```
Introduction
Millions of people are using Twitter and expressing their emotions like happiness, sadness, angry, etc. The Sentiment analysis is also about detec



Problem description/definition:
To devise a sentimental analyzer for overcoming the challenges to identify the twitter tweets text sentiments (positive, negative) by implementing
Evolution measures:
After training the model, we apply the evaluation measures to check that how the model is getting predictions. We will use the following evaluatic

Accuracy
Confusion matrix with plot
ROC Curve
Technical Approach
We are using python language in the implementations and Jupter Notebook that support the machine learning and data science projects. We will builc

Source of Data:
https://www.kaggle.com/kazanova/sentiment140
```

Importing Python Libraries

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
plt.style.use('ggplot')
from sklearn.metrics import roc_curve, auc
from sklearn.metrics import classification_report, confusion_matrix
import nltk
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer
from sklearn.model_selection import train_test_split
from mlxtend.plotting import plot_confusion_matrix
import matplotlib.cm as cm
from matplotlib import rcParams
from collections import Counter
from nltk.tokenize import RegexpTokenizer
import re
import string
from tensorflow.keras.layers import LSTM, Activation, Dense, Dropout, Input, Embedding
from tensorflow.keras.models import Model
from tensorflow.keras.optimizers import RMSprop
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing import sequence
%matplotlib inline

import warnings
warnings.filterwarnings("ignore")
```

We are uisng the following versions of the libraries: numpy == 1.18.5

pandas == 1.1.3

tensorflow ==1.7.0

keras == 2.4.3

nltk ==3.5

seaborn ==0.11.0

How we can install the libraries in python? To install the python library is very easy pip install name_of_library Like if you wanted to install tensorflow? pip install tensforflow

```
# Loading the data 📁 📂
data = pd.read_csv("/content/training.1600000.processed.noemoticon.csv", encoding = "ISO-8859-1", engine="python")
data.columns = ["label", "time", "date", "query", "username", "text"]
```

```
#Exploratory data analysis 🔎 📊
#Five top records of data
data.head()
```

	label	time	date	query	username	text
0	0	1467810672	Mon Apr 06 22:19:49 PDT 2009	NO_QUERY	scotthamilton	is upset that he can't update his Facebook by ...
1	0	1467810917	Mon Apr 06 22:19:53 PDT 2009	NO_QUERY	mattycus	@Kenichan I dived many times for the ball. Man...
2	0	1467811184	Mon Apr 06 22:19:57 PDT 2009	NO_QUERY	ElleCTF	my whole body feels itchy and like its on fire
3	0	1467811193	Mon Apr 06 22:19:57 PDT 2009	NO_QUERY	Karoli	@nationwideclass no, it's not behaving at all....
4	0	1467811372	Mon Apr 06 22:20:00 PDT 2009	NO_QUERY	joy_wolf	@Kwesidei not the whole crew

```
data.tail()
```

	label	time	date	query	username	text
1599994	4	2193601966	Tue Jun 16 08:40:49 PDT 2009	NO_QUERY	AmandaMarie1028	Just woke up. Having no school is the best fee...
1599995	4	2193601969	Tue Jun 16 08:40:49 PDT 2009	NO_QUERY	TheWDBoards	TheWDB.com - Very cool to hear old Walt interv...
1599996	4	2193601991	Tue Jun 16 08:40:49 PDT 2009	NO_QUERY	bpbabe	Are you ready for your MoJo Makeover? Ask me f...

```
data.columns
```

```
Index(['label', 'time', 'date', 'query', 'username', 'text'], dtype='object')
```

```
print('lenght of data is', len(data))
```

```
lenght of data is 1599999
```

```
data.shape
```

```
(1599999, 6)
```

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1599999 entries, 0 to 1599998
Data columns (total 6 columns):
 #   Column    Non-Null Count    Dtype 
---  ------    --------------    ----- 
 0   label     1599999 non-null  int64 
 1   time      1599999 non-null  int64 
 2   date      1599999 non-null  object
 3   query     1599999 non-null  object
 4   username  1599999 non-null  object
 5   text      1599999 non-null  object
dtypes: int64(2), object(4)
memory usage: 73.2+ MB
```

```
data.dtypes
```

	0
label	int64
time	int64
date	object
query	object
username	object
text	object

```
np.sum(data.isnull().any(axis=1))
```

```
0
```

```
print('Count of columns in the data is:  ', len(data.columns))
```

```
Count of columns in the data is:   6
```

```
print('Count of rows in the data is:  ', len(data))
```

```
Count of rows in the data is:   1599999
```

Data Preparation 📝 Selection of interested coloumns

Assinged 1 to class 4

Took one fourth data so we can run on our machine easily

Combined positive and negative tweets

We will convert the text in lower case for the further working on tweet text.

We will clean and remove the stop words(of, a, in etc) from statement because these words are not useuseful to support the labels of sentiments data

We will clean and remove the punctuations because these are the noise in the data and not meaningfull

We will clean and remove repeating characters in the words

We will clean and remove emails

We will clean and remove URL's

We will clean and remove the numbers in the data

We will apply tokenization(to separate the sentence into words)

We will apply stemming and lemmatization on the text. The concept of both is following:

We will apply stemming on the tweet text.

We will apply lemmatization on the tweet text.

Separated input feature and labels

Extracted features from input feature

Separated the 70% data for training and 30% data for testing

```
#Selecting the text and label coloumn
data=data[['text','label']]
```

```
#Assigning 1 to Positive sentment 4
data['label'][data['label']==4]=1
```

```
#Separating positive and negative tweets
data_pos = data[data['label'] == 1]
data_neg = data[data['label'] == 0]
```

```
#taking one fourth data so we can run on our machine easily
data_pos = data_pos.iloc[:int(20000)]
data_neg = data_neg.iloc[:int(20000)]
```

```
#Combining positive and negative tweets
data = pd.concat([data_pos, data_neg])
```

```
#Making statement text in lower cases
data['text']=data['text'].str.lower()
```

```
data['text'].tail()
```

	text
19995	one more day of holidays
19996	feeling so down right now .. i hate you damn h...
19997	geez,i hv to read the whole book of personalit...
19998	i threw my sign at donnie and he bent over to ...
19999	@heather2711 good thing i didn't find any then...

```
!pip install nltk

import nltk
```

```
nltk.download('stopwords')
```

```
Requirement already satisfied: nltk in /usr/local/lib/python3.10/dist-packages (3.9.1)
Requirement already satisfied: click in /usr/local/lib/python3.10/dist-packages (from nltk) (8.1.8)
Requirement already satisfied: joblib in /usr/local/lib/python3.10/dist-packages (from nltk) (1.4.2)
Requirement already satisfied: regex>=2021.8.3 in /usr/local/lib/python3.10/dist-packages (from nltk) (2024.11.6)
Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-packages (from nltk) (4.67.1)
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
True
```

```
#Cleaning and removing Stop words of english

stopwords_list = stopwords.words('english')
```

```
from nltk.corpus import stopwords
", ".join(stopwords.words('english'))
```

```
'i, me, my, myself, we, our, ours, ourselves, you, you're, you've, you'll, you'd, your, yours, yourself, yourselves, he, him, his,
himself, she, she's, her, hers, herself, it, it's, its, itself, they, them, their, theirs, themselves, what, which, who, whom, thi
s, that, that'll, these, those, am, is, are, was, were, be, been, being, have, has, had, having, do, does, did, doing, a, an, the,
and, but, if, or, because, as, until, while, of, at, by, for, with, about, against, between, into, through, during, before, after,
above, below, to, from, up, down, in, out, on, off, over, under, again, further, then, once, here, there, when, where, why, how, al
```

```
#Cleaning and removing the above stop words list from the tweet text
STOPWORDS = set(stopwords.words('english'))
def cleaning_stopwords(text):
    return " ".join([word for word in str(text).split() if word not in STOPWORDS])
data['text'] = data['text'].apply(lambda text: cleaning_stopwords(text))
data['text'].head()
```

```
                                                   text
799999                  love @health4uandpets u guys r best!!
800000       im meeting one besties tonight! cant wait!! - ...
800001   @darealsunisakim thanks twitter add, sunisa! g...
800002       sick really cheap hurts much eat real food plu...
800003                     @lovesbrooklyn2 effect everyone
```

```
#Cleaning and removing punctuations
english_punctuations = string.punctuation
punctuations_list = english_punctuations
def cleaning_punctuations(text):
    translator = str.maketrans('', '', punctuations_list)
    return text.translate(translator)
```

```
data['text']= data['text'].apply(lambda x: cleaning_punctuations(x))
data['text'].tail()
```

```
                                                  text
19995                                     one day holidays
19996                         feeling right hate damn humprey
19997   geezi hv read whole book personality types emb...
19998      threw sign donnie bent get thingee made sad face
19999     heather2711 good thing find none ones like com...
```

```
#Cleaning and removing repeating characters
def cleaning_repeating_char(text):
    return re.sub(r'(.)\1+', r'\1', text)
```

```
data['text'] = data['text'].apply(lambda x: cleaning_repeating_char(x))
data['text'].tail()
```

	text
19995	one day holidays
19996	feling right hate damn humprey
19997	gezi hv read whole bok personality types embar...
19998	threw sign donie bent get thinge made sad face
19999	heather271 god thing find none ones like come ...

```
#Cleaning and removing email
def cleaning_email(data):
    return re.sub('@[^\s]+', ' ', data)
```

```
data['text']= data['text'].apply(lambda x: cleaning_email(x))
data['text'].tail()
```

	text
19995	one day holidays
19996	feling right hate damn humprey
19997	gezi hv read whole bok personality types embar...
19998	threw sign donie bent get thinge made sad face
19999	heather271 god thing find none ones like come ...

```
#Cleaning and removing URL's
def cleaning_URLs(data):
    return re.sub('((www\.[^\s]+)|(https?://[^\s]+))',' ',data)
```

```
data['text'] = data['text'].apply(lambda x: cleaning_URLs(x))
data['text'].tail()
```

	text
19995	one day holidays
19996	feling right hate damn humprey
19997	gezi hv read whole bok personality types embar...
19998	threw sign donie bent get thinge made sad face
19999	heather271 god thing find none ones like come ...

```
#Cleaning and removing Numeric numbers
def cleaning_numbers(data):
    return re.sub('[0-9]+', '', data)
```

```
data['text'] = data['text'].apply(lambda x: cleaning_numbers(x))
data['text'].tail()
```

	text
19995	one day holidays
19996	feling right hate damn humprey
19997	gezi hv read whole bok personality types embar...
19998	threw sign donie bent get thinge made sad face
19999	heather god thing find none ones like come siz...

```
#Getting tokenization of tweet text
tokenizer = RegexpTokenizer(r'\w+')
data['text'] = data['text'].apply(tokenizer.tokenize)
```

```
data['text'].head()
```

	text
799999	[love, healthuandpets, u, guys, r, best]
800000	[im, meting, one, besties, tonight, cant, wait...
800001	[darealsunisakim, thanks, twiter, ad, sunisa, ...
800002	[sick, realy, cheap, hurts, much, eat, real, f...
800003	[lovesbroklyn, efect, everyone]

```
#Applying Stemming
st = nltk.PorterStemmer()
def stemming_on_text(data):
    text = [st.stem(word) for word in data]
    return data

data['text']= data['text'].apply(lambda x: stemming_on_text(x))
```

```
data['text'].head()
```

	text
799999	[love, healthuandpets, u, guys, r, best]
800000	[im, meting, one, besties, tonight, cant, wait...
800001	[darealsunisakim, thanks, twiter, ad, sunisa, ...
800002	[sick, realy, cheap, hurts, much, eat, real, f...
800003	[lovesbroklyn, efect, everyone]

```
!pip install nltk
import nltk

nltk.download('wordnet')
nltk.download('stopwords')

# removed the extra 'python' at the end as it caused a syntax error
```

```
Requirement already satisfied: nltk in /usr/local/lib/python3.10/dist-packages (3.9.1)
Requirement already satisfied: click in /usr/local/lib/python3.10/dist-packages (from nltk) (8.1.8)
Requirement already satisfied: joblib in /usr/local/lib/python3.10/dist-packages (from nltk) (1.4.2)
Requirement already satisfied: regex>=2021.8.3 in /usr/local/lib/python3.10/dist-packages (from nltk) (2024.11.6)
Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-packages (from nltk) (4.67.1)
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
True
```

```
#Applying Lemmatizer
lm = nltk.WordNetLemmatizer()
def lemmatizer_on_text(data):
    text = [lm.lemmatize(word) for word in data]
    return data

data['text'] = data['text'].apply(lambda x: lemmatizer_on_text(x))
```

```
data['text'].head()
```

	text
799999	[love, healthuandpets, u, guys, r, best]
800000	[im, meting, one, besties, tonight, cant, wait...
800001	[darealsunisakim, thanks, twiter, ad, sunisa, ...
800002	[sick, realy, cheap, hurts, much, eat, real, f...
800003	[lovesbroklyn, efect, everyone]

Labels : labels are the targets like in this project senitments of the tweets are labels.

Inputs : Inputs are the data that we feed into machine learning like in this project tweets texts are the inputs.

Training Data-- We use training data when we train the models. We feed train data to machine learning and deep learning models so that model can learn from the data.

Validation Data-- We use validation data while training the model. We use this data to evalaute the performance that how the model perform on training time.

Testing Data-- We use testing data after training the model. We use this data to evalaute the performance that how the model perform after training. So in this way first we get predictions from the trained model without giving the labels and then we compare the true labels with predictions and get the performance of th model..

```
#Separating input feature and label
X=data.text
y=data.label
```

Preparing the input features for training

1.We converting the text words into arrays form.

2.aximum 500 features/words selected for training. These 500 words will be selected on the importance that will distinguish between the positive tweets and negative tweets

```
max_len = 500
tok = Tokenizer(num_words=2000)
tok.fit_on_texts(X)
sequences = tok.texts_to_sequences(X)
sequences_matrix = sequence.pad_sequences(sequences,maxlen=max_len)
```

```
#As we can see that there total 40000 tweets and the number words/features are 500.
```

```
sequences_matrix.shape
```

```
(40000, 500)
```

Separating the 70% data for training data and 30% for testing data As we prepared all the tweets, now we are separating/splitting the tweets into training data and testing data.

1.70% tweets will be used in the training

2.30% tweets will be used to test the performance of the model.

```
X_train, X_test, Y_train, Y_test = train_test_split(sequences_matrix, y, test_size=0.3, random_state=2)
```

Implementing Tensorflow based model for training 🧪

Step 1

- The input to model is 500 words because these are the number features/words that we extracted above from text of tweets.

Step 2

- Embeddings provide the presentation of words and their relative meanings. Like in this, we are feeding the limit of maximum words, lenght of input words and the inputs of previous layer.

Step 3

- LSTM (long short term memory) save the words and predict the next words based on the previous words. LSTM is a sequance predictor of next coming words. Ref: https://static.wixstatic.com/media/3eee0b_969c1d3e8d7943f0bd693d6151199f69~mv2.gif

Step 4

- Dense layer reduce the outputs by getting inputs from Faltten layer. Dense layer use all the inputs of previous layer neurons and perform calculations and send 256 outputs

Step 5

- Activation function is node that is put at the end of all layers of neural network model or in between neural network layers. Activation function help to decide which neuron should be pass and which neuron should fire. So activation function of node defines the output of that node given an input or set of inputs.

Step 6

- Droupout layer drop some neurons from previous layers. why we apply this? We apply this to avoid the overfitting problems. In overfitting, model give good accuracy on training time but not good on testing time.

```
def tensorflow_based_model(): #Defined tensorflow_based_model function for training tenforflow based model
    inputs = Input(name='inputs',shape=[max_len])#step1
    layer = Embedding(2000,50,input_length=max_len)(inputs) #step2
    layer = LSTM(64)(layer) #step3
    layer = Dense(256,name='FC1')(layer) #step4
    layer = Activation('relu')(layer) # step5
    layer = Dropout(0.5)(layer) # step6
    layer = Dense(1,name='out_layer')(layer) #step4 again but this time its giving only one output as because we need to classify the twe
    layer = Activation('sigmoid')(layer) #step5 but this time activation function is sigmoid for only one output.
    model = Model(inputs=inputs,outputs=layer) #here we are getting the final output value in the model for classification
    return model #function returning the value when we call it
```

Model compilation-

First we are calling the model

We are using 2 classes so we set "binary_crossentropy" and if we use more than two classes then we use "categorical_crossentropy"

Optimizer is a function that used to change the features of neural network such as learning rate (how the model learn with features) in order to reduce the losses. So the learning rate of neural network to reduce the losses is defined by optimizer.

We are setting metrics=accuracy because we are going to caluclate the percentage of correct predictions over all predictions on the validation set

```
model = tensorflow_based_model() # here we are calling the function of created model
model.compile(loss='binary_crossentropy',optimizer=RMSprop(),metrics=['accuracy'])
```

Training and validating with parameter tuning¶

We are feeding the training data and getting 10% data for validation from training data

We set the following parameters:

Batch size =80 so the model take 80 tweets in each iteration and train them. Batch size is a term used in machine learning and refers to the number of training examples utilized in one iteration.

Epochs =6 so the model will train on the data 6 times.Epoch is a term used in machine learning and indicates the number of passes of the entire training dataset the machine learning algorithm has completed.

We can choose batch_size, and epochs as we want so the good practice is to set some values and train the model if the model will not give the good results we can change it and then try again for the training of the model. We can repeat this process many time untill we will not get the good results and this process called as parameter tuning.

```
history=model.fit(X_train,Y_train,batch_size=80,epochs=6, validation_split=0.1)# here we are starting the training of model by feeding t
print('Training finished !!')
```

```
Epoch 1/6
315/315 ━━━━━━━━━━━━━━━━━━━━ 136s 424ms/step - accuracy: 0.5729 - loss: 0.6644 - val_accuracy: 0.7304 - val_loss: 0.5335
Epoch 2/6
315/315 ━━━━━━━━━━━━━━━━━━━━ 131s 416ms/step - accuracy: 0.7513 - loss: 0.5160 - val_accuracy: 0.7439 - val_loss: 0.5195
Epoch 3/6
315/315 ━━━━━━━━━━━━━━━━━━━━ 132s 418ms/step - accuracy: 0.7309 - loss: 0.5582 - val_accuracy: 0.7418 - val_loss: 0.5181
Epoch 4/6
315/315 ━━━━━━━━━━━━━━━━━━━━ 130s 413ms/step - accuracy: 0.7630 - loss: 0.5002 - val_accuracy: 0.7104 - val_loss: 0.5508
Epoch 5/6
315/315 ━━━━━━━━━━━━━━━━━━━━ 140s 406ms/step - accuracy: 0.7612 - loss: 0.4943 - val_accuracy: 0.7425 - val_loss: 0.5167
Epoch 6/6
315/315 ━━━━━━━━━━━━━━━━━━━━ 131s 417ms/step - accuracy: 0.7737 - loss: 0.4795 - val_accuracy: 0.7443 - val_loss: 0.5177
Training finished !!
```

We need to do all the above configurations to train the model.

If we will not set all settings correctly then we could not get the desired results.

Testing the Trained model on test data Getting predictions/classifying the sentiments (positve/negative) on the test data using trained model.

```
accr1 = model.evaluate(X_test,Y_test) #we are starting to test the model here
```

```
375/375 ━━━━━━━━━━━━━━━━━━━━ 29s 78ms/step - accuracy: 0.7387 - loss: 0.5239
```

Accuracy

Accuracy is the number of correctly classify tweets from all the tweets of positive and negative.

For example, if the trained model classify the 70 tweets correct and 30 tweets wrong from total of 100 tweets then the accuracy score will be 70%.

Accuracy= Total number of correct predictions/Total number of predictions

```
print('Test set\n  Accuracy: {:0.2f}'.format(accr1[1])) #the accuracy of the model on test data is given below
```

```
Test set
  Accuracy: 0.74
```

Test set Accuracy: 0.75

Getting prediction of the test data and then we will compare the true labels/classes of the data with predictions

As the model give probabilties so we are setting a threshold 0.5. More than 0.5 will be the positive tweets and lower will be negative tweets

```
y_pred = model.predict(X_test) #getting predictions on the trained model
y_pred = (y_pred > 0.5)
```

```
375/375 ━━━━━━━━━━━━━━━━━━━━ 28s 74ms/step
```

Confusion matrix

These are the evaluation measures to evaluate the performance of the model. Dark blue boxes are the correct predictions with the trained model and sky blue boxes shows the wrong predictions.

4610 tweets correctly predicted as negative sentiments.

1399 tweets predicted positive sentiments but that were actually negative sentiments.

4247 tweets correctly predicted as postive sentiments.

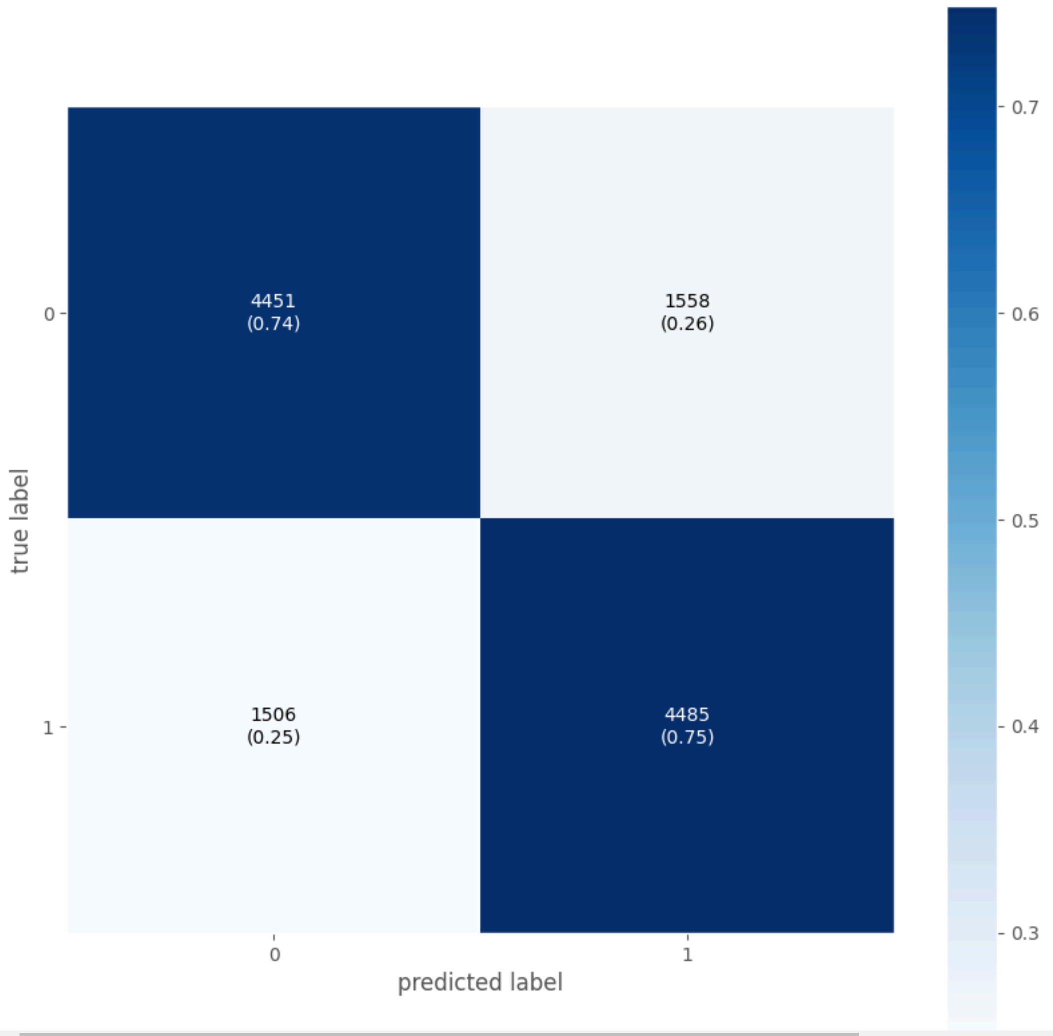
1744 tweets predicted negative sentiments but that were actually positive sentiments.

```
print('\n')
print("confusion matrix")
print('\n')
CR=confusion_matrix(Y_test, y_pred)
print(CR)
print('\n')

fig, ax = plot_confusion_matrix(conf_mat=CR,figsize=(10, 10),
                                show_absolute=True,
                                show_normed=True,
                                colorbar=True)
plt.show()
```

```
confusion matrix


[[4451 1558]
 [1506 4485]]
```



ROC CURVE ROC curve show the performance of the model as well.

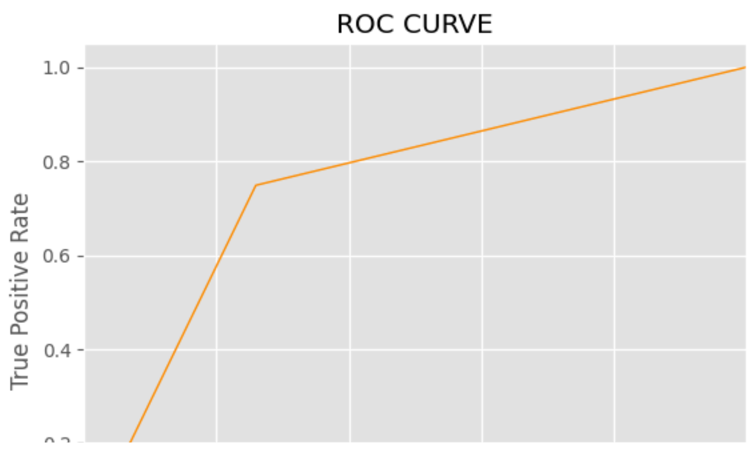
We can see that the model started from the 0 percent predictions and then moved to true positive predictions that are correct ROC curve (receiver operating characteristic curve) show the performance of a classification model at all the classification thresholds.

ROC plots two parameters,

True Positive Rate (correct predictions/classifications)

False Positive Rate (wrong predictions/classifications)

```
fpr, tpr, thresholds = roc_curve(Y_test, y_pred)
roc_auc = auc(fpr, tpr)
plt.figure()
plt.plot(fpr, tpr, color='darkorange', lw=1, label='ROC curve (area = %0.2f)' % roc_auc)
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC CURVE')
plt.legend(loc="lower right")
plt.show()
```



Conclusion 📝

We used the twitter sentiment analysis dataset and explored the data with different ways.

We prepared the text data of tweets by removing the unnecessary things. We trained model based on tensorflow with all settings.

We evaluated thye model with different evaluation measures.

If you are interested to work on any text based project, you can simply apply the same methodolgy but might be you will need to change little settings like name of coloumns etc.

We worked on the classification problem and sepcifically we call it binary classification which is two class classification.

```
Start coding or generate with AI.
```

```
Start coding or generate with AI.
```

```
Start coding or generate with AI.
```

1. **Data preprocessing:** The code performs extensive text preprocessing, including removing stop words, punctuation, and URLs, handling