```
#Mount Google Drive for Data
from google.colab import drive
drive.mount('/content/drive')
```

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

```
#Standard Python Library

import pandas as pd
import numpy as np
import seaborn as sns
import json
import os

#Standar NLP Pre-Processing Library

import nltk as nlp
from nltk.corpus import stopwords
import string
import re
import matplotlib.pyplot as plt
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer
from sklearn import feature_extraction, linear_model, model_selection, preprocessing
from sklearn.multiclass import OneVsRestClassifier
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn.pipeline import Pipeline
from sklearn.svm import LinearSVC
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import GaussianNB,BernoulliNB,MultinomialNB,MultinomialNB
from tensorflow.keras.utils import plot_model

#SK Learn and TF library for NLP NueralNet Layers
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import SGDClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.preprocessing import LabelEncoder, MultiLabelBinarizer
from sklearn.gaussian_process import GaussianProcessClassifier
from tensorflow.keras.models import Model, Sequential
from tensorflow.keras.layers import LSTM, Activation, Dense, Dropout, Input, Embeddinc
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing import sequence
from tensorflow.keras.callbacks import EarlyStopping
from tensorflow.keras.utils import to_categorical

from sklearn.metrics import accuracy_score, confusion_matrix, classification_report, 1
```

```
#Logistic Regression Library
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.neural_network import MLPClassifier

#Naive Bayes Library
from sklearn.naive_bayes import GaussianNB

#KNN Libraries
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import classification_report, confusion_matrix


#LSTM Libraries
from tensorflow.keras.layers import BatchNormalization, LSTM, GRU, Input, SpatialDropo
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras import optimizers
import tensorflow.keras.backend as backend
from nltk import word_tokenize, KneserNeyProbDist, SimpleGoodTuringProbDist, FreqDist,

#Import Ploting Library
import matplotlib.pyplot as plt

"""
READS Text File and return all of body as a string
"""
def read_input(input_path:str) -> str:
    file_data = open(input_path , 'r')

    return file_data.read()

"""
READS Directory Text File and return list of each text file
"""
def read_directory(input_dir:str):
  data = []
  files = [f for f in os.listdir(input_dir)] #if os.path.isfile(f)]
  for f in files:
    f_path = os.path.join(input_dir, f)
    with open (f_path, "r") as myfile:
      data.append(myfile.read())
  return data

#The Four Group of Data
train_pos = read_directory('/content/drive/MyDrive/aclImdb/train/pos')
train_neg = read_directory('/content/drive/MyDrive/aclImdb/train/neg')
test_pos = read_directory('/content/drive/MyDrive/aclImdb/test/pos')
test_neg = read_directory('/content/drive/MyDrive/aclImdb/test/pos')
```

```
"""
Text pre-processing function
"""
def preprocess(paragraph, label, sample_size):

    data_set = [paragraph.strip() for paragraph in paragraphs if len(paragraph) > samp
    data = [re.sub('[\W_]+', ' ', sample.lower().strip()) for sample in data_set]
    size = len(data)

    label_array = np.ones((size,)) * label
    df = pd.DataFrame({'paragraph': data, 'category':label_array })
    print('The total number of examples for category ' + str(label)+ ' is: ' + str(siz
    return df, size



X = pd.concat([pd.DataFrame(np.array(train_pos)),pd.DataFrame(np.array(train_neg))])
X.shape
X2 = pd.concat([pd.DataFrame(np.array(test_pos)),pd.DataFrame(np.array(test_neg))])
X2.shape
```

```
(25000, 1)
```

```
df = pd.DataFrame(np.ones((len(train_pos),1)))
df2 = pd.DataFrame(np.zeros((len(train_neg),1)))
Y = pd.concat([df,df2])
Y.shape

df = pd.DataFrame(np.ones((len(test_pos),1)))
df2 = pd.DataFrame(np.zeros((len(test_neg),1)))
Y2 = pd.concat([df,df2])
Y2.shape
```

```
(25000, 1)
```

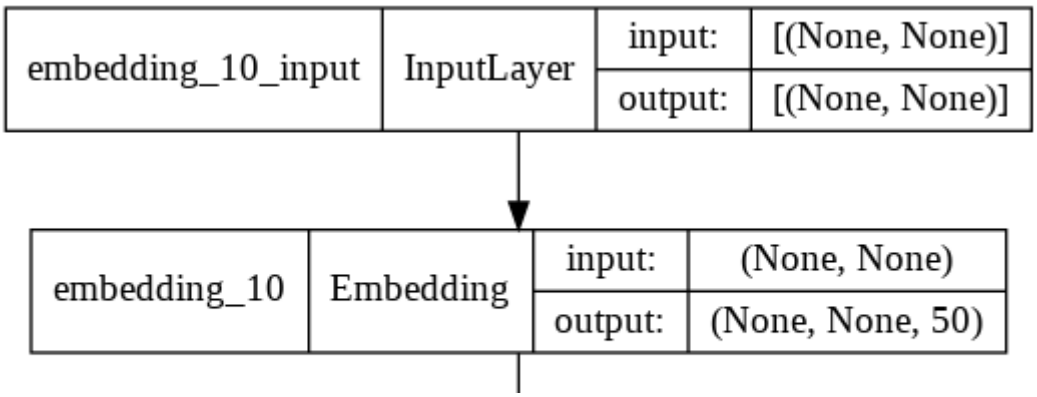
```
vocab = read_input('/content/drive/MyDrive/aclImdb/imdb.vocab').split('\n')
vocab_size = len(vocab) + 1


tokenizer = Tokenizer(num_words=3000)
tokenizer.fit_on_texts(X[0])

X_train = tokenizer.texts_to_sequences(X[0])
X_test = tokenizer.texts_to_sequences(X2[0])

maxlen = 100

X_train = pad_sequences(X_train, padding='post', maxlen=maxlen)
X_test = pad_sequences(X_test, padding='post', maxlen=maxlen)
```

```
label_train = to_categorical(Y)
label_test = to_categorical(Y2)
```

```
inputs = Input(shape=(maxlen,))
lstm_model = Sequential()
lstm_model.add(Embedding(vocab_size, 100))
lstm_model.add(Dense(8, activation='relu'))
lstm_model.add(LSTM(100))
lstm_model.add(Dense(2, activation='softmax'))
lstm_model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accura
print(lstm_model.summary())
```

```
Model: "sequential_12"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 embedding_10 (Embedding)    (None, None, 50)          4476400   

 dense_10 (Dense)            (None, None, 8)           408       

 lstm_9 (LSTM)               (None, 50)                11800     

 dense_11 (Dense)            (None, 2)                 102       

=================================================================
Total params: 4,488,710
Trainable params: 4,488,710
Non-trainable params: 0
_________________________________________________________________
None
```

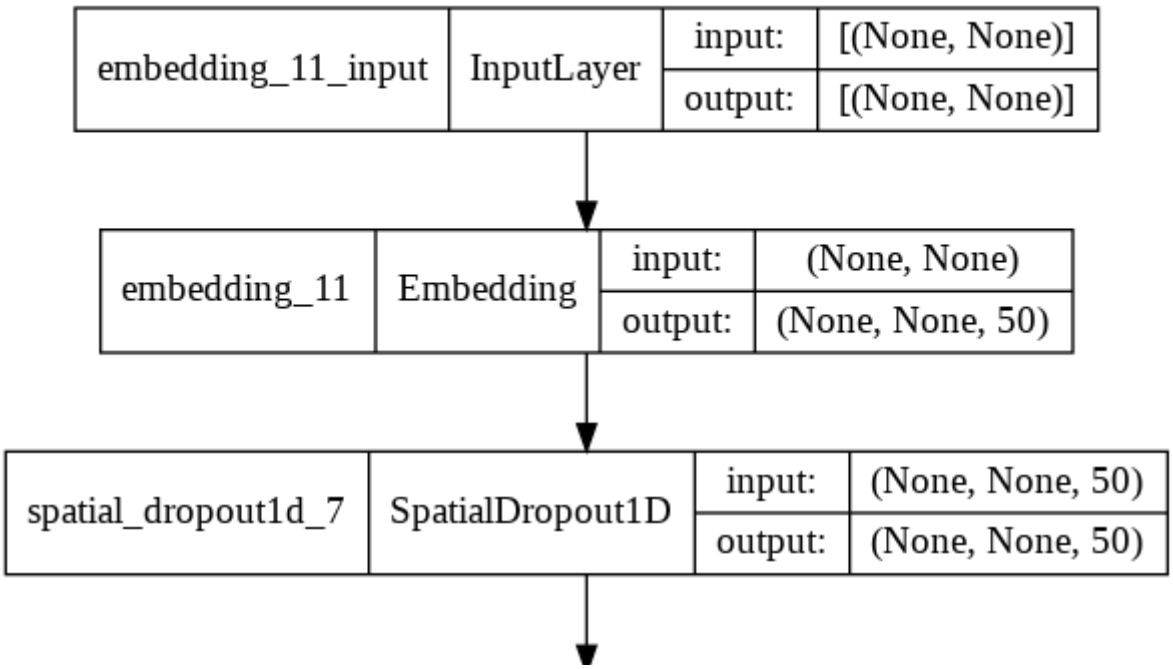
```
plot_model(lstm_model, show_shapes=True, to_file='lstm_model.png')
```




```
model = Sequential()
model.add(Embedding(vocab_size, 100))
model.add(SpatialDropout1D(0.2))
model.add(LSTM(50, dropout=0.2, recurrent_dropout=0.2))
model.add(Dense(2, activation='softmax'))
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])

print(model.summary())
```

```
Model: "sequential_13"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_11 (Embedding)    (None, None, 50)          4476400

 spatial_dropout1d_7 (Spatia  (None, None, 50)         0
 lDropout1D)

 lstm_10 (LSTM)              (None, 50)                20200

 dense_12 (Dense)            (None, 2)                 102

=================================================================
Total params: 4,496,702
Trainable params: 4,496,702
Non-trainable params: 0
_________________________________________________________________
None
```

```
plot_model(model, show_shapes=True, to_file='model.png')
```




```
epochs = 10
batch_size = 128
```

```
history = model.fit(X_train, label_train, epochs=epochs, batch_size=batch_size)
```

```
Epoch 1/50
49/49 [==============================] - 31s 578ms/step - loss: 0.6124 - accuracy
Epoch 2/50
49/49 [==============================] - 28s 574ms/step - loss: 0.3690 - accuracy
Epoch 3/50
49/49 [==============================] - 28s 576ms/step - loss: 0.3131 - accuracy
Epoch 4/50
49/49 [==============================] - 28s 577ms/step - loss: 0.2898 - accuracy
Epoch 5/50
49/49 [==============================] - 28s 579ms/step - loss: 0.2792 - accuracy
Epoch 6/50
49/49 [==============================] - 28s 580ms/step - loss: 0.2718 - accuracy
Epoch 7/50
49/49 [==============================] - 28s 576ms/step - loss: 0.2642 - accuracy
Epoch 8/50
49/49 [==============================] - 28s 576ms/step - loss: 0.2570 - accuracy
Epoch 9/50
49/49 [==============================] - 28s 577ms/step - loss: 0.2519 - accuracy
Epoch 10/50
49/49 [==============================] - 28s 577ms/step - loss: 0.2405 - accuracy
Epoch 11/50
49/49 [==============================] - 28s 581ms/step - loss: 0.2391 - accuracy
Epoch 12/50
49/49 [==============================] - 28s 580ms/step - loss: 0.2243 - accuracy
Epoch 13/50
49/49 [==============================] - 28s 578ms/step - loss: 0.2164 - accuracy
Epoch 14/50
49/49 [==============================] - 29s 583ms/step - loss: 0.2089 - accuracy
Epoch 15/50
49/49 [==============================] - 28s 579ms/step - loss: 0.2002 - accuracy
```

```
Epoch 16/50
49/49 [==============================] - 29s 583ms/step - loss: 0.1924 - accuracy
Epoch 17/50
49/49 [==============================] - 28s 580ms/step - loss: 0.1844 - accuracy
Epoch 18/50
49/49 [==============================] - 28s 577ms/step - loss: 0.1788 - accuracy
Epoch 19/50
49/49 [==============================] - 28s 579ms/step - loss: 0.1753 - accuracy
Epoch 20/50
49/49 [==============================] - 28s 572ms/step - loss: 0.1672 - accuracy
Epoch 21/50
49/49 [==============================] - 28s 581ms/step - loss: 0.1708 - accuracy
Epoch 22/50
49/49 [==============================] - 28s 578ms/step - loss: 0.1670 - accuracy
Epoch 23/50
49/49 [==============================] - 28s 580ms/step - loss: 0.1512 - accuracy
Epoch 24/50
49/49 [==============================] - 28s 575ms/step - loss: 0.1544 - accuracy
Epoch 25/50
49/49 [==============================] - 28s 571ms/step - loss: 0.1470 - accuracy
Epoch 26/50
49/49 [==============================] - 28s 581ms/step - loss: 0.1428 - accuracy
Epoch 27/50
49/49 [==============================] - 28s 579ms/step - loss: 0.1415 - accuracy
Epoch 28/50
49/49 [==============================] - 28s 576ms/step - loss: 0.1402 - accuracy
Epoch 29/50
49/49 [==============================] - 29s 581ms/step - loss: 0.1370 - accuracy
```

```
lstm_model.fit(X_train, label_train, epochs=epochs, batch_size=batch_size)
```

```
Epoch 1/50
49/49 [==============================] - 17s 310ms/step - loss: 0.6554 - accuracy
Epoch 2/50
49/49 [==============================] - 15s 310ms/step - loss: 0.4222 - accuracy
Epoch 3/50
49/49 [==============================] - 15s 310ms/step - loss: 0.3231 - accuracy
Epoch 4/50
49/49 [==============================] - 15s 312ms/step - loss: 0.2978 - accuracy
Epoch 5/50
49/49 [==============================] - 15s 312ms/step - loss: 0.2814 - accuracy
Epoch 6/50
49/49 [==============================] - 15s 315ms/step - loss: 0.2694 - accuracy
Epoch 7/50
49/49 [==============================] - 15s 312ms/step - loss: 0.2629 - accuracy
Epoch 8/50
49/49 [==============================] - 15s 314ms/step - loss: 0.2565 - accuracy
Epoch 9/50
49/49 [==============================] - 15s 312ms/step - loss: 0.2540 - accuracy
Epoch 10/50
49/49 [==============================] - 15s 313ms/step - loss: 0.2491 - accuracy
Epoch 11/50
49/49 [==============================] - 15s 310ms/step - loss: 0.2451 - accuracy
Epoch 12/50
49/49 [==============================] - 15s 306ms/step - loss: 0.2439 - accuracy
Epoch 13/50
```

```
49/49 [==============================] - 15s 308ms/step - loss: 0.2417 - accuracy
Epoch 14/50
49/49 [==============================] - 15s 308ms/step - loss: 0.2371 - accuracy
Epoch 15/50
49/49 [==============================] - 15s 307ms/step - loss: 0.2365 - accuracy
Epoch 16/50
49/49 [==============================] - 15s 308ms/step - loss: 0.2331 - accuracy
Epoch 17/50
49/49 [==============================] - 15s 309ms/step - loss: 0.2315 - accuracy
Epoch 18/50
49/49 [==============================] - 15s 307ms/step - loss: 0.2285 - accuracy
Epoch 19/50
49/49 [==============================] - 15s 306ms/step - loss: 0.2276 - accuracy
Epoch 20/50
49/49 [==============================] - 15s 307ms/step - loss: 0.2226 - accuracy
Epoch 21/50
49/49 [==============================] - 15s 308ms/step - loss: 0.2181 - accuracy
Epoch 22/50
49/49 [==============================] - 15s 310ms/step - loss: 0.2138 - accuracy
Epoch 23/50
49/49 [==============================] - 15s 307ms/step - loss: 0.2137 - accuracy
Epoch 24/50
49/49 [==============================] - 15s 307ms/step - loss: 0.2105 - accuracy
Epoch 25/50
49/49 [==============================] - 15s 308ms/step - loss: 0.2043 - accuracy
Epoch 26/50
49/49 [==============================] - 15s 309ms/step - loss: 0.1999 - accuracy
Epoch 27/50
49/49 [==============================] - 15s 308ms/step - loss: 0.1936 - accuracy
Epoch 28/50
49/49 [==============================] - 15s 308ms/step - loss: 0.1926 - accuracy
Epoch 29/50
49/49 [==============================] - 15s 309ms/step - loss: 0.1881 - accuracy
```

```
lstm_loss = lstm_model.evaluate(X_test, label_test)
loss = lstm_loss[0]
l_acc_test = lstm_loss[1]
print('loss = ' + str(loss))
print('accuracy = ' + str(l_acc_test))
```

```
782/782 [==============================] - 11s 13ms/step - loss: 2.1911 - accura
loss = 2.1911263465881348
accuracy = 0.5
```

```
model_loss = model.evaluate(X_test, label_test)
m_loss = model_loss[0]
acc_test = model_loss[1]
print('loss = ' + str(m_loss))
print('accuracy = ' + str(acc_test))
```

```
782/782 [==============================] - 13s 16ms/step - loss: 2.9451 - accura
loss = 2.945063352584839
accuracy = 0.5
```

✓ 12s completed at 9:05 PM