```
# Importing all the required libraries:
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
import sklearn
import re
import string
import nltk
from nltk.stem import WordNetLemmatizer
from nltk.stem import LancasterStemmer
from sklearn import model_selection, preprocessing, linear_model, naive_bayes, metrics
from sklearn.feature_extraction.text import TfidfVectorizer, CountVectorizer,HashingVectorizer
import tensorflow as tf
import pathlib
import shutil
import tempfile
```

```
# !pip install tensorflow-gpu
```

```
!pip install -q git+https://github.com/tensorflow/docs
import tensorflow_hub as hub
import tensorflow_docs as tfdocs
import tensorflow_docs.modeling
import tensorflow_docs.plots

print("Version: ", tf.__version__)
print("Hub version: ", hub.__version__)
print("GPU is", "available" if tf.config.list_physical_devices('GPU') else "NOT AVAILABLE")

logdir = pathlib.Path(tempfile.mkdtemp())/"tensorboard_logs"
shutil.rmtree(logdir, ignore_errors=True)
```

```
  Building wheel for tensorflow-docs (setup.py) ... done
Version:  2.9.2
Hub version:  0.12.0
GPU is available
```

```
!pip install tensorflow_hub
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: tensorflow_hub in /usr/local/lib/python3.7/dist-packages (0.12.0)
Requirement already satisfied: numpy>=1.12.0 in /usr/local/lib/python3.7/dist-packages (from tensorflow_hub) (1.21.6)
Requirement already satisfied: protobuf>=3.8.0 in /usr/local/lib/python3.7/dist-packages (from tensorflow_hub) (3.17.3)
Requirement already satisfied: six>=1.9 in /usr/local/lib/python3.7/dist-packages (from protobuf>=3.8.0->tensorflow_hub) (1.15.0)
```

```
!pip install keras-tuner --upgrade
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Collecting keras-tuner
  Downloading keras_tuner-1.1.3-py3-none-any.whl (135 kB)
     |████████████████████████████████| 135 kB 33.9 MB/s
Requirement already satisfied: numpy in /usr/local/lib/python3.7/dist-packages (from keras-tuner) (1.21.6)
Requirement already satisfied: ipython in /usr/local/lib/python3.7/dist-packages (from keras-tuner) (7.9.0)
Requirement already satisfied: tensorboard in /usr/local/lib/python3.7/dist-packages (from keras-tuner) (2.9.1)
Requirement already satisfied: packaging in /usr/local/lib/python3.7/dist-packages (from keras-tuner) (21.3)
Collecting kt-legacy
  Downloading kt_legacy-1.0.4-py3-none-any.whl (9.6 kB)
Requirement already satisfied: requests in /usr/local/lib/python3.7/dist-packages (from keras-tuner) (2.23.0)
Requirement already satisfied: pygments in /usr/local/lib/python3.7/dist-packages (from ipython->keras-tuner) (2.6.1)
Requirement already satisfied: traitlets>=4.2 in /usr/local/lib/python3.7/dist-packages (from ipython->keras-tuner) (5.1.1)
Collecting jedi>=0.10
  Downloading jedi-0.18.1-py2.py3-none-any.whl (1.6 MB)
     |████████████████████████████████| 1.6 MB 46.2 MB/s
Requirement already satisfied: setuptools>=18.5 in /usr/local/lib/python3.7/dist-packages (from ipython->keras-tuner) (57.4.0)
Requirement already satisfied: pickleshare in /usr/local/lib/python3.7/dist-packages (from ipython->keras-tuner) (0.7.5)
Requirement already satisfied: backcall in /usr/local/lib/python3.7/dist-packages (from ipython->keras-tuner) (0.2.0)
Requirement already satisfied: decorator in /usr/local/lib/python3.7/dist-packages (from ipython->keras-tuner) (4.4.2)
Requirement already satisfied: prompt-toolkit<2.1.0,>=2.0.0 in /usr/local/lib/python3.7/dist-packages (from ipython->keras-tuner) (
Requirement already satisfied: pexpect in /usr/local/lib/python3.7/dist-packages (from ipython->keras-tuner) (4.8.0)
Requirement already satisfied: parso<0.9.0,>=0.8.0 in /usr/local/lib/python3.7/dist-packages (from jedi>=0.10->ipython->keras-tuner
Requirement already satisfied: six>=1.9.0 in /usr/local/lib/python3.7/dist-packages (from prompt-toolkit<2.1.0,>=2.0.0->ipython->ke
Requirement already satisfied: wcwidth in /usr/local/lib/python3.7/dist-packages (from prompt-toolkit<2.1.0,>=2.0.0->ipython->keras
Requirement already satisfied: pyparsing!=3.0.5,>=2.0.2 in /usr/local/lib/python3.7/dist-packages (from packaging->keras-tuner) (3.
Requirement already satisfied: ptyprocess>=0.5 in /usr/local/lib/python3.7/dist-packages (from pexpect->ipython->keras-tuner) (0.7.
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.7/dist-packages (from requests->keras-tuner) (2.10)
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.7/dist-packages (from requests->keras-tuner) (3.0.4)
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in /usr/local/lib/python3.7/dist-packages (from requests->ke
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.7/dist-packages (from requests->keras-tuner) (2022.9.24
Requirement already satisfied: absl-py>=0.4 in /usr/local/lib/python3.7/dist-packages (from tensorboard->keras-tuner) (1.3.0)
Requirement already satisfied: google-auth<3,>=1.6.3 in /usr/local/lib/python3.7/dist-packages (from tensorboard->keras-tuner) (1.3
```

```
Requirement already satisfied: tensorboard-data-server<0.7.0,>=0.6.0 in /usr/local/lib/python3.7/dist-packages (from tensorboard->k
Requirement already satisfied: markdown>=2.6.8 in /usr/local/lib/python3.7/dist-packages (from tensorboard->keras-tuner) (3.4.1)
Requirement already satisfied: tensorboard-plugin-wit>=1.6.0 in /usr/local/lib/python3.7/dist-packages (from tensorboard->keras-tun
Requirement already satisfied: grpcio>=1.24.3 in /usr/local/lib/python3.7/dist-packages (from tensorboard->keras-tuner) (1.50.0)
Requirement already satisfied: protobuf<3.20,>=3.9.2 in /usr/local/lib/python3.7/dist-packages (from tensorboard->keras-tuner) (3.1
Requirement already satisfied: werkzeug>=1.0.1 in /usr/local/lib/python3.7/dist-packages (from tensorboard->keras-tuner) (1.0.1)
Requirement already satisfied: wheel>=0.26 in /usr/local/lib/python3.7/dist-packages (from tensorboard->keras-tuner) (0.38.1)
Requirement already satisfied: google-auth-oauthlib<0.5,>=0.4.1 in /usr/local/lib/python3.7/dist-packages (from tensorboard->keras-
Requirement already satisfied: cachetools<5.0,>=2.0.0 in /usr/local/lib/python3.7/dist-packages (from google-auth<3,>=1.6.3->tensor
Requirement already satisfied: rsa<5,>=3.1.4 in /usr/local/lib/python3.7/dist-packages (from google-auth<3,>=1.6.3->tensorboard->ke
Requirement already satisfied: pyasn1-modules>=0.2.1 in /usr/local/lib/python3.7/dist-packages (from google-auth<3,>=1.6.3->tensorb
Requirement already satisfied: requests-oauthlib>=0.7.0 in /usr/local/lib/python3.7/dist-packages (from google-auth-oauthlib<0.5,>=
Requirement already satisfied: importlib-metadata>=4.4 in /usr/local/lib/python3.7/dist-packages (from markdown>=2.6.8->tensorboard
Requirement already satisfied: typing-extensions>=3.6.4 in /usr/local/lib/python3.7/dist-packages (from importlib-metadata>=4.4->ma
Requirement already satisfied: zipp>=0.5 in /usr/local/lib/python3.7/dist-packages (from importlib-metadata>=4.4->markdown>=2.6.8->
Requirement already satisfied: pyasn1<0.5.0,>=0.4.6 in /usr/local/lib/python3.7/dist-packages (from pyasn1-modules>=0.2.1->google-a
Requirement already satisfied: oauthlib>=3.0.0 in /usr/local/lib/python3.7/dist-packages (from requests-oauthlib>=0.7.0->google-aut
Installing collected packages: jedi, kt-legacy, keras-tuner
Successfully installed jedi-0.18.1 keras-tuner-1.1.3 kt-legacy-1.0.4
```

```
1 !nvidia-smi
```

```
Wed Nov  9 13:11:17 2022
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 460.32.03    Driver Version: 460.32.03    CUDA Version: 11.2     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|                               |                      |               MIG M. |
|===============================+======================+======================|
|   0  Tesla T4            Off  | 00000000:00:04.0 Off |                    0 |
| N/A   41C    P8     9W /  70W |      3MiB / 15109MiB |      0%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
|  No running processes found                                                 |
+-----------------------------------------------------------------------------+
```

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

```
[nltk_data] Downloading package omw-1.4 to /root/nltk_data...
True
```

```
1 from google.colab import drive
2 drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
1 # Importing the data set:
2
3 dataset = pd.read_csv("/content/drive/MyDrive/emotion-emotion_69k.csv")
4 dataset.head()
```

	Unnamed: 0	Situation	emotion	empathetic_dialogues	labels	Unnamed: 5	Unnamed: 6
0	0	I remember going to the fireworks with my best...	sentimental	Customer :I remember going to see the firework...	Was this a friend you were in love with, or ju...	NaN	NaN
1	1	I remember going to the fireworks with my best...	sentimental	Customer :This was a best friend. I miss her.\...	Where has she gone?	NaN	NaN
2	2	I remember going to the fireworks with my best...	sentimental	Customer :We no longer talk.\nAgent :	Oh was this something that happened because of...	NaN	NaN

```
1 dataset['emotion'].value_counts()
```

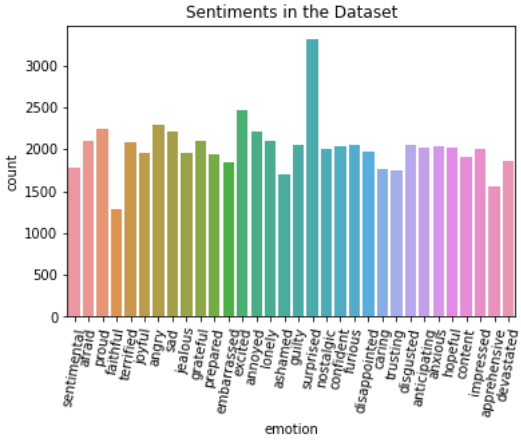
```
surprised       3306
excited         2468
angry           2296
proud           2247
sad             2216
annoyed         2213
lonely          2106
afraid          2094
grateful        2091
terrified       2078
guilty          2053
```

```
furious         2048
disgusted       2044
confident       2041
anxious         2037
anticipating    2026
hopeful         2019
impressed       2009
nostalgic       1996
disappointed    1969
jealous         1955
joyful          1953
prepared        1937
content         1903
devastated      1856
embarrassed     1844
sentimental     1773
caring          1765
trusting        1755
ashamed         1698
apprehensive    1552
faithful        1283
Name: emotion, dtype: int64
```

Initial Exploratory Data Analysis:

```
plt.title("Sentiments in the Dataset")
sns.countplot("emotion", data = dataset)
plt.xticks(rotation = 80)
plt.show()
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass the following variable as a keyword arg: x. F
  FutureWarning
```



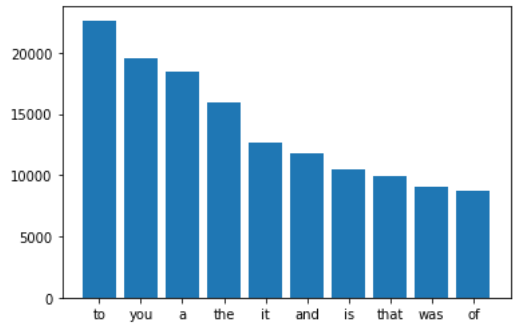
Exploratory Data Analysis:

```
from nltk.corpus import stopwords # Also, for the data preprocessing
nltk.download('stopwords')
nltk.download('wordnet')
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
[nltk_data] Downloading package wordnet to /root/nltk_data...
True
```

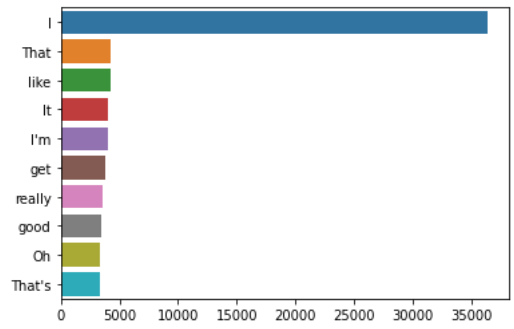
```
# Firstly we create a corpus:
corpus = []
def plot_top_stopwords_barchart(text):
    stop = set(stopwords.words('english'))
    new = text.str.split()
    new = new.values.tolist()
    corpus = [word for i in new for word in i]

    from collections import defaultdict
    dedict = defaultdict(int)
    for word in corpus:
        if word in stop:
            dedict[word]+=1
    top = sorted(dedict.items(), key = lambda x:x[1], reverse = True)[:10]
    x,y = zip(*top)
    plt.bar(x,y)

```

```
18
19 plot_top_stopwords_barchart(dataset['labels'])
```



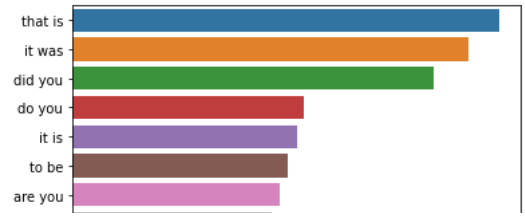
Here we can see that the word "**to**" is used many times and hence it is the most used single stop-word in the dataset.

```
1 # Before going into preprocessing, we should check the most frequent words:
2 # We will check those words which aren't stopwords and are frequent too.
3 from collections import  Counter
4 def plot_top_non_stopwords_barchart(text):
5     stop=set(stopwords.words('english'))
6
7     new= text.str.split()
8     new=new.values.tolist()
9     corpus=[word for i in new for word in i]
10
11     counter=Counter(corpus)
12     most=counter.most_common()
13     x, y=[], []
14     for word,count in most[:40]:
15         if (word not in stop):
16             x.append(word)
17             y.append(count)
18
19     sns.barplot(x=y,y=x)
20
21 plot_top_non_stopwords_barchart(dataset['labels'])
```



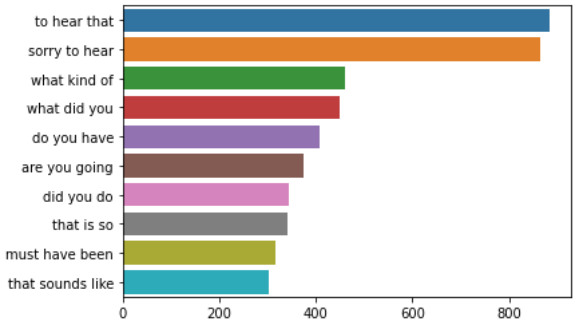
The most used word is none other than "**I**"". We definately know this!

```
1 from sklearn.feature_extraction.text import CountVectorizer
2 from collections import  Counter
3 def get_top_ngram(corpus, n=None):
4     vec = CountVectorizer(ngram_range=(n, n)).fit(corpus)
5     bag_of_words = vec.transform(corpus)
6     sum_words = bag_of_words.sum(axis=0)
7     words_freq = [(word, sum_words[0, idx])
8                   for word, idx in vec.vocabulary_.items()]
9     words_freq =sorted(words_freq, key = lambda x: x[1], reverse=True)
10     return words_freq[:10]
11
12 top_n_bigrams=get_top_ngram(dataset['labels'],2)[:10]
13
14 x,y=map(list,zip(*top_n_bigrams))
15 sns.barplot(x=y,y=x)
16 plt.show()
```



This graph shows that, "**that is**" is the most frequently used bigrams.

```
top_tri_grams=get_top_ngram(dataset['labels'],n=3)
x,y=map(list,zip(*top_tri_grams))
sns.barplot(x=y,y=x)
plt.show()
```



The most used tri-grams is "**to hear that**".

Data Preproceessing:

```
dataset.drop(dataset.columns[dataset.columns.str.contains('unnamed',case = False)],axis = 1, inplace = True)
```

```
dataset.drop(['empathetic_dialogues', 'labels'], axis = 1, inplace = True)
```

```
dataset.isnull().sum()
```

```
Situation    0
emotion      5
dtype: int64
```

```
dataset.dropna(inplace = True)
```

```
dataset.isnull().sum()
```

```
Situation    0
emotion      0
dtype: int64
```

```
dataset['Length'] = dataset['Situation'].apply(len)
```

```
dataset
```

	Situation	emotion	Length
0	I remember going to the fireworks with my best...	sentimental	120

```
dataset.isna().sum()
```

```
Situation    0
emotion      0
Length       0
dtype: int64
```

```
dataset['emotion'].value_counts()
```

```
surprised       3306
excited         2468
angry           2296
proud           2247
sad             2216
annoyed         2213
lonely          2106
afraid          2094
grateful        2091
terrified       2078
guilty          2053
furious         2048
disgusted       2044
confident       2041
anxious         2037
anticipating    2026
hopeful         2019
impressed       2009
nostalgic       1996
disappointed    1969
jealous         1955
joyful          1953
prepared        1937
content         1903
devastated      1856
embarrassed     1844
sentimental     1773
caring          1765
trusting        1755
ashamed         1698
apprehensive    1552
faithful        1283
Name: emotion, dtype: int64
```

```
df1 = dataset.copy()
```

```
df1 =df1.drop_duplicates()
```

```
df1
```

	Situation	emotion	Length
0	I remember going to the fireworks with my best...	sentimental	120
5	i used to scare for darkness	afraid	29
10	I showed a guy how to run a good bead in weldi...	proud	78
14	I have always been loyal to my wife.	faithful	36
17	A recent job interview that I had made me feel...	terrified	103
...	...	...	...
64620	I was watching professional rodeo the other da...	impressed	135
64623	I am waiting to see if I pass my graduate exam...	anticipating	73
64626	My house burned down and I had to rescue my fa...	afraid	74
64629	I found some pictures of my grandma in the att...	sentimental	59
64633	I woke up this morning to my wife telling me s...	surprised	62

19306 rows × 3 columns

```
df1['emotion'] = df1['emotion'].astype('category')
```

```
df1['Sentiment'] = pd.factorize(df1['emotion'])[0] + 1
```

```
print(df1['emotion'].unique())
```

```
['sentimental', 'afraid', 'proud', 'faithful', 'terrified', ..., 'hopeful', 'content', 'impressed', 'apprehensive', 'devastated']
Length: 32
Categories (32, object): ['afraid', 'angry', 'annoyed', 'anticipating', ..., 'sentimental',
                          'surprised', 'terrified', 'trusting']
```

```
df1.dtypes
```

```
Situation      object
emotion      category
Length          int64
Sentiment       int64
dtype: object
```

```
df1['Sentiment'].value_counts()
```

```
18    1000
13     741
7      686
3      670
8      664
14     662
10     638
15     633
2      626
30     617
5      617
28     616
25     614
17     613
27     612
20     611
26     599
21     599
6      598
19     597
22     595
11     589
9      579
29     570
12     559
32     557
1      515
23     504
24     499
16     490
31     463
4      373
Name: Sentiment, dtype: int64
```

```
df2 = df1.copy()
```

```
df2.drop_duplicates(subset = "Situation",keep='first', inplace = True)
```

```
df2.shape
```

```
(19207, 4)
```

```
print("The original shape: ",df1.shape)
df1.drop_duplicates(inplace=True)
print("After dropiing duplicates:",df2.shape)
```

```
The original shape:  (19306, 4)
After dropiing duplicates: (19207, 4)
```

```
df2.head()
```

	Situation	emotion	Length	Sentiment
0	I remember going to the fireworks with my best...	sentimental	120	1
5	i used to scare for darkness	afraid	29	2
10	I showed a guy how to run a good bead in weldi...	proud	78	3
14	I have always been loyal to my wife.	faithful	36	4
17	A recent job interview that I had made me feel...	terrified	103	5

```
df2
```

	Situation	emotion	Length	Sentiment
0	I remember going to the fireworks with my best...	sentimental	120	1
5	i used to scare for darkness	afraid	29	2
10	I showed a guy how to run a good bead in weldi...	proud	78	3
14	I have always been loyal to my wife.	faithful	36	4
17	A recent job interview that I had made me feel...	terrified	103	5
...	...	...	...	...
64620	I was watching professional rodeo the other da...	impressed	135	30
64623	I am waiting to see if I pass my graduate exam...	anticipating	73	26
64626	My house burned down and I had to rescue my fa...	afraid	74	2
64629	I found some pictures of my grandma in the att...	sentimental	59	1
64633	I woke up this morning to my wife telling me s...	surprised	62	18

19207 rows × 4 columns

```
# Get Independent Features:
#X = df2.drop(['labels','empathetic_dialogues'], axis = 1)
X = df2['Situation']
# Get Dependent Features:
y = pd.get_dummies(df2['emotion']).values
```

```
print(X.shape)
print("----"*3)
print(y.shape)
```

```
(19207,)
------------
(19207, 32)
```

```
# One hot Encoding:
Messages = X.copy()
Messages = Messages.reset_index()
```

```
from nltk.corpus import stopwords
nltk.download('stopwords')
nltk.download('wordnet')
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
True
```

```
ls = LancasterStemmer()
wordNet = WordNetLemmatizer()
```

```
corpus = []
for i in range(len(Messages)):
    record = re.sub("[^a-zA-Z]"," ",Messages['Situation'][i])
    record = record.lower()
    record = record.split()
    record = [wordNet.lemmatize(word) for word in record if not word in stopwords.words('english')]
    record = ' '.join(record)
    corpus.append(record)
```

Deep Learning Techniques:

```
from tensorflow.keras.layers import Embedding
from tensorflow.keras.layers import SpatialDropout1D
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.models import Sequential
from tensorflow.keras.preprocessing.text import one_hot
from tensorflow.keras.layers import LSTM
from tensorflow.keras.layers import GRU
from tensorflow.keras.layers import Bidirectional
from tensorflow.keras.layers import Dense
from tensorflow.compat.v1.keras.layers import CuDNNLSTM, CuDNNGRU
from tensorflow.keras.layers import Dropout
```

```
# Vocabulary Size:
voc_size = 5000
```

```
onehot_repr = [one_hot(words, voc_size) for words in corpus]
#onehot_repr
```

```
## Embedding Representation:
sent_length = 20
embedded_docs = pad_sequences(onehot_repr, padding = 'pre', maxlen = sent_length)
print(embedded_docs)
```

```
[[   0    0    0 ... 1247 3083 3007]
 [   0    0    0 ... 2406 4308 2865]
 [   0    0    0 ... 1612  242  484]
 ...
 [   0    0    0 ... 3945 3665 1096]
 [   0    0    0 ... 4070  839 1096]
 [   0    0    0 ... 4937 2771 3760]]
```

```
embedded_docs.shape
```

```
(19207, 20)
```

```
import numpy as np
X_final = np.array(embedded_docs)
y_final = np.array(y)
```

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X_final,y_final, test_size = 0.20, random_state=42 )
```

```
print("The train x shape: ", X_train.shape)
print("The train y shape: ", y_train.shape)
print("The test x shape: ", X_test.shape)
print("The test y shape: ", y_test.shape)
```

```
The train x shape:  (15365, 20)
The train y shape:  (15365, 32)
The test x shape:  (3842, 20)
The test y shape:  (3842, 32)
```

```
embedding_vector_features = 80
def train_and_evaluate_model(model_name, trainable=False):
    if(model_name == 'LSTM'):
        #hub_layer = hub.kerasLayer(model_name, dtype=tf.string,  trainable=trainable)
        model = Sequential()
        model.add(Embedding(voc_size,embedding_vector_features, input_length = sent_length))
        model.add(Dropout(0.5))
        model.add(CuDNNLSTM(200, return_sequences = True)) # 400 Neurons
        model.add(CuDNNLSTM(200))
        model.add(Dropout(0.5))
        model.add(Dense(32, activation = 'softmax', kernel_initializer = 'glorot_uniform'))
        model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=0.0001),
                  loss=tf.losses.CategoricalCrossentropy(),
                  metrics = ['accuracy','Precision', 'Recall','AUC'])

        model.summary()
        history = model.fit(X_train, y_train, validation_data = (X_test, y_test), epochs = 50, batch_size = 64)
        y_pred_proba.append(np.argmax(model.predict(X_test), axis = -1))
        # print(np.argmax(model.predict(X_test), axis = -1))

    elif(model_name == 'Bi-LSTM'):
        #hub_layer = hub.kerasLayer(model_name, dtype=tf.string,  trainable=trainable)
        model = Sequential()
        model.add(Embedding(voc_size,embedding_vector_features, input_length = sent_length))
        model.add(Dropout(0.5))
        model.add(Bidirectional(CuDNNLSTM(200, return_sequences=True))) # 400 Neurons
        model.add(Bidirectional(CuDNNLSTM(200))) # 400 Neurons
        model.add(Dropout(0.5))
        model.add(Dense(32, activation = 'softmax', kernel_initializer = 'glorot_uniform'))
        model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=0.0001),
                  loss=tf.losses.CategoricalCrossentropy(),
                  metrics = ['accuracy','Precision', 'Recall','AUC'])
        model.summary()
        history = model.fit(X_train, y_train, validation_data = (X_test, y_test), epochs = 50, batch_size = 64)
        #y_pred_proba.append(model.predict(X_test) / 100)
        np.argmax(model.predict(X_test), axis = -1)

    elif(model_name == 'GRU'):
```

```
        #hub_layer = hub.kerasLayer(model_name, dtype=tf.string,  trainable=trainable)
        model = Sequential()
        model.add(Embedding(voc_size,embedding_vector_features, input_length = sent_length))
        model.add(Dropout(0.5))
        model.add((CuDNNGRU(200, return_sequences = True))) # 400 Neurons
        model.add((CuDNNGRU(200))) # 400 Neurons
        model.add(Dropout(0.5))
        model.add(Dense(32, activation = 'softmax', kernel_initializer = 'glorot_uniform'))
        model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=0.0001),
                  loss=tf.losses.CategoricalCrossentropy(),
                  metrics = ['accuracy','Precision', 'Recall','AUC'])
        model.summary()
        history = model.fit(X_train, y_train, validation_data = (X_test, y_test), epochs = 50, batch_size = 64)
        #y_pred_proba.append(model.predict(X_test) / 100)
        np.argmax(model.predict(X_test), axis = -1)

    elif(model_name == 'Bi-GRU'):
        #hub_layer = hub.kerasLayer(model_name, dtype=tf.string,  trainable=trainable)
        model = Sequential()
        model.add(Embedding(voc_size,embedding_vector_features, input_length = sent_length))
        model.add(Dropout(0.5))
        model.add(Bidirectional((CuDNNGRU(400, return_sequences=True)))) # 400 Neurons
        model.add(Bidirectional((CuDNNGRU(400)))) # 400 Neurons
        model.add(Dropout(0.5))
        model.add(Dense(32, activation = 'softmax', kernel_initializer = 'glorot_uniform'))
        model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=0.0001),
                  loss=tf.losses.CategoricalCrossentropy(),
                  metrics = ['accuracy','Precision', 'Recall','AUC'])
        model.summary()
        history = model.fit(X_train, y_train, validation_data = (X_test, y_test), epochs = 50, batch_size = 64)
        #y_pred_proba.append(model.predict(X_test) / 100)
        np.argmax(model.predict(X_test), axis = -1)
    return history, y_pred_proba
```

```
# We should be cognizant to save the history objects.
# For storing  the y_predict_probas for all the methods:
histories = {}
y_pred_proba = []
```

```
# model_name = input("Please enter the model name: ")
model_name = 'LSTM'
histories[model_name], y_pred_proba = train_and_evaluate_model(model_name, trainable=False)
histories


# model_name = input("Please enter the model name: ")
# model = train_and_evaluate_model(model_name, trainable=False)
# model
# LSTM
# Bi-LSTM
# GRU
# Bi-GRU

```

```
Model: "sequential_18"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_18 (Embedding)    (None, 20, 80)            400000

 dropout_36 (Dropout)        (None, 20, 80)            0

 cu_dnnlstm_36 (CuDNNLSTM)   (None, 20, 200)           225600

 cu_dnnlstm_37 (CuDNNLSTM)   (None, 200)               321600

 dropout_37 (Dropout)        (None, 200)               0

 dense_18 (Dense)            (None, 32)                6432

=================================================================
Total params: 953,632
Trainable params: 953,632
Non-trainable params: 0
_________________________________________________________________
Epoch 1/50
241/241 [==============================] - 6s 15ms/step - loss: 3.4582 - accuracy: 0.0469 - precision: 0.0000e+00 - recall: 0.00
Epoch 2/50
241/241 [==============================] - 3s 11ms/step - loss: 3.4505 - accuracy: 0.0510 - precision: 0.0000e+00 - recall: 0.00
Epoch 3/50
241/241 [==============================] - 3s 11ms/step - loss: 3.4415 - accuracy: 0.0523 - precision: 0.0000e+00 - recall: 0.00
Epoch 4/50
241/241 [==============================] - 3s 11ms/step - loss: 3.3864 - accuracy: 0.0677 - precision: 0.0000e+00 - recall: 0.00
```

```
Epoch 5/50
241/241 [==============================] - 3s 11ms/step - loss: 3.1092 - accuracy: 0.1276 - precision: 0.5540 - recall: 0.0050 -
Epoch 6/50
241/241 [==============================] - 3s 11ms/step - loss: 2.8928 - accuracy: 0.1736 - precision: 0.5994 - recall: 0.0131 -
Epoch 7/50
241/241 [==============================] - 3s 11ms/step - loss: 2.7401 - accuracy: 0.2065 - precision: 0.5952 - recall: 0.0224 -
Epoch 8/50
241/241 [==============================] - 3s 11ms/step - loss: 2.6089 - accuracy: 0.2429 - precision: 0.6086 - recall: 0.0348 -
Epoch 9/50
241/241 [==============================] - 3s 11ms/step - loss: 2.5253 - accuracy: 0.2669 - precision: 0.6131 - recall: 0.0445 -
Epoch 10/50
241/241 [==============================] - 3s 11ms/step - loss: 2.4246 - accuracy: 0.2907 - precision: 0.6183 - recall: 0.0568 -
Epoch 11/50
241/241 [==============================] - 3s 12ms/step - loss: 2.3565 - accuracy: 0.3082 - precision: 0.6363 - recall: 0.0719 -
Epoch 12/50
241/241 [==============================] - 3s 12ms/step - loss: 2.2945 - accuracy: 0.3246 - precision: 0.6435 - recall: 0.0870 -
Epoch 13/50
241/241 [==============================] - 3s 11ms/step - loss: 2.2372 - accuracy: 0.3396 - precision: 0.6501 - recall: 0.1021 -
Epoch 14/50
241/241 [==============================] - 3s 11ms/step - loss: 2.1903 - accuracy: 0.3566 - precision: 0.6495 - recall: 0.1084 -
Epoch 15/50
241/241 [==============================] - 4s 15ms/step - loss: 2.1382 - accuracy: 0.3636 - precision: 0.6502 - recall: 0.1230 -
Epoch 16/50
241/241 [==============================] - 3s 11ms/step - loss: 2.0882 - accuracy: 0.3843 - precision: 0.6476 - recall: 0.1409 -
Epoch 17/50
241/241 [==============================] - 3s 11ms/step - loss: 2.0519 - accuracy: 0.3941 - precision: 0.6665 - recall: 0.1503 -
Epoch 18/50
```

```
# print(y_pred_proba[0].shape)
# y_test.shape
# matrix = metrics.confusion_matrix(y_test, y_pred_proba[0])
# # print(matrix)
```

```
model_name = 'Bi-LSTM'
histories[model_name], y_pred_proba = train_and_evaluate_model(model_name, trainable=False)
# histories
```

```
Model: "sequential_19"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_19 (Embedding)    (None, 20, 80)            400000

 dropout_38 (Dropout)        (None, 20, 80)            0

 bidirectional (Bidirectiona  (None, 20, 400)          451200
 l)

 bidirectional_1 (Bidirectio  (None, 400)              963200
 nal)

 dropout_39 (Dropout)        (None, 400)               0

 dense_19 (Dense)            (None, 32)                12832

=================================================================
Total params: 1,827,232
Trainable params: 1,827,232
Non-trainable params: 0
_________________________________________________________________
Epoch 1/50
241/241 [==============================] - 8s 20ms/step - loss: 3.4577 - accuracy: 0.0473 - precision: 0.0000e+00 - recall: 0.00
Epoch 2/50
241/241 [==============================] - 4s 15ms/step - loss: 3.4405 - accuracy: 0.0538 - precision: 0.0000e+00 - recall: 0.00
Epoch 3/50
241/241 [==============================] - 4s 15ms/step - loss: 3.2771 - accuracy: 0.0875 - precision: 0.0000e+00 - recall: 0.00
Epoch 4/50
241/241 [==============================] - 4s 16ms/step - loss: 3.0562 - accuracy: 0.1202 - precision: 0.5789 - recall: 0.0021 -
Epoch 5/50
241/241 [==============================] - 4s 15ms/step - loss: 2.9365 - accuracy: 0.1397 - precision: 0.5171 - recall: 0.0079 -
Epoch 6/50
241/241 [==============================] - 4s 15ms/step - loss: 2.8457 - accuracy: 0.1609 - precision: 0.5318 - recall: 0.0136 -
Epoch 7/50
241/241 [==============================] - 4s 16ms/step - loss: 2.7650 - accuracy: 0.1811 - precision: 0.5450 - recall: 0.0201 -
Epoch 8/50
241/241 [==============================] - 4s 15ms/step - loss: 2.6851 - accuracy: 0.2031 - precision: 0.5562 - recall: 0.0251 -
Epoch 9/50
241/241 [==============================] - 4s 16ms/step - loss: 2.6184 - accuracy: 0.2162 - precision: 0.5645 - recall: 0.0305 -
Epoch 10/50
241/241 [==============================] - 4s 15ms/step - loss: 2.5337 - accuracy: 0.2413 - precision: 0.5946 - recall: 0.0417 -
Epoch 11/50
241/241 [==============================] - 4s 15ms/step - loss: 2.4737 - accuracy: 0.2502 - precision: 0.6229 - recall: 0.0549 -
Epoch 12/50
241/241 [==============================] - 4s 15ms/step - loss: 2.4036 - accuracy: 0.2679 - precision: 0.5986 - recall: 0.0614 -
Epoch 13/50
241/241 [==============================] - 4s 15ms/step - loss: 2.3522 - accuracy: 0.2844 - precision: 0.6002 - recall: 0.0717 -
Epoch 14/50
```

```
241/241 [==============================] - 4s 15ms/step - loss: 2.3051 - accuracy: 0.2963 - precision: 0.6131 - recall: 0.0836 -
Epoch 15/50
241/241 [==============================] - 4s 15ms/step - loss: 2.2593 - accuracy: 0.3093 - precision: 0.5946 - recall: 0.0943 -
Epoch 16/50
241/241 [==============================] - 4s 15ms/step - loss: 2.2190 - accuracy: 0.3273 - precision: 0.6201 - recall: 0.1042 -
Epoch 17/50
241/241 [==============================] - 4s 16ms/step - loss: 2.1666 - accuracy: 0.3373 - precision: 0.6157 - recall: 0.1153 -
```

```
model_name = 'GRU'
histories[model_name], y_pred_proba = train_and_evaluate_model(model_name, trainable=False)
# histories
```

```
Model: "sequential_20"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_20 (Embedding)    (None, 20, 80)            400000

 dropout_40 (Dropout)        (None, 20, 80)            0

 cu_dnngru (CuDNNGRU)        (None, 20, 200)           169200

 cu_dnngru_1 (CuDNNGRU)      (None, 200)               241200

 dropout_41 (Dropout)        (None, 200)               0

 dense_20 (Dense)            (None, 32)                6432

=================================================================
Total params: 816,832
Trainable params: 816,832
Non-trainable params: 0
_________________________________________________________________
Epoch 1/50
241/241 [==============================] - 5s 13ms/step - loss: 3.4607 - accuracy: 0.0456 - precision: 0.0000e+00 - recall: 0.00
Epoch 2/50
241/241 [==============================] - 3s 11ms/step - loss: 3.4458 - accuracy: 0.0510 - precision: 0.0000e+00 - recall: 0.00
Epoch 3/50
241/241 [==============================] - 3s 11ms/step - loss: 3.4079 - accuracy: 0.0672 - precision: 0.0000e+00 - recall: 0.00
Epoch 4/50
241/241 [==============================] - 3s 11ms/step - loss: 3.1618 - accuracy: 0.1270 - precision: 0.6163 - recall: 0.0034 -
Epoch 5/50
241/241 [==============================] - 3s 11ms/step - loss: 2.8953 - accuracy: 0.1742 - precision: 0.5333 - recall: 0.0109 -
Epoch 6/50
241/241 [==============================] - 3s 11ms/step - loss: 2.7542 - accuracy: 0.2031 - precision: 0.5904 - recall: 0.0208 -
Epoch 7/50
241/241 [==============================] - 3s 13ms/step - loss: 2.6340 - accuracy: 0.2312 - precision: 0.6114 - recall: 0.0307 -
Epoch 8/50
241/241 [==============================] - 5s 19ms/step - loss: 2.5330 - accuracy: 0.2501 - precision: 0.6053 - recall: 0.0413 -
Epoch 9/50
241/241 [==============================] - 4s 17ms/step - loss: 2.4521 - accuracy: 0.2698 - precision: 0.6185 - recall: 0.0530 -
Epoch 10/50
241/241 [==============================] - 5s 19ms/step - loss: 2.3849 - accuracy: 0.2894 - precision: 0.6181 - recall: 0.0679 -
Epoch 11/50
241/241 [==============================] - 5s 20ms/step - loss: 2.3282 - accuracy: 0.2991 - precision: 0.6367 - recall: 0.0760 -
Epoch 12/50
241/241 [==============================] - 4s 18ms/step - loss: 2.2821 - accuracy: 0.3107 - precision: 0.6186 - recall: 0.0847 -
Epoch 13/50
241/241 [==============================] - 5s 19ms/step - loss: 2.2432 - accuracy: 0.3216 - precision: 0.6135 - recall: 0.0890 -
Epoch 14/50
241/241 [==============================] - 3s 14ms/step - loss: 2.2013 - accuracy: 0.3384 - precision: 0.6349 - recall: 0.1003 -
Epoch 15/50
241/241 [==============================] - 3s 11ms/step - loss: 2.1706 - accuracy: 0.3421 - precision: 0.6419 - recall: 0.1106 -
Epoch 16/50
241/241 [==============================] - 3s 11ms/step - loss: 2.1290 - accuracy: 0.3574 - precision: 0.6374 - recall: 0.1190 -
Epoch 17/50
241/241 [==============================] - 3s 11ms/step - loss: 2.1066 - accuracy: 0.3587 - precision: 0.6515 - recall: 0.1233 -
Epoch 18/50
241/241 [==============================] - 3s 11ms/step - loss: 2.0747 - accuracy: 0.3716 - precision: 0.6602 - recall: 0.1361 -
```

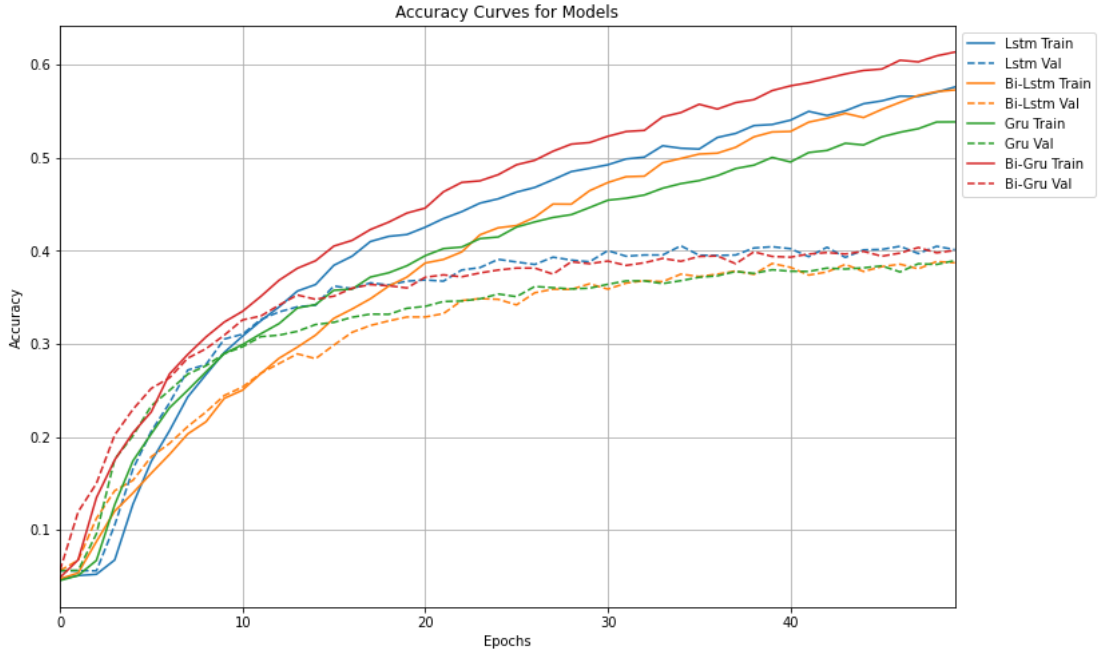
```
model_name = 'Bi-GRU'
histories[model_name], y_pred_proba = train_and_evaluate_model(model_name, trainable=False)
# histories
```

```
Model: "sequential_21"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_21 (Embedding)    (None, 20, 80)            400000

 dropout_42 (Dropout)        (None, 20, 80)            0

 bidirectional_2 (Bidirectio  (None, 20, 800)          1156800
 nal)

 bidirectional_3 (Bidirectio  (None, 800)              2884800
```

```
nal)

dropout_43 (Dropout)        (None, 800)               0

dense_21 (Dense)            (None, 32)                25632

=================================================================
Total params: 4,467,232
Trainable params: 4,467,232
Non-trainable params: 0
_________________________________________________________________
Epoch 1/50
241/241 [==============================] - 9s 25ms/step - loss: 3.4541 - accuracy: 0.0487 - precision: 0.0000e+00 - recall: 0.00
Epoch 2/50
241/241 [==============================] - 5s 21ms/step - loss: 3.3910 - accuracy: 0.0679 - precision: 0.0000e+00 - recall: 0.00
Epoch 3/50
241/241 [==============================] - 5s 23ms/step - loss: 3.0236 - accuracy: 0.1347 - precision: 0.6000 - recall: 0.0053 -
Epoch 4/50
241/241 [==============================] - 5s 22ms/step - loss: 2.8261 - accuracy: 0.1751 - precision: 0.6071 - recall: 0.0155 -
Epoch 5/50
241/241 [==============================] - 5s 21ms/step - loss: 2.6882 - accuracy: 0.2044 - precision: 0.5811 - recall: 0.0196 -
Epoch 6/50
241/241 [==============================] - 5s 21ms/step - loss: 2.5751 - accuracy: 0.2267 - precision: 0.6224 - recall: 0.0336 -
Epoch 7/50
241/241 [==============================] - 5s 21ms/step - loss: 2.4679 - accuracy: 0.2675 - precision: 0.6449 - recall: 0.0507 -
Epoch 8/50
241/241 [==============================] - 5s 21ms/step - loss: 2.3824 - accuracy: 0.2885 - precision: 0.6515 - recall: 0.0709 -
Epoch 9/50
241/241 [==============================] - 5s 22ms/step - loss: 2.3095 - accuracy: 0.3073 - precision: 0.6451 - recall: 0.0825 -
Epoch 10/50
241/241 [==============================] - 5s 21ms/step - loss: 2.2366 - accuracy: 0.3233 - precision: 0.6478 - recall: 0.0990 -
Epoch 11/50
241/241 [==============================] - 5s 21ms/step - loss: 2.1874 - accuracy: 0.3349 - precision: 0.6417 - recall: 0.1103 -
Epoch 12/50
241/241 [==============================] - 5s 21ms/step - loss: 2.1238 - accuracy: 0.3509 - precision: 0.6572 - recall: 0.1261 -
Epoch 13/50
241/241 [==============================] - 5s 21ms/step - loss: 2.0781 - accuracy: 0.3682 - precision: 0.6617 - recall: 0.1398 -
Epoch 14/50
241/241 [==============================] - 5s 21ms/step - loss: 2.0393 - accuracy: 0.3811 - precision: 0.6626 - recall: 0.1531 -
Epoch 15/50
241/241 [==============================] - 5s 21ms/step - loss: 1.9960 - accuracy: 0.3892 - precision: 0.6689 - recall: 0.1619 -
Epoch 16/50
241/241 [==============================] - 5s 21ms/step - loss: 1.9505 - accuracy: 0.4049 - precision: 0.6754 - recall: 0.1823 -
Epoch 17/50
```

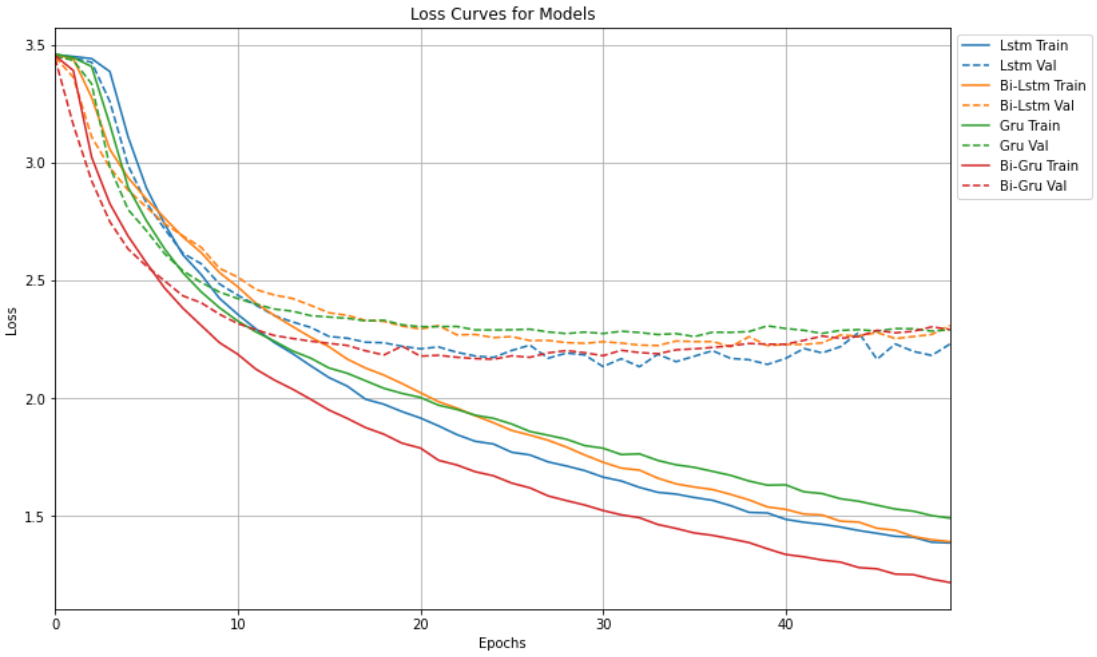
Performance Metrics and Analysis:

```
plt.rcParams['figure.figsize'] = (12, 8)
plotter = tfdocs.plots.HistoryPlotter(metric = 'accuracy')
plotter.plot(histories)
plt.xlabel("Epochs")
plt.legend(bbox_to_anchor=(1.0, 1.0), loc='upper left')
plt.title("Accuracy Curves for Models")
plt.show()
```



Loss Plot using tensorflow Hub:

```
plotter = tfdocs.plots.HistoryPlotter(metric = 'loss')
plotter.plot(histories)
plt.xlabel("Epochs")
plt.legend(bbox_to_anchor=(1.0, 1.0), loc='upper left')
plt.title("Loss Curves for Models")
plt.show()
```



ROC AUC Score and Curve:

```
print("LSTM")
print("Accuracy : ",max(histories['LSTM'].history['accuracy']))
print("Precision : ",max(histories['LSTM'].history['precision']))
print("Recall : ",max(histories['LSTM'].history['recall']))
print("AUC : ",max(histories['LSTM'].history['auc']))
```

```
LSTM
Accuracy :  0.5758542418479919
Precision :  0.7603234052658081
Recall :  0.3978522717952728
AUC :  0.9594595432281494
```

```
print("Bi-LSTM")
print("Accuracy : ",max(histories['Bi-LSTM'].history['accuracy']))
print("Precision : ",max(histories['Bi-LSTM'].history['precision']))
print("Recall : ",max(histories['Bi-LSTM'].history['recall']))
print("AUC : ",max(histories['Bi-LSTM'].history['auc']))
```

```
Bi-LSTM
Accuracy :  0.5726651549339294
Precision :  0.7380564212799072
Recall :  0.423104465007782
AUC :  0.9574123024940491
```

```
print("GRU")
print("Accuracy : ",max(histories['GRU'].history['accuracy']))
print("Precision : ",max(histories['GRU'].history['precision']))
print("Recall : ",max(histories['GRU'].history['recall']))
print("AUC : ",max(histories['GRU'].history['auc']))
```

```
GRU
Accuracy :  0.538366436958313
Precision :  0.7325630784034729
Recall :  0.35509273409843445
AUC :  0.9527528882026672
```

```
print("Bi-GRU")
print("Accuracy : ",max(histories['Bi-GRU'].history['accuracy']))
print("Precision : ",max(histories['Bi-GRU'].history['precision']))
```

```
print("Recall : ",max(histories['Bi-GRU'].history['recall']))
print("AUC : ",max(histories['Bi-GRU'].history['auc']))
```

```
Bi-GRU
Accuracy :  0.6134070754051208
Precision :  0.7608110904693604
Recall :  0.4786202311515808
AUC :  0.96768718957901
```

```
# print(y_pred_proba)
```

Tensorflow Board:

```
# %load_ext tensorboard

# %tensorboard --logdir {logdir}
```

TensorBoard

Data could not be loaded.

The TensorBoard server may be down or inaccessible.

Last reload:

```

```