In [1]:
```
#!unzip dank_data-master.zip
#!pip install tensorflow_addons
#!wget http://nlp.stanford.edu/data/glove.6B.zip
#!unzip glove*.zip
```

In [2]:
```
import glob
import pandas as pd
import warnings
warnings.filterwarnings("ignore")
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.layers import Dense,Input,MaxPool1D,Activation,Dropout,F
latten,Embedding,LSTM,concatenate
from tensorflow.keras.models import Model
import tensorflow as tf
import numpy as np
from tensorflow.keras.preprocessing.text import Tokenizer
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import StandardScaler
import tensorflow_addons as tfa
from tensorflow.keras.callbacks import LearningRateScheduler
from tensorflow.keras.callbacks import ReduceLROnPlateau
from tensorflow.keras.callbacks import ModelCheckpoint
from tensorflow.keras.callbacks import EarlyStopping
from sklearn.metrics import confusion_matrix,accuracy_score,f1_score
import seaborn as sns
import matplotlib.pyplot as plt
```

In [3]:
```
training='/content/dank_data-master/data/training/*'
test='/content/dank_data-master/data/test/*'
validation='/content/dank_data-master/data/validation/*'
```

In [4]:
```
training = glob.glob(training)
test = glob.glob(test)
validation = glob.glob(validation)
```

In [5]:
```
final_dank=pd.read_csv('/content/dank_data-master/data/final_dank.csv')
train_labels = [fn.split('/')[-1].split('.')[0].strip() for fn in training]
validation_labels = [fn.split('/')[-1].split('.')[0].strip() for fn in validat
ion]
test_labels = [fn.split('/')[-1].split('.')[0].strip() for fn in test]
```

In [6]:
```
for labels in train_labels:
    if labels==train_labels[0]:
        train_data =final_dank[final_dank['id']==labels]
    else :
        train_data =train_data.append(final_dank[final_dank['id']==labels],sor
t=False)
for labels in validation_labels:
    if labels==validation_labels[0]:
        val_data =final_dank[final_dank['id']==labels]
    else :
        val_data =val_data.append(final_dank[final_dank['id']==labels],sort=Fa
lse)
for labels in test_labels:
    if labels==test_labels[0]:
        test_data =final_dank[final_dank['id']==labels]
    else :
        test_data =test_data.append(final_dank[final_dank['id']==labels],sort=
False)
print(train_data.shape)
print(test_data.shape)
print(val_data.shape)
train_data.head(5)
```

```
(3405, 68)
(1719, 68)
(1688, 68)
```

Out[6]:

	Unnamed: 0	level_0	index	author	awards	processed_words	created_utc
15987	32142	33669.0	33669.0	orby9990	[]	['true', 'stori', 'btwthe', 'world', 'iranian'...	1.584339e+09
58209	103760	8570.0	40223.0	DarkKnight_Jedi	[]	['deserv', 'march', 'prove', 'cost', 'husband'...	1.584860e+09
45465	76129	79789.0	11442.0	LifeAfterRedditFalls	[]	['think', 'share', 'peac', 'final', 'option']	1.584535e+09
9635	22567	23666.0	23666.0	subbot9	[]	['love', 'love', 'reddit']	1.584394e+09
40396	68463	71751.0	3404.0	CalmProfit	[]	['behistorian', 'wonder', 'internet', 'respond...	1.584577e+09

In [7]:
```
train_data_words=train_data['processed_words'].values
validation_words=val_data['processed_words'].values
test_data_words=test_data['processed_words'].values

tokenizer = Tokenizer()
tokenizer.fit_on_texts(train_data_words)
vocab_size=len(tokenizer.word_index)
encoded_Xtrain_words = [tf.keras.preprocessing.text.one_hot(d, vocab_size,filt
ers='!"#$%&()*+,-./:;<=>?@[\\]^`{|}~\t\n') for d in train_data_words]
encoded_validation_words = [tf.keras.preprocessing.text.one_hot(d, vocab_size,
filters='!"#$%&()*+,-./:;<=>?@[\\]^`{|}~\t\n') for d in validation_words]
encoded_Xtest_words = [tf.keras.preprocessing.text.one_hot(d, vocab_size,filte
rs='!"#$%&()*+,-./:;<=>?@[\\]^`{|}~\t\n') for d in test_data_words]

padded_Xtrain_words = tf.keras.preprocessing.sequence.pad_sequences(encoded_Xt
rain_words, maxlen=20, padding='post')
padded_Xvalidation_words = tf.keras.preprocessing.sequence.pad_sequences(encod
ed_validation_words, maxlen=20, padding='post')
padded_Xtest_words = tf.keras.preprocessing.sequence.pad_sequences(encoded_Xte
st_words, maxlen=20, padding='post')
```

In [8]:
```
embeddings_index = dict()
f = open('/content/glove.6B.300d.txt')

for line in f:
    values = line.split()
    word = values[0]
    coefs = np.asarray(values[1:], dtype='float32')
    embeddings_index[word] = coefs

f.close()
print('Loaded %s word vectors.' % len(embeddings_index))
```

```
Loaded 400000 word vectors.
```

In [9]:
```
embedding_matrix = np.zeros((vocab_size+1, 300))
for word, i in tokenizer.word_index.items():
    embedding_vector = embeddings_index.get(word)
    if embedding_vector is not None:
        embedding_matrix[i] = embedding_vector
```

In [10]:
```
labelencoder = LabelEncoder()
labelencoder.fit(train_data['subreddit'].values)
subreddit_train=labelencoder.transform(train_data['subreddit'].values).reshape
(-1,1)
subreddit_validation=labelencoder.transform(val_data['subreddit'].values).resh
ape(-1,1)
subreddit_test=labelencoder.transform(test_data['subreddit'].values).reshape(-
1, 1)

print(subreddit_train.shape)
print(subreddit_test.shape)
print(subreddit_validation.shape)
```

```
(3405, 1)
(1719, 1)
(1688, 1)
```

In [11]:
```
labelencoder = LabelEncoder()
labelencoder.fit(train_data['is_nsfw'].values)
is_nsfw_train=labelencoder.transform(train_data['is_nsfw'].values).reshape(-1,
1)
is_nsfw_validation=labelencoder.transform(val_data['is_nsfw'].values).reshape(
-1,1)
is_nsfw_test=labelencoder.transform(test_data['is_nsfw'].values).reshape(-1,1)

print(is_nsfw_train.shape)
print(is_nsfw_test.shape)
print(is_nsfw_validation.shape)
```

```
(3405, 1)
(1719, 1)
(1688, 1)
```

In [12]:
```
scaler = StandardScaler()
scaler=scaler.fit(train_data['created_utc'].values.reshape(-1, 1))

created_utc_train=scaler.transform(train_data['created_utc'].values.reshape(-1
, 1))
created_utc_validation=scaler.transform(val_data['created_utc'].values.reshape
(-1, 1))
created_utc_test=scaler.transform(test_data['created_utc'].values.reshape(-1,
1))

print(created_utc_train.shape)
print(created_utc_test.shape)
print(created_utc_validation.shape)
```

```
(3405, 1)
(1719, 1)
(1688, 1)
```

In [13]:
```
time_of_day_train=(train_data['time_of_day'].values).reshape(-1,1)
time_of_day_validation=(val_data['time_of_day'].values).reshape(-1,1)
time_of_day_test=(test_data['time_of_day'].values).reshape(-1,1)

print(time_of_day_train.shape)
print(time_of_day_validation.shape)
print(time_of_day_test.shape)
```

```
(3405, 1)
(1688, 1)
(1719, 1)
```

In [14]:
```
scaler = StandardScaler()
scaler=scaler.fit(train_data['subscribers'].values.reshape(-1, 1))

subscribers_train=scaler.transform(train_data['subscribers'].values.reshape(-1
, 1))
subscribers_validation=scaler.transform(val_data['subscribers'].values.reshape
(-1, 1))
subscribers_test=scaler.transform(test_data['subscribers'].values.reshape(-1,
1))

print(subscribers_train.shape)
print(subscribers_validation.shape)
print(subscribers_test.shape)
```

```
(3405, 1)
(1688, 1)
(1719, 1)
```

In [15]:
```
#words  embeding layer
words =Input(shape=(20,),name="words")
embeding=Embedding(vocab_size+1,300,weights=[embedding_matrix],input_length=20
,trainable=False)(words)
lstm_layer=LSTM(100)(embeding)
flatten1 = Flatten(data_format='channels_last')(lstm_layer)

#categore_data
subreddit_train_layer =Input(shape=(subreddit_train.shape[1],),name="subreddit
_train_layer")
flatten2= Flatten(data_format='channels_last')(subreddit_train_layer)
###
is_nsfw_train_layer =Input(shape=(is_nsfw_train.shape[1],),name="is_nsfw_train
_layer")
flatten3 = Flatten(data_format='channels_last')(is_nsfw_train_layer)
####
time_of_day_train_layer =Input(shape=(time_of_day_train.shape[1],),name="time_
of_day_train_layer")
flatten4 = Flatten(data_format='channels_last')(time_of_day_train_layer)

#numeric_data
created_utc_train_layer =Input(shape=(created_utc_train.shape[1],),name="creat
ed_utc_train_layer")
created_utc_dence = Dense(units=3,activation='relu',kernel_initializer=tf.kera
s.initializers.glorot_normal(seed=33))(created_utc_train_layer)

#numeric_data
subscribers_train_layer =Input(shape=(subscribers_train.shape[1],),name="subsc
ribers_train_layer")
subscribers_dence = Dense(units=3,activation='relu',kernel_initializer=tf.kera
s.initializers.glorot_normal(seed=33))(subscribers_train_layer)

#concat layer
concatenated = concatenate([subscribers_train_layer,created_utc_dence,flatten4
,flatten3,flatten2,flatten1],axis = -1)

dense_layer1 = Dense(units=128,activation='relu',kernel_initializer=tf.keras.i
nitializers.glorot_normal(seed=33))(concatenated)
dropout1=Dropout(0.3)(dense_layer1)

dense_layer2 = Dense(units=64,activation='relu',kernel_initializer=tf.keras.in
itializers.glorot_normal(seed=33))(dropout1)
dropout2=Dropout(0.3)(dense_layer2)

dense_layer3 = Dense(units=64,activation='relu',kernel_initializer=tf.keras.in
itializers.glorot_normal(seed=33))(dropout2)


Out = Dense(units=2,activation='softmax',kernel_initializer=tf.keras.initializ
ers.glorot_normal(seed=3),name='Output')(dense_layer3)

model = Model(inputs=[words,subreddit_train_layer,is_nsfw_train_layer,time_of_
day_train_layer,created_utc_train_layer,subscribers_train_layer],outputs=Out)
#model = Model(inputs=[words,subreddit_train_layer,is_nsfw_train_layer,time_of
_day_train_layer,created_utc_train_layer,subscribers_train_layer],outputs=Out)
model.summary()
```

```
Model: "model"
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
words (InputLayer)              [(None, 20)]         0
__________________________________________________________________________________________________
embedding (Embedding)           (None, 20, 300)      2712000     words[0][0]
__________________________________________________________________________________________________
created_utc_train_layer (InputL [(None, 1)]          0
__________________________________________________________________________________________________
time_of_day_train_layer (InputL [(None, 1)]          0
__________________________________________________________________________________________________
is_nsfw_train_layer (InputLayer [(None, 1)]          0
__________________________________________________________________________________________________
subreddit_train_layer (InputLay [(None, 1)]          0
__________________________________________________________________________________________________
lstm (LSTM)                     (None, 100)          160400      embedding[0]
[0]
__________________________________________________________________________________________________
subscribers_train_layer (InputL [(None, 1)]          0
__________________________________________________________________________________________________
dense (Dense)                   (None, 3)            6           created_utc_
train_layer[0][0]
__________________________________________________________________________________________________
flatten_3 (Flatten)             (None, 1)            0           time_of_day_
train_layer[0][0]
__________________________________________________________________________________________________
flatten_2 (Flatten)             (None, 1)            0           is_nsfw_trai
n_layer[0][0]
__________________________________________________________________________________________________
flatten_1 (Flatten)             (None, 1)            0           subreddit_tr
ain_layer[0][0]
__________________________________________________________________________________________________
flatten (Flatten)               (None, 100)          0           lstm[0][0]
__________________________________________________________________________________________________
concatenate (Concatenate)       (None, 107)          0           subscribers_
train_layer[0][0]
                                                                 dense[0][0]
                                                                 flatten_3[0]
[0]
                                                                 flatten_2[0]
[0]
```

```
                                                                    flatten_1[0]
[0]
                                                                    flatten[0]
[0]
__________________________________________________________________________________
__________________
dense_2 (Dense)                 (None, 128)          13824       concatenate
[0][0]
__________________________________________________________________________________
__________________
dropout (Dropout)               (None, 128)          0           dense_2[0]
[0]
__________________________________________________________________________________
__________________
dense_3 (Dense)                 (None, 64)           8256        dropout[0]
[0]
__________________________________________________________________________________
__________________
dropout_1 (Dropout)             (None, 64)           0           dense_3[0]
[0]
__________________________________________________________________________________
__________________
dense_4 (Dense)                 (None, 64)           4160        dropout_1[0]
[0]
__________________________________________________________________________________
__________________
Output (Dense)                  (None, 2)            130         dense_4[0]
[0]
==================================================================================
====================
Total params: 2,898,776
Trainable params: 186,776
Non-trainable params: 2,712,000
__________________________________________________________________________________
__________________
```

In [16]:
```
def scheduler(epoch,lr):
    if((epoch+1)%3==0):
        lr=lr*0.95
        return lr
    else:
        return lr
```

In [17]:
```
filepath="model_save/weights-{epoch:02d}-{val_accuracy:.4f}.h5"
checkpoint = ModelCheckpoint(filepath=filepath, monitor='val_accuracy',mode='a
uto')



lrschedule = tf.keras.callbacks.LearningRateScheduler(scheduler,verbose=0.1)

#stop the training if your validation accuracy is not increased in last 2 epoc
hs.
early_stop= EarlyStopping(monitor='val_accuracy', patience=2,verbose=1)

#If your validation accuracy at that epoch is less than previous epoch accurac
y, you have to decrese the
#learning rate by 10%
reduce_lr = ReduceLROnPlateau(monitor='val_accuracy', factor=0.9,
                              patience=0, min_lr=0.001,verbose=1)

model.compile(
  loss='categorical_crossentropy',
  optimizer=tf.keras.optimizers.Adam(learning_rate=0.00001, beta_1=0.9, beta_2
=0.999, epsilon=1e-07, amsgrad=False),
  metrics=['accuracy',tf.keras.metrics.Precision(),tf.keras.metrics.Recall(),t
fa.metrics.F1Score(num_classes=2)]
)
```

In [21]:
```
y_train =tf.keras.utils.to_categorical(train_data['dank_level'].values,2)
y_test =tf.keras.utils.to_categorical(test_data['dank_level'].values,2)
y_test.shape
```

Out[21]:
```
(1719, 2)
```

In [22]:
```
history=model.fit({"words":padded_Xtrain_words,"subreddit_train_layer":subreddit_train,"is_nsfw_train_layer":is_nsfw_train,"time_of_day_train_layer":time_of_day_train,
                   "created_utc_train_layer":created_utc_train,"subscribers_train_layer":subscribers_train},
                    y_train,epochs=20,batch_size=30,
                  validation_data=({"words":padded_Xtest_words,"subreddit_train_layer":subreddit_test,"is_nsfw_train_layer":is_nsfw_test,"time_of_day_train_layer":time_of_day_test,
                   "created_utc_train_layer":created_utc_test,"subscribers_train_layer":subscribers_test},y_test),callbacks=[lrschedule,checkpoint,reduce_lr])
```

```
Epoch 1/20

Epoch 00001: LearningRateScheduler reducing learning rate to 9.99999974737875
2e-06.
114/114 [==============================] - 4s 33ms/step - loss: 0.7058 - accu
racy: 0.4996 - precision: 0.4996 - recall: 0.4996 - f1_score: 0.4580 - val_lo
ss: 0.6930 - val_accuracy: 0.5183 - val_precision: 0.5183 - val_recall: 0.518
3 - val_f1_score: 0.3839
Epoch 2/20

Epoch 00002: LearningRateScheduler reducing learning rate to 9.99999974737875
2e-06.
114/114 [==============================] - 3s 26ms/step - loss: 0.7045 - accu
racy: 0.5022 - precision: 0.5022 - recall: 0.5022 - f1_score: 0.4708 - val_lo
ss: 0.6923 - val_accuracy: 0.5265 - val_precision: 0.5265 - val_recall: 0.526
5 - val_f1_score: 0.4133
Epoch 3/20

Epoch 00003: LearningRateScheduler reducing learning rate to 9.49999976000981
3e-06.
114/114 [==============================] - 3s 27ms/step - loss: 0.7013 - accu
racy: 0.5063 - precision: 0.5063 - recall: 0.5063 - f1_score: 0.4829 - val_lo
ss: 0.6918 - val_accuracy: 0.5393 - val_precision: 0.5393 - val_recall: 0.539
3 - val_f1_score: 0.4486
Epoch 4/20

Epoch 00004: LearningRateScheduler reducing learning rate to 9.49999957811087
4e-06.
114/114 [==============================] - 3s 26ms/step - loss: 0.7001 - accu
racy: 0.4996 - precision: 0.4996 - recall: 0.4996 - f1_score: 0.4812 - val_lo
ss: 0.6915 - val_accuracy: 0.5398 - val_precision: 0.5398 - val_recall: 0.539
8 - val_f1_score: 0.4527
Epoch 5/20

Epoch 00005: LearningRateScheduler reducing learning rate to 9.49999957811087
4e-06.
114/114 [==============================] - 3s 26ms/step - loss: 0.6979 - accu
racy: 0.5110 - precision: 0.5110 - recall: 0.5110 - f1_score: 0.5002 - val_lo
ss: 0.6914 - val_accuracy: 0.5468 - val_precision: 0.5468 - val_recall: 0.546
8 - val_f1_score: 0.4760
Epoch 6/20

Epoch 00006: LearningRateScheduler reducing learning rate to 9.02499959920533
e-06.
114/114 [==============================] - 3s 26ms/step - loss: 0.6968 - accu
racy: 0.5034 - precision: 0.5034 - recall: 0.5034 - f1_score: 0.4930 - val_lo
ss: 0.6911 - val_accuracy: 0.5439 - val_precision: 0.5439 - val_recall: 0.543
9 - val_f1_score: 0.4750
Epoch 7/20

Epoch 00007: LearningRateScheduler reducing learning rate to 9.02499959920533
e-06.
114/114 [==============================] - 3s 26ms/step - loss: 0.6989 - accu
racy: 0.5013 - precision: 0.5013 - recall: 0.5013 - f1_score: 0.4937 - val_lo
ss: 0.6909 - val_accuracy: 0.5474 - val_precision: 0.5474 - val_recall: 0.547
4 - val_f1_score: 0.4769
Epoch 8/20
```

```
Epoch 00008: LearningRateScheduler reducing learning rate to 9.02499959920533
e-06.
114/114 [==============================] - 3s 27ms/step - loss: 0.6990 - accu
racy: 0.5010 - precision: 0.5010 - recall: 0.5010 - f1_score: 0.4932 - val_lo
ss: 0.6907 - val_accuracy: 0.5521 - val_precision: 0.5521 - val_recall: 0.552
1 - val_f1_score: 0.5214
Epoch 9/20

Epoch 00009: LearningRateScheduler reducing learning rate to 8.57374961924506
4e-06.
114/114 [==============================] - 3s 26ms/step - loss: 0.6961 - accu
racy: 0.5075 - precision: 0.5075 - recall: 0.5075 - f1_score: 0.4987 - val_lo
ss: 0.6906 - val_accuracy: 0.5590 - val_precision: 0.5590 - val_recall: 0.559
0 - val_f1_score: 0.5403
Epoch 10/20

Epoch 00010: LearningRateScheduler reducing learning rate to 8.57374925544718
3e-06.
114/114 [==============================] - 3s 28ms/step - loss: 0.6993 - accu
racy: 0.5075 - precision: 0.5075 - recall: 0.5075 - f1_score: 0.5015 - val_lo
ss: 0.6905 - val_accuracy: 0.5457 - val_precision: 0.5457 - val_recall: 0.545
7 - val_f1_score: 0.5340
Epoch 11/20

Epoch 00011: LearningRateScheduler reducing learning rate to 8.57374925544718
3e-06.
114/114 [==============================] - 3s 27ms/step - loss: 0.6994 - accu
racy: 0.4990 - precision: 0.4990 - recall: 0.4990 - f1_score: 0.4958 - val_lo
ss: 0.6905 - val_accuracy: 0.5480 - val_precision: 0.5480 - val_recall: 0.548
0 - val_f1_score: 0.5398
Epoch 12/20

Epoch 00012: LearningRateScheduler reducing learning rate to 8.14506179267482
4e-06.
114/114 [==============================] - 3s 26ms/step - loss: 0.6952 - accu
racy: 0.5095 - precision: 0.5095 - recall: 0.5095 - f1_score: 0.5067 - val_lo
ss: 0.6903 - val_accuracy: 0.5445 - val_precision: 0.5445 - val_recall: 0.544
5 - val_f1_score: 0.5414
Epoch 13/20

Epoch 00013: LearningRateScheduler reducing learning rate to 8.14506165625061
8e-06.
114/114 [==============================] - 3s 27ms/step - loss: 0.6958 - accu
racy: 0.5072 - precision: 0.5072 - recall: 0.5072 - f1_score: 0.5043 - val_lo
ss: 0.6903 - val_accuracy: 0.5439 - val_precision: 0.5439 - val_recall: 0.543
9 - val_f1_score: 0.5408
Epoch 14/20

Epoch 00014: LearningRateScheduler reducing learning rate to 8.14506165625061
8e-06.
114/114 [==============================] - 3s 26ms/step - loss: 0.6960 - accu
racy: 0.5189 - precision: 0.5189 - recall: 0.5189 - f1_score: 0.5154 - val_lo
ss: 0.6901 - val_accuracy: 0.5445 - val_precision: 0.5445 - val_recall: 0.544
5 - val_f1_score: 0.5415
Epoch 15/20
```

```
Epoch 00015: LearningRateScheduler reducing learning rate to 7.73780857343808
7e-06.
114/114 [==============================] - 3s 26ms/step - loss: 0.6954 - accu
racy: 0.5242 - precision: 0.5242 - recall: 0.5242 - f1_score: 0.5207 - val_lo
ss: 0.6901 - val_accuracy: 0.5404 - val_precision: 0.5404 - val_recall: 0.540
4 - val_f1_score: 0.5390
Epoch 16/20

Epoch 00016: LearningRateScheduler reducing learning rate to 7.73780811869073
7e-06.
114/114 [==============================] - 3s 26ms/step - loss: 0.6948 - accu
racy: 0.5192 - precision: 0.5192 - recall: 0.5192 - f1_score: 0.5176 - val_lo
ss: 0.6900 - val_accuracy: 0.5398 - val_precision: 0.5398 - val_recall: 0.539
8 - val_f1_score: 0.5385
Epoch 17/20

Epoch 00017: LearningRateScheduler reducing learning rate to 7.73780811869073
7e-06.
114/114 [==============================] - 3s 27ms/step - loss: 0.6948 - accu
racy: 0.5257 - precision: 0.5257 - recall: 0.5257 - f1_score: 0.5240 - val_lo
ss: 0.6899 - val_accuracy: 0.5381 - val_precision: 0.5381 - val_recall: 0.538
1 - val_f1_score: 0.5366
Epoch 18/20

Epoch 00018: LearningRateScheduler reducing learning rate to 7.3509177127562e
-06.
114/114 [==============================] - 3s 26ms/step - loss: 0.6927 - accu
racy: 0.5101 - precision: 0.5101 - recall: 0.5101 - f1_score: 0.5087 - val_lo
ss: 0.6899 - val_accuracy: 0.5393 - val_precision: 0.5393 - val_recall: 0.539
3 - val_f1_score: 0.5378
Epoch 19/20

Epoch 00019: LearningRateScheduler reducing learning rate to 7.3509177127562e
-06.
114/114 [==============================] - 3s 27ms/step - loss: 0.6958 - accu
racy: 0.5148 - precision: 0.5148 - recall: 0.5148 - f1_score: 0.5130 - val_lo
ss: 0.6898 - val_accuracy: 0.5398 - val_precision: 0.5398 - val_recall: 0.539
8 - val_f1_score: 0.5384
Epoch 20/20

Epoch 00020: LearningRateScheduler reducing learning rate to 7.3509177127562e
-06.
114/114 [==============================] - 3s 27ms/step - loss: 0.6877 - accu
racy: 0.5392 - precision: 0.5392 - recall: 0.5392 - f1_score: 0.5374 - val_lo
ss: 0.6897 - val_accuracy: 0.5393 - val_precision: 0.5393 - val_recall: 0.539
3 - val_f1_score: 0.5377
```

In [24]:

```
model_checkpoint = Model(inputs=[words,subreddit_train_layer,is_nsfw_train_lay
er,time_of_day_train_layer,created_utc_train_layer,subscribers_train_layer],ou
tputs=Out)
model_checkpoint.load_weights('/content/model_save/weights-09-0.5590.h5')
model_checkpoint.save('bestmodel_lstm.h5')
new_model = tf.keras.models.load_model('bestmodel_lstm.h5')
```

```
WARNING:tensorflow:No training configuration found in the save file, so the m
odel was *not* compiled. Compile it manually.
```

In [28]:
```
test_prediction=new_model.predict([padded_Xtest_words,subreddit_test,is_nsfw_test,time_of_day_test,
                     created_utc_test,subscribers_test])
test_prediction=np.argmax(test_prediction,axis=-1)
print(test_prediction.shape)
y_test =tf.keras.utils.to_categorical(test_data['dank_level'].values,2)
y_test=np.argmax(y_test,axis=-1)
y_test.shape
```

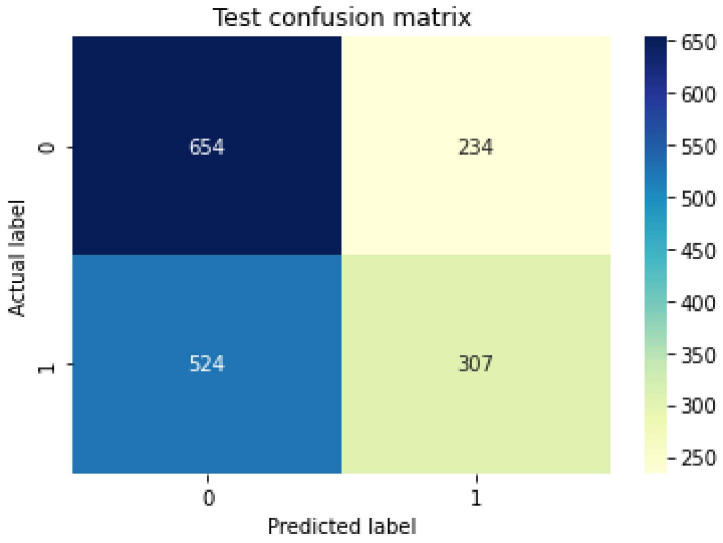
```
(1719,)
```

Out[28]:
```
(1719,)
```

In [29]:
```
accuracy=accuracy_score(y_test,test_prediction)
print("Test accuracy_score",accuracy)
f1_test_score=f1_score(y_test,test_prediction)
print("Test F1_score",f1_test_score)
print("Test confusion matrix")
cnf_matrix2=confusion_matrix(y_test,test_prediction)
p = sns.heatmap(pd.DataFrame(cnf_matrix2), annot=True, cmap="YlGnBu" ,fmt='g')
plt.title('Test confusion matrix', y=1.1)
plt.ylabel('Actual label')
plt.xlabel('Predicted label')
```

```
Test accuracy_score 0.5590459569517161
Test F1_score 0.44752186588921283
Test confusion matrix
```

Out[29]:
```
Text(0.5, 15.0, 'Predicted label')
```



In [27]:
```
file = '/content/model_1.png'
tf.keras.utils.plot_model(model,to_file=file, show_shapes=True)
```

Out[27]: