In [2]:
```
import pandas as pd
df = pd.read_csv("Twitter_Data.csv", encoding='latin1')
```

In [3]:
```
import pandas as pd
import numpy as np
import re
import string
from nltk.corpus import stopwords
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.preprocessing import LabelEncoder
```

In [1]:
```
import string
import nltk
nltk.download('stopwords')

def clean_text(text):
    if isinstance(text, str):
        text = text.lower()
        text = text.translate(str.maketrans('', '', string.punctuation))
        text = re.sub(r'\d+', '', text)
        text = ' '.join(text.split())
        stop_words = set(stopwords.words('english'))
        text = ' '.join(word for word in text.split() if word not in stop_words)
    else:
        text = ''
    return text
```

```
[nltk_data] Downloading package stopwords to C:\Users\MANICKA
[nltk_data]     MEENAKSHI.S\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

In [22]:
```
df['clean_text'].fillna('', inplace=True)
```

```
C:\Users\MANICKA MEENAKSHI.S\AppData\Local\Temp\ipykernel_34444\4131130163.py:1: Fut
ureWarning: A value is trying to be set on a copy of a DataFrame or Series through c
hained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work because
the intermediate object on which we are setting values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method
({col: value}, inplace=True)' or df[col] = df[col].method(value) instead, to perform
the operation inplace on the original object.


  df['clean_text'].fillna('', inplace=True)
```

In [23]:
```
df['selected'] = df['clean_text'].apply(clean_text)
```

In [4]:
```
import re
import string
from bs4 import BeautifulSoup
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
import contractions
```

```
import nltk
nltk.download('stopwords')
nltk.download('wordnet')
lemmatizer = WordNetLemmatizer()
stop_words = set(stopwords.words('english'))

def clean_text(text):
    if not isinstance(text, str):
        return ''
    text = text.lower()
    text = re.sub(r'http\S+|www\S+|https\S+', '', text, flags=re.MULTILINE)
    text = BeautifulSoup(text, "html.parser").text
    text = contractions.fix(text)
    text = re.sub(r'[^a-zA-Z\s]', '', text)
    text = ' '.join(text.split())
    text = ' '.join(word for word in text.split() if word not in stop_words)
    text = ' '.join(lemmatizer.lemmatize(word) for word in text.split())

    return text
```

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

In [5]:
```
df['selected'] = df['clean_text'].apply(clean_text)
```

```
C:\Users\MANICKA MEENAKSHI.S\AppData\Local\Temp\ipykernel_22180\4222066634.py:18: Ma
rkupResemblesLocatorWarning: The input looks more like a filename than markup. You m
ay want to open this file and pass the filehandle into Beautiful Soup.
  text = BeautifulSoup(text, "html.parser").text
```

In [6]:
```
def categorize_text(text):
    if not isinstance(text, str):
        return 'unknown'

    text = text.lower()

    threat_keywords = ['threat', 'harass', 'abuse', 'intimidate', 'coerce', 'bully'
    malpractice_keywords = ['unethical', 'negligence', 'malpractice', 'irresponsibl
    bullying_keywords = ['bully', 'tease', 'torment', 'harass', 'pick on']
    discrimination_keywords = ['discriminate', 'inequality', 'bias', 'prejudice', '
    anger_keywords = ['anger', 'rage', 'fury', 'irritation', 'hostility']
    vengeance_keywords = ['revenge', 'retaliate', 'retribution', 'vendetta', 'venge
    manipulation_keywords = ['manipulate', 'deceive', 'mislead', 'exploit', 'scheme
    happy_keywords = ['happy', 'joy', 'delighted', 'content', 'pleased', 'cheerful'
    neutral_keywords = ['safe', 'secure', 'protected', 'stable', 'reliable', 'peace


    if any(keyword in text for keyword in threat_keywords):
        return 'threat'
    elif any(keyword in text for keyword in malpractice_keywords):
        return 'malpractice'
    elif any(keyword in text for keyword in bullying_keywords):
```

```
        return 'bullying'
    elif any(keyword in text for keyword in discrimination_keywords):
        return 'discrimination'
    elif any(keyword in text for keyword in anger_keywords):
        return 'anger'
    elif any(keyword in text for keyword in vengeance_keywords):
        return 'vengeance'
    elif any(keyword in text for keyword in manipulation_keywords):
        return 'manipulation'
    elif any(keyword in text for keyword in happy_keywords):
        return 'happy'
    elif any(keyword in text for keyword in neutral_keywords):
        return 'neutral'
    else:
        return 'unknown'
df['label'] = df['clean_text'].apply(categorize_text)
print(df)
```

```
                                               clean_text  category  \
0       when modi promised â□□minimum government maxim...      -1.0
1       talk all the nonsense and continue all the dra...       0.0
2       what did just say vote for modi  welcome bjp t...       1.0
3       asking his supporters prefix chowkidar their n...       1.0
4       answer who among these the most powerful world...       1.0
...                                                   ...       ...
162975  why these 456 crores paid neerav modi not reco...      -1.0
162976  dear rss terrorist payal gawar what about modi...      -1.0
162977  did you cover her interaction forum where she ...       0.0
162978  there big project came into india modi dream p...       0.0
162979  have you ever listen about like gurukul where ...       1.0

                                                 selected    label
0       modi promised minimum government maximum gover...  unknown
1                   talk nonsense continue drama vote modi  unknown
2       say vote modi welcome bjp told rahul main camp...  unknown
3       asking supporter prefix chowkidar name modi gr...   threat
4       answer among powerful world leader today trump...  unknown
...                                                   ...      ...
162975  crore paid neerav modi recovered congress lead...  neutral
162976  dear r terrorist payal gawar modi killing plus...   threat
162977                       cover interaction forum left  unknown
162978  big project came india modi dream project happ...  unknown
162979  ever listen like gurukul discipline maintained...  unknown

[162980 rows x 4 columns]
```

In [27]:
```
df2 = df[df['label'] != 'unknown']


df2.reset_index(drop=True, inplace=True)


df2.to_csv('df_combined.csv', index=False)


#df2.to_pickle('df_without_unknown.pkl')
```

```
print(df2)
```

```
                                            clean_text  category  \
0      asking his supporters prefix chowkidar their n...       1.0
1      hope tuthukudi people would prefer honest well...       1.0
2      has already taken notice and ordered probe now...       0.0
3      farmersâ welfare about 474 farmers get secon...       0.0
4      last time you paid loan some entrepreneurs nam...       0.0
...                                                  ...       ...
15188  lalu convicted person eye never but culture di...      -1.0
15189  india todays antifake news war room has found ...       1.0
15190  playing with the insecurities the people with ...       1.0
15191  why these 456 crores paid neerav modi not reco...      -1.0
15192  dear rss terrorist payal gawar what about modi...      -1.0

                                            selected           label
0      asking supporter prefix chowkidar name modi gr...          threat
1      hope tuthukudi people would prefer honest well...           anger
2      already taken notice ordered probe time modi t...          threat
3      farmer welfare farmer get second installment n...    manipulation
4      last time paid loan entrepreneur named vijay m...         neutral
...                                                  ...             ...
15188  lalu convicted person eye never culture differ...  discrimination
15189  india today antifake news war room found viral...    manipulation
15190  playing insecurity people credible opposition ...         neutral
15191  crore paid neerav modi recovered congress lead...         neutral
15192  dear r terrorist payal gawar modi killing plus...          threat

[15193 rows x 4 columns]
```

In [28]:
```
import pandas as pd
data = pd.read_csv("df_combined.csv", encoding='latin1')
```

In [31]:
```
df = df.reset_index(drop=True)
X = X[:len(df)]
y = df['label'].values

print(f"Shape of X after reindexing: {X.shape}")
print(f"Length of y after reindexing: {len(y)}")
```

```
Shape of X after reindexing: (15193, 43)
Length of y after reindexing: 15193
```

In [32]:
```
df['label'] = df['label'].fillna(label_mapping['unknown'])
df = df.dropna(subset=['label'])

X = X[df.index]
y = df['label'].values
```

In [41]:
```
import numpy as np
import pandas as pd
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Embedding, GlobalAveragePooling1D
```

```
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
label_mapping = {
    'threat': 0,
    'malpractice': 1,
    'neutral': 2,
    'bullying': 3,
    'discrimination': 4,
    'anger': 5,
    'vengeance': 6,
    'manipulation': 7,
    'unknown': 8,
    'happpy':9

}

df['label'] = df['clean_text'].apply(categorize_text)
df['label'] = df['label'].map(label_mapping)
df = df[df['label'] != 8]

tokenizer = Tokenizer(num_words=1000)
tokenizer.fit_on_texts(df['clean_text'])
sequences = tokenizer.texts_to_sequences(df['clean_text'])
X = pad_sequences(sequences)
y = df['label'].values

input_length = 42

model = Sequential()
model.add(Embedding(input_dim=1000, output_dim=64, input_length=input_length))
model.add(GlobalAveragePooling1D())
model.add(Dense(16, activation='relu'))
model.add(Dense(len(label_mapping), activation='softmax'))


model.compile(loss='sparse_categorical_crossentropy', optimizer='adam', metrics=['a

model.compile(loss='sparse_categorical_crossentropy', optimizer='adam', metrics=['a
model.fit(X, y, epochs=49, batch_size=32, validation_split=0.2)


new_texts = [
    "I can't believe they did that, I'll get my terrorist gang done",
    "let make a good plan to kill our prime minister . tell prasad to be ready",
    "I am peaceful now"

]

new_sequences = tokenizer.texts_to_sequences(new_texts)
new_X = pad_sequences(new_sequences, maxlen=X.shape[1])
predictions = model.predict(new_X)
predicted_labels = np.argmax(predictions, axis=1)

inv_label_mapping = {v: k for k, v in label_mapping.items()}
predicted_categories = [inv_label_mapping[label] for label in predicted_labels]
```

```
print(predicted_categories)
```

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

```
Epoch 1/49
318/318 ━━━━━━━━━━━━━━━━━━━━ 4s 4ms/step - accuracy: 0.3006 - loss: 1.7840 - val_acc
uracy: 0.1937 - val_loss: 1.6511
Epoch 2/49
318/318 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.5693 - loss: 1.3672 - val_acc
uracy: 0.7063 - val_loss: 1.1623
Epoch 3/49
318/318 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.7722 - loss: 0.8971 - val_acc
uracy: 0.8012 - val_loss: 0.7983
Epoch 4/49
318/318 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.8263 - loss: 0.6184 - val_acc
uracy: 0.8157 - val_loss: 0.6423
Epoch 5/49
318/318 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.8415 - loss: 0.5040 - val_acc
uracy: 0.8390 - val_loss: 0.5715
Epoch 6/49
318/318 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.8759 - loss: 0.4238 - val_acc
uracy: 0.8406 - val_loss: 0.5178
Epoch 7/49
318/318 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.8876 - loss: 0.3799 - val_acc
uracy: 0.8484 - val_loss: 0.5005
Epoch 8/49
318/318 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.8886 - loss: 0.3651 - val_acc
uracy: 0.8528 - val_loss: 0.4751
Epoch 9/49
318/318 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.9089 - loss: 0.3097 - val_acc
uracy: 0.8606 - val_loss: 0.4521
Epoch 10/49
318/318 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.9161 - loss: 0.2886 - val_acc
uracy: 0.8575 - val_loss: 0.4545
Epoch 11/49
318/318 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.9149 - loss: 0.2758 - val_acc
uracy: 0.8654 - val_loss: 0.4468
Epoch 12/49
318/318 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.9163 - loss: 0.2685 - val_acc
uracy: 0.8587 - val_loss: 0.4591
Epoch 13/49
318/318 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.9219 - loss: 0.2556 - val_acc
uracy: 0.8622 - val_loss: 0.4439
Epoch 14/49
318/318 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.9227 - loss: 0.2390 - val_acc
uracy: 0.8622 - val_loss: 0.4572
Epoch 15/49
318/318 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.9317 - loss: 0.2226 - val_acc
uracy: 0.8638 - val_loss: 0.4418
Epoch 16/49
318/318 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - accuracy: 0.9353 - loss: 0.2013 - val_acc
uracy: 0.8646 - val_loss: 0.4477
Epoch 17/49
318/318 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.9323 - loss: 0.2129 - val_acc
uracy: 0.8669 - val_loss: 0.4541
Epoch 18/49
318/318 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.9371 - loss: 0.2072 - val_acc
uracy: 0.8669 - val_loss: 0.4814
Epoch 19/49
318/318 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.9383 - loss: 0.2034 - val_acc
```

```
uracy: 0.8724 - val_loss: 0.4656
Epoch 20/49
318/318 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.9483 - loss: 0.1820 - val_acc
uracy: 0.8677 - val_loss: 0.4668
Epoch 21/49
318/318 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.9505 - loss: 0.1697 - val_acc
uracy: 0.8677 - val_loss: 0.4782
Epoch 22/49
318/318 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.9529 - loss: 0.1642 - val_acc
uracy: 0.8724 - val_loss: 0.5079
Epoch 23/49
318/318 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.9516 - loss: 0.1662 - val_acc
uracy: 0.8681 - val_loss: 0.5154
Epoch 24/49
318/318 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.9550 - loss: 0.1500 - val_acc
uracy: 0.8736 - val_loss: 0.4838
Epoch 25/49
318/318 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.9632 - loss: 0.1378 - val_acc
uracy: 0.8720 - val_loss: 0.5161
Epoch 26/49
318/318 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.9611 - loss: 0.1312 - val_acc
uracy: 0.8713 - val_loss: 0.5018
Epoch 27/49
318/318 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.9627 - loss: 0.1322 - val_acc
uracy: 0.8732 - val_loss: 0.5352
Epoch 28/49
318/318 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.9674 - loss: 0.1181 - val_acc
uracy: 0.8650 - val_loss: 0.6227
Epoch 29/49
318/318 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.9645 - loss: 0.1260 - val_acc
uracy: 0.8736 - val_loss: 0.5409
Epoch 30/49
318/318 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.9694 - loss: 0.1174 - val_acc
uracy: 0.8756 - val_loss: 0.5437
Epoch 31/49
318/318 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.9723 - loss: 0.1035 - val_acc
uracy: 0.8705 - val_loss: 0.5718
Epoch 32/49
318/318 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.9691 - loss: 0.1127 - val_acc
uracy: 0.8681 - val_loss: 0.6337
Epoch 33/49
318/318 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.9743 - loss: 0.0939 - val_acc
uracy: 0.8693 - val_loss: 0.5763
Epoch 34/49
318/318 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.9730 - loss: 0.0986 - val_acc
uracy: 0.8665 - val_loss: 0.6295
Epoch 35/49
318/318 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.9739 - loss: 0.0975 - val_acc
uracy: 0.8677 - val_loss: 0.6252
Epoch 36/49
318/318 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.9768 - loss: 0.0894 - val_acc
uracy: 0.8744 - val_loss: 0.6220
Epoch 37/49
318/318 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.9777 - loss: 0.0881 - val_acc
uracy: 0.8693 - val_loss: 0.6200
Epoch 38/49
```

```
318/318 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.9801 - loss: 0.0779 - val_acc
uracy: 0.8594 - val_loss: 0.7249
Epoch 39/49
318/318 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.9828 - loss: 0.0732 - val_acc
uracy: 0.8709 - val_loss: 0.6822
Epoch 40/49
318/318 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.9801 - loss: 0.0769 - val_acc
uracy: 0.8669 - val_loss: 0.6900
Epoch 41/49
318/318 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.9805 - loss: 0.0737 - val_acc
uracy: 0.8634 - val_loss: 0.7406
Epoch 42/49
318/318 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.9806 - loss: 0.0720 - val_acc
uracy: 0.8638 - val_loss: 0.7367
Epoch 43/49
318/318 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.9844 - loss: 0.0649 - val_acc
uracy: 0.8705 - val_loss: 0.7204
Epoch 44/49
318/318 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.9859 - loss: 0.0597 - val_acc
uracy: 0.8634 - val_loss: 0.7688
Epoch 45/49
318/318 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.9843 - loss: 0.0581 - val_acc
uracy: 0.8610 - val_loss: 0.8081
Epoch 46/49
318/318 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.9874 - loss: 0.0540 - val_acc
uracy: 0.8689 - val_loss: 0.7654
Epoch 47/49
318/318 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.9849 - loss: 0.0558 - val_acc
uracy: 0.8646 - val_loss: 0.7881
Epoch 48/49
318/318 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.9884 - loss: 0.0514 - val_acc
uracy: 0.8681 - val_loss: 0.7790
Epoch 49/49
318/318 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.9861 - loss: 0.0483 - val_acc
uracy: 0.8681 - val_loss: 0.7929
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 120ms/step
['threat', 'anger', 'neutral']
```

In [ ]:

In [ ]:

In [32]:
```
print(df['label'].isna().sum())
```

```
2495
```

In [34]:
```
df = df.dropna(subset=['label'])


df['label'].fillna(label_mapping['neutral'], inplace=True)
```

```
C:\Users\MANICKA MEENAKSHI.S\AppData\Local\Temp\ipykernel_34444\2149398451.py:5: Fut
ureWarning: A value is trying to be set on a copy of a DataFrame or Series through c
hained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work because
the intermediate object on which we are setting values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method
({col: value}, inplace=True)' or df[col] = df[col].method(value) instead, to perform
the operation inplace on the original object.


  df['label'].fillna(label_mapping['neutral'], inplace=True)
```

In [37]:
```
print(df['label'].unique())
```

```
[0. 5. 7. 2. 6. 4. 3. 1.]
```

In [38]:
```
import numpy as np

# Check for labels outside the valid range
invalid_labels = np.where(y >= len(label_mapping))[0]

print(f"Indices of invalid labels: {invalid_labels}")
print(f"Invalid label values: {y[invalid_labels]}")
```

```
Indices of invalid labels: []
Invalid label values: []
```

In [39]:
```
# Map label '10' to a valid label, for example, '9'
y = np.where(y == 10, 9, y)
```

In [51]:
```
model.save('Sentiment_model.h5')
model.save('Sentiment_model.keras')
```

```
WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.
saving.save_model(model)`. This file format is considered legacy. We recommend using
instead the native Keras format, e.g. `model.save('my_model.keras')` or `keras.savin
g.save_model(model, 'my_model.keras')`.
C:\Users\MANICKA MEENAKSHI.S\anaconda3\Lib\site-packages\keras\src\saving\saving_ap
i.py:102: UserWarning: You are saving a model that has not yet been built. It might
not contain any weights yet. Consider building the model first by calling it on some
data.
  return saving_lib.save_model(model, filepath)
```

In [ ]:
```

```