...
731  [(organizing, v), (virtual, a), (talent, n), (...

[732 rows x 3 columns]
```

```
# Obtaining the stem words – Lemmatization

from nltk.stem import WordNetLemmatizer
wordnet_lemmatizer = WordNetLemmatizer()
def lemmatize(pos_data):
    lemma_rew = " "
    for word, pos in pos_data:
        if not pos:
            lemma = word
            lemma_rew = lemma_rew + " " + lemma
        else:
            lemma = wordnet_lemmatizer.lemmatize(word, pos=pos)
            lemma_rew = lemma_rew + " " + lemma
    return lemma_rew

data_sentiment_analysis['Lemma'] = data_sentiment_analysis['POS
tagged'].apply(lemmatize)
data_sentiment_analysis.head()
```

```
C:\Users\patil\AppData\Local\Temp\ipykernel_17180\3136914373.py:16:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation:
https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#
returning-a-view-versus-a-copy
  data_sentiment_analysis['Lemma'] = data_sentiment_analysis['POS
tagged'].apply(lemmatize)
```

```
                                                Text  \
0   Enjoying a beautiful day at the park!        ...
1   Traffic was terrible this morning.           ...
2   Just finished an amazing workout! □          ...
3   Excited about the upcoming weekend getaway!  ...
4   Trying out a new recipe for dinner tonight.  ...

                                           Clean_Text  \
0   enjoying a beautiful day at the park         ...
1   traffic was terrible this morning            ...
2   just finished an amazing workout
```

```
3   excited about the upcoming weekend getaway   ...
4   trying out a new recipe for dinner tonight   ...

                                          POS tagged  \
0  [(enjoying, v), (beautiful, a), (day, n), (par...
1        [(traffic, n), (terrible, a), (morning, n)]
2        [(finished, v), (amazing, a), (workout, n)]
3  [(excited, v), (upcoming, a), (weekend, n), (g...
4  [(trying, v), (new, a), (recipe, n), (dinner, ...

                                Lemma
0           enjoy beautiful day park
1           traffic terrible morning
2             finish amazing workout
3    excite upcoming weekend getaway
4      try new recipe dinner tonight
```

```
!pip install vaderSentiment
```

```
Requirement already satisfied: vaderSentiment in c:\users\patil\
anaconda3\lib\site-packages (3.3.2)
Requirement already satisfied: requests in c:\users\patil\anaconda3\
lib\site-packages (from vaderSentiment) (2.32.3)
Requirement already satisfied: charset-normalizer<4,>=2 in c:\users\
patil\anaconda3\lib\site-packages (from requests->vaderSentiment)
(3.3.2)
Requirement already satisfied: idna<4,>=2.5 in c:\users\patil\
anaconda3\lib\site-packages (from requests->vaderSentiment) (2.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in c:\users\patil\
anaconda3\lib\site-packages (from requests->vaderSentiment) (2.2.3)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\patil\
anaconda3\lib\site-packages (from requests->vaderSentiment)
(2024.12.14)
```

```
from vaderSentiment.vaderSentiment import SentimentIntensityAnalyzer
analyzer = SentimentIntensityAnalyzer()


# function to calculate vader sentiment
def vadersentimentanalysis(review):
    vs = analyzer.polarity_scores(review)
    return vs['compound']

data_sentiment_analysis['Sentiment'] =
data_sentiment_analysis['Lemma'].apply(vadersentimentanalysis)

# function to analyse
def vader_analysis(compound):
    if compound >= 0.5:
        return 'Positive'
    elif compound < 0 :
```

```
        return 'Negative'
    else:
        return 'Neutral'
data_sentiment_analysis['Analysis'] =
data_sentiment_analysis['Sentiment'].apply(vader_analysis)
data_sentiment_analysis
```

```
C:\Users\patil\AppData\Local\Temp\ipykernel_17180\1890537781.py:10:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation:
https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#
returning-a-view-versus-a-copy
  data_sentiment_analysis['Sentiment'] =
data_sentiment_analysis['Lemma'].apply(vadersentimentanalysis)
C:\Users\patil\AppData\Local\Temp\ipykernel_17180\1890537781.py:20:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation:
https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#
returning-a-view-versus-a-copy
  data_sentiment_analysis['Analysis'] =
data_sentiment_analysis['Sentiment'].apply(vader_analysis)
```

```
                                                  Text  \
0     Enjoying a beautiful day at the park!        ...
1     Traffic was terrible this morning.           ...
2     Just finished an amazing workout! □          ...
3     Excited about the upcoming weekend getaway!  ...
4     Trying out a new recipe for dinner tonight.  ...
..                                                 ...
727  Collaborating on a science project that receiv...
728  Attending a surprise birthday party organized ...
729  Successfully fundraising for a school charity ...
730  Participating in a multicultural festival, cel...
731  Organizing a virtual talent show during challe...

                                             Clean_Text  \
0     enjoying a beautiful day at the park         ...
1     traffic was terrible this morning            ...
2     just finished an amazing workout
3     excited about the upcoming weekend getaway   ...
4     trying out a new recipe for dinner tonight   ...
..                                                 ...
727  collaborating on a science project that receiv...
728  attending a surprise birthday party organized ...
```

```
729  successfully fundraising for a school charity ...
730  participating in a multicultural festival cele...
731  organizing a virtual talent show during challe...

                                                POS tagged  \
0    [(enjoying, v), (beautiful, a), (day, n), (par...
1          [(traffic, n), (terrible, a), (morning, n)]
2          [(finished, v), (amazing, a), (workout, n)]
3    [(excited, v), (upcoming, a), (weekend, n), (g...
4    [(trying, v), (new, a), (recipe, n), (dinner, ...
..                                                 ...
727  [(collaborating, v), (science, n), (project, n...
728  [(attending, v), (surprise, n), (birthday, n),...
729  [(successfully, r), (fundraising, v), (school,...
730  [(participating, v), (multicultural, a), (fest...
731  [(organizing, v), (virtual, a), (talent, n), (...

                                                 Lemma  Sentiment
Analysis
0                              enjoy beautiful day park     0.7964
Positive
1                              traffic terrible morning    -0.4767
Negative
2                                finish amazing workout     0.5859
Positive
3                         excite upcoming weekend getaway     0.4767
Neutral
4                           try new recipe dinner tonight     0.0000
Neutral
..                                                   ...        ...
...
727    collaborate science project receive recognit...     0.7845
Positive
728    attend surprise birthday party organize frie...     0.9538
Positive
729    successfully fundraise school charity initia...     0.8689
Positive
730    participate multicultural festival celebrate...     0.8910
Positive
731    organize virtual talent show challenge time ...     0.6808
Positive

[732 rows x 6 columns]
```

```
vader_counts = data_sentiment_analysis['Analysis'].value_counts()
vader_counts
```

```
Analysis
Positive    304
Neutral     241
```

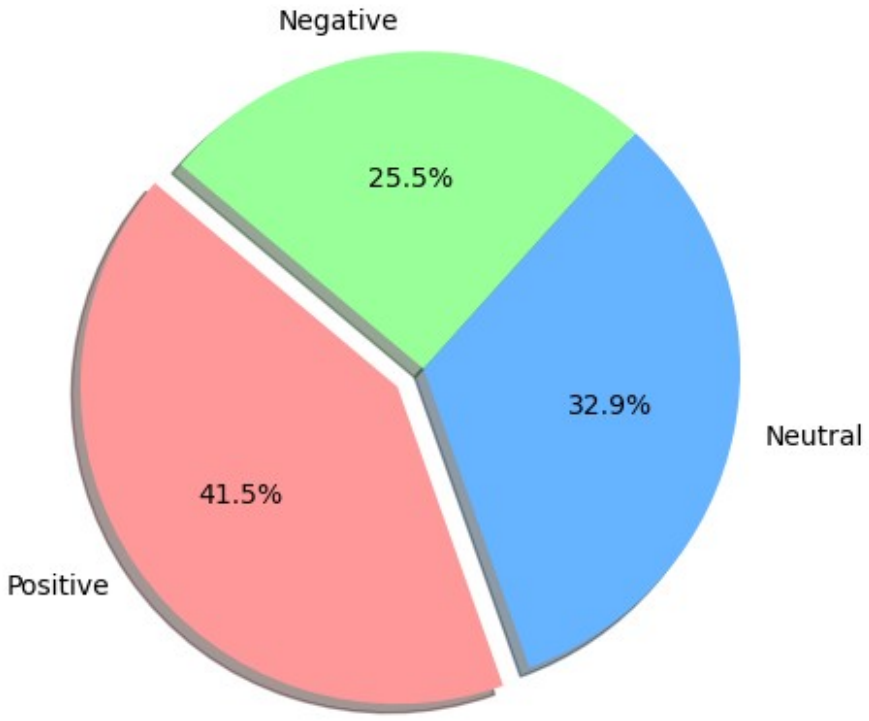
```
Negative    187
Name: count, dtype: int64
```

```
import matplotlib.pyplot as plt

# Data for the pie chart
labels = ['Positive', 'Neutral', 'Negative']
sizes = [304, 241, 187]
colors = ['#ff9999', '#66b3ff', '#99ff99']
explode = (0.1, 0, 0)  # Highlight the 'Positive' category

# Create the pie chart
plt.pie(sizes, explode=explode, labels=labels, colors=colors,
autopct='%1.1f%%', shadow=True, startangle=140)
plt.axis('equal')  # Ensures the pie chart is circular

# Display the pie chart
plt.show()
```




```
data_model_building = pd.DataFrame()  # If it's meant to be a
DataFrame
data_model_building['Words']=data_sentiment_analysis['Lemma']

data_model_building
```

```
                                                 Words
0                              enjoy beautiful day park
1                              traffic terrible morning
2                                finish amazing workout
3                        excite upcoming weekend getaway
4                          try new recipe dinner tonight
..                                                   ...
727    collaborate science project receive recognit...
728    attend surprise birthday party organize frie...
729    successfully fundraise school charity initia...
730    participate multicultural festival celebrate...
731    organize virtual talent show challenge time ...

[732 rows x 1 columns]
```

```
# Tokenizer configuration
tokenizer = Tokenizer(num_words=2500, oov_token="<OOV>")

# It will convert words into their corresponding unique integer index values,
#limited to the top 500 words specified earlier.
tokenizer.fit_on_texts(data_model_building['Words'].values)

# Convert texts to sequences
X = tokenizer.texts_to_sequences(data_model_building['Words'].values)
X=pad_sequences(X)
X
```

```
array([[   0,    0,    0, ...,  323,    4,  445],
       [   0,    0,    0, ..., 1017, 1018,  324],
       [   0,    0,    0, ...,  446, 1019,  240],
       ...,
       [   0,    0,    0, ..., 1015,  610,   71],
       [   0,    0,    0, ...,    6,  928,  934],
       [   0,    0,    0, ...,  384,  443,  165]])
```

```
print(f"Shape of X: {X.shape}")
# Should be (number_of_samples, sequence_length)
```

```
Shape of X: (732, 16)
```

```
print(f"Unique words: {len(tokenizer.word_index)}")
```

```
Unique words: 2149
```

```
# For binary sentiment (positive/negative):
from sklearn.preprocessing import LabelBinarizer

lb = LabelBinarizer()
y = lb.fit_transform(data_sentiment_analysis['Analysis'])  # Column
```

```
with 'positive'/'negative'

# For multi-class sentiment (positive/neutral/negative):
y = pd.get_dummies(data_sentiment_analysis['Analysis']).values

print(f"Shape of X: {y.shape}")
```

```
Shape of X: (732, 3)
```

```
X_train, X_test , y_train, y_test
=train_test_split(X,y,test_size=0.3,random_state=42)



import numpy as np

# Convert to arrays (ensure this happens AFTER train-test split)
X_train = np.asarray(X_train).astype('float32')  # Explicit float32
for GPU efficiency
X_test = np.asarray(X_test).astype('float32')
y_train = np.asarray(y_train)
y_test = np.asarray(y_test)

# Quality checks
print(f"X_train dtype: {X_train.dtype}, shape: {X_train.shape}")
print(f"y_train dtype: {y_train.dtype}, shape: {y_train.shape}")
```

```
X_train dtype: float32, shape: (512, 16)
y_train dtype: bool, shape: (512, 3)
```

```


print("Corrected shapes:")
print(f"X_train: {X_train.shape}")  # Should be (samples,
sequence_length)
print(f"y_train: {y_train.shape}")  # Should be (samples, 2) or
(samples, 3) etc.
```

```
Corrected shapes:
X_train: (512, 16)
y_train: (512, 3)
```

```


from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Embedding, SpatialDropout1D, LSTM,
Dense

# Updated model with adjustments based on your data characteristics
model = Sequential()

# 1. Embedding Layer (adjusted for vocabulary)
```

```python
model.add(Embedding(
    input_dim=2500,  # 2000 words + 1 for OOV
    output_dim=120,  # Embedding dimension (good starting point)
    input_length=15   # Matches your padded sequence length
))

# 2. Regularization
model.add(SpatialDropout1D(0.3))  # Slightly reduced from 0.4

# 3. LSTM Layer
model.add(LSTM(
    128,  # Reduced units for better generalization
    dropout=0.2,
    recurrent_dropout=0.2,
    return_sequences=False  # Only need last output for classification
))

# 4. Output Layer
model.add(Dense(y_train.shape[1], activation='softmax'))

# Compile with class weighting if imbalance exists
model.compile(
    loss='categorical_crossentropy',
    optimizer='adam',
    metrics=['accuracy']
)

print(model.summary())
```

```
C:\Users\patil\anaconda3\Lib\site-packages\keras\src\layers\core\
embedding.py:90: UserWarning: Argument `input_length` is deprecated.
Just remove it.
  warnings.warn(
```

```
Model: "sequential"
```

```
┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━
━━━━━┓
┃ Layer (type)                     ┃ Output Shape            ┃
Param # ┃
┡━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━
━━━━━┩
│ embedding (Embedding)            │ ?                       │   0
(unbuilt) │
├──────────────────────────────────┼─────────────────────────┼─────────
─────┤
│ spatial_dropout1d                │ ?                       │
0 │
│ (SpatialDropout1D)               │                         │
│
```

```
├─────────────────────────────────┼─────────────────────────┼──────────
───────┤
│ lstm (LSTM)                     │ ?                       │   0
(unbuilt) │
├─────────────────────────────────┼─────────────────────────┼──────────
───────┤
│ dense (Dense)                   │ ?                       │   0
(unbuilt) │
└─────────────────────────────────┴─────────────────────────┴──────────
───────┘

 Total params: 0 (0.00 B)

 Trainable params: 0 (0.00 B)

 Non-trainable params: 0 (0.00 B)

None
```

```
batch_size=32
history=model.fit(X_train, y_train, epochs =500,
batch_size=batch_size, verbose = 1)
```

```
Epoch 1/500
16/16 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step - accuracy: 0.3885 - loss:
1.0906
Epoch 2/500
16/16 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - accuracy: 0.4196 - loss:
1.0587
Epoch 3/500
16/16 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.4890 - loss:
0.9616
Epoch 4/500
16/16 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.7247 - loss:
0.6908
Epoch 5/500
16/16 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.9072 - loss:
0.4331
Epoch 6/500
16/16 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.9145 - loss:
0.2455
Epoch 7/500
16/16 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.9864 - loss:
0.1215
Epoch 8/500
16/16 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.9869 - loss:
0.0587
Epoch 9/500
16/16 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.9977 - loss:
0.0204
Epoch 10/500
```

```
model. evaluate (X_test, y_test )
```

```
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - accuracy: 0.6978 - loss: 1.6447
```

```
[1.8888267278671265, 0.6772727370262146]
```

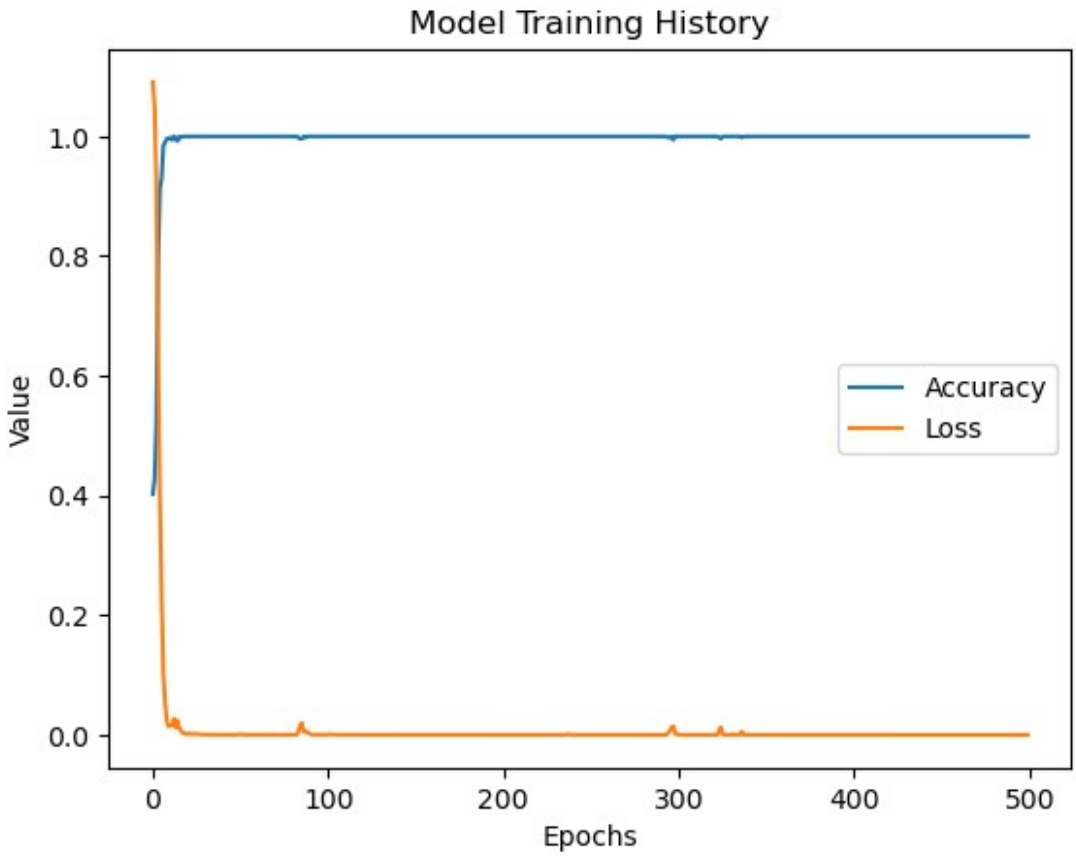
```
# Convert the training history to a DataFrame
history_dict = history.history
history_df = pd.DataFrame(history_dict)

# Display the DataFrame
print(history_df)
```

```
     accuracy      loss
0    0.402344  1.090665
1    0.427734  1.048668
2    0.519531  0.919519
3    0.792969  0.645935
4    0.912109  0.406009
..        ...       ...
495  1.000000  0.000002
496  1.000000  0.000002
497  1.000000  0.000002
498  1.000000  0.000004
499  1.000000  0.000002

[500 rows x 2 columns]
```

```
# Plotting accuracy and loss
plt.plot(history.history['accuracy'], label='Accuracy')
plt.plot(history.history['loss'], label='Loss')
plt.xlabel('Epochs')
plt.ylabel('Value')
plt.title('Model Training History')
plt.legend()
plt.show()
```


Model Training History
1.0
0.8
0.6
0.4
0.2
0.0
Value
0
100
200
300
400
500
Epochs
Accuracy
Loss