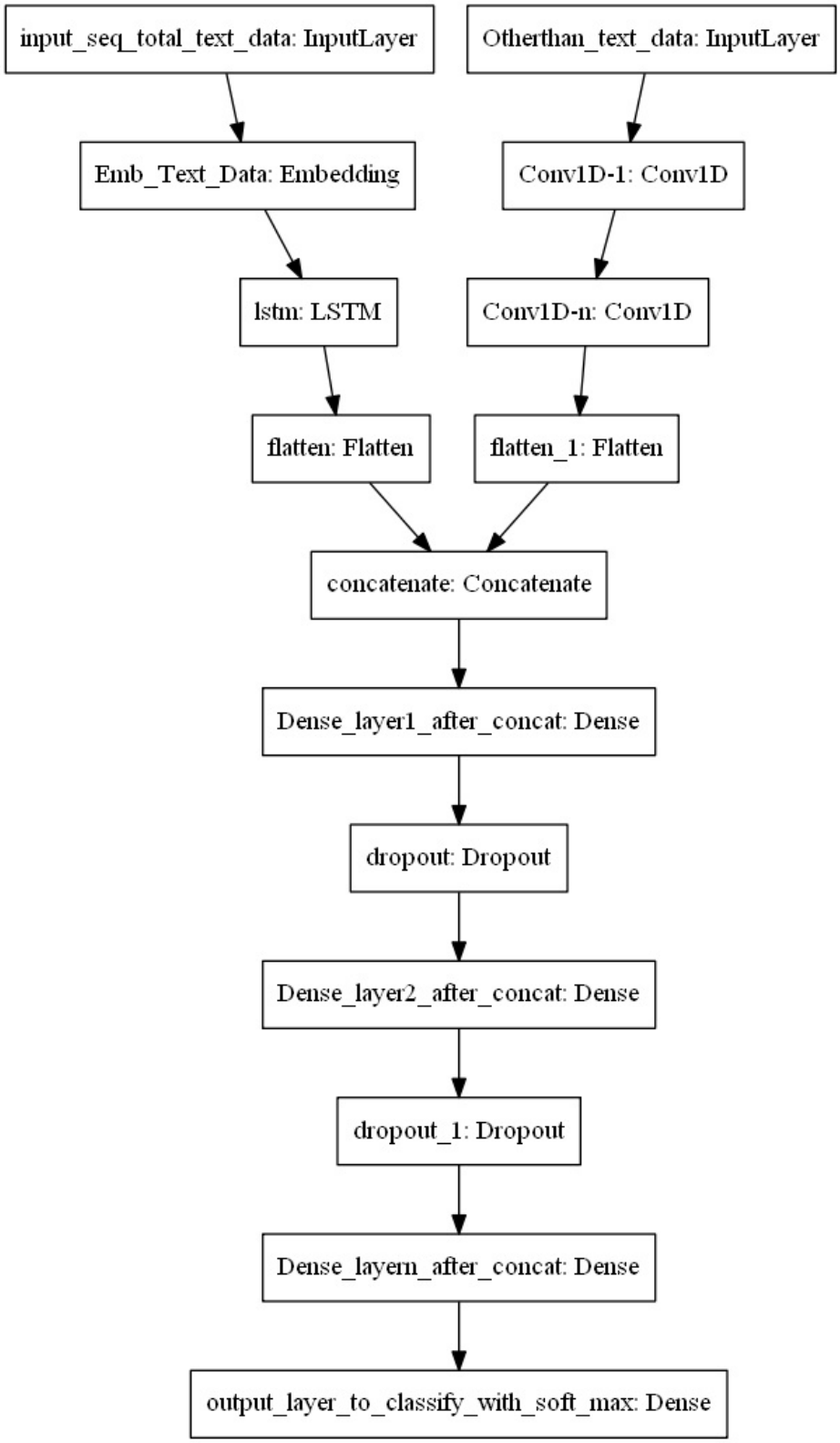
Model-3



ref: https://i.imgur.com/fkQ8nGo.png

In [1]:
```
#in this model you can use the text vectorized data from model1
#for other than text data consider the following steps
# you have to perform one hot encoding of categorical features. You can use onehotencoder() or countvectorizer(
# Stack up standardised numerical features and all the one hot encoded categorical features
#the input to conv1d layer is 3d, you can convert your 2d data to 3d using np.newaxis
# Note - deep learning models won't work with sparse features, you have to convert them to dense features befor
```

In [2]:
```
import tensorflow
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Input
from tensorflow.keras.layers import LSTM
from tensorflow.keras.layers import Embedding
from tensorflow.keras import regularizers
from tensorflow.keras.regularizers import l2
```

```
from tensorflow.keras.layers import Flatten
from tensorflow.keras.layers import Dense, Input , Dropout
from tensorflow.keras.layers import concatenate
from tensorflow.keras.layers import BatchNormalization
from tensorflow.keras.callbacks import TensorBoard
import tensorflow as tf
from sklearn.metrics import roc_auc_score
from tensorflow.keras.metrics import AUC

from sklearn import preprocessing
import numpy as np
from sklearn.preprocessing import OrdinalEncoder
import numpy as np
from scipy.sparse import coo_matrix, hstack
from scipy.sparse import csr_matrix
from numpy import asarray
from numpy import zeros
from keras.preprocessing.text import Tokenizer
from keras_preprocessing.sequence import pad_sequences
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import Flatten
from keras.layers import Embedding
from tqdm import tqdm
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
from sklearn.model_selection import train_test_split
from keras.utils import np_utils
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.feature_extraction.text import CountVectorizer
import warnings
warnings.filterwarnings("ignore")
tf.keras.backend.clear_session()
```

In [ ]:

In [ ]:

In [3]:
```
#read the csv file
import pandas as pd
p1 = '/content/drive/MyDrive/AAIC/Assignments/LSTM on Donors Choose/preprocessed_data_final.csv'
p2 = "C:/Users/darsh/Downloads/Srujan/Donars Choose Assignment/preprocessed_data_final.csv"
df = pd.read_csv(p2)
```

In [4]:
```
 df[pd.isnull(df).any(axis=1)]
```

Out[4]:

	teacher_number_of_previously_posted_projects	resource_summary_contains_numerical_digits	price	quantity	school_state	project_grade_cat

In [5]:
```
y = df['project_is_approved'].values
df.drop(['project_is_approved'],axis=1,inplace=True)
```

In [6]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(df, y,
                                                    stratify=y,
                                                    test_size=0.25,random_state=0)
```

In [ ]:

In [7]:
```
X_train.head(2)
```

Out[7]:

	teacher_number_of_previously_posted_projects	resource_summary_contains_numerical_digits	price	quantity	school_state	project_gra
92061	6	0	234.95	4	nc	
83229	47	0	324.06	6	nc	

In [ ]:

Text vectorization

In [8]:
```
text_input = ['essay','project_title','project_resource_summary',]
X_train['total_text_input'] = X_train['essay'] + ' ' + X_train['project_title'] + ' ' + X_train['project_resour
X_test['total_text_input'] = X_test['essay'] + ' ' + X_test['project_title'] + ' ' + X_test['project_resource_s
```

In [9]:
```
num_words = 1000
oov_token = '<UNK>'
pad_type = 'post'
trunc_type = 'post'

# Tokenize our training data
tokenizer = Tokenizer(num_words=num_words, oov_token=oov_token)
tokenizer.fit_on_texts(X_train['total_text_input'])

# Get our training data word index
word_index = tokenizer.word_index

# Encode training data sentences into sequences
train_sequences = tokenizer.texts_to_sequences(X_train['total_text_input'])
test_sequences = tokenizer.texts_to_sequences(X_test['total_text_input'])

# Get max training sequence length
maxlen = max([len(x) for x in train_sequences])

# Pad the training sequences
train_padded = pad_sequences(train_sequences, padding=pad_type, truncating=trunc_type, maxlen=maxlen)
test_padded = pad_sequences(test_sequences, padding=pad_type, truncating=trunc_type, maxlen=maxlen)
```

In [10]:
```
# Output the results of our work
#print("Word index:\n", word_index)
#print("\nTraining sequences:\n", train_sequences)
#print("\nPadded training sequences:\n", train_padded)
print("\nPadded training shape, Test Shape:", train_padded.shape,test_padded.shape)
print("Training sequences data type:", type(train_sequences),type(test_sequences))
print("Padded Training sequences data type:", type(train_padded),type(test_padded))
```

```
Padded training shape, Test Shape: (81936, 355) (27312, 355)
Training sequences data type: <class 'list'> <class 'list'>
Padded Training sequences data type: <class 'numpy.ndarray'> <class 'numpy.ndarray'>
```

In [ ]:

In [11]:
```
# load the whole embedding into memory
embeddings_index = dict()
p1 = '/content/drive/MyDrive/AAIC/Assignments/LSTM on Donors Choose/glove.6B.300d.txt'
p2 = "C:/Users/darsh/Downloads/Srujan/Donars Choose Assignment/glove.6B.300d.txt"
f = open(p2,encoding="utf8")
for line in tqdm(f):
    values = line.split()
    word = values[0]
    coefs = asarray(values[1:], dtype='float32')
    embeddings_index[word] = coefs
f.close()
print('\nLoaded %s word vectors.' % len(embeddings_index))
```

```
400000it [00:19, 20102.10it/s]
Loaded 400000 word vectors.
```

In [12]:
```
vocab_size = len(tokenizer.word_index) + 1
```

In [13]:
```
# create a weight matrix for words in training docs
embedding_matrix = zeros((vocab_size, 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 [ ]:

categorical and Numerical vectorization

In [14]:
```
#in this model you can use the text vectorized data from model1
#for other than text data consider the following steps
# you have to perform one hot encoding of categorical features. You can use onehotencoder() or countvectorizer(
# Stack up standardised numerical features and all the one hot encoded categorical features
#the input to conv1d layer is 3d, you can convert your 2d data to 3d using np.newaxis
# Note - deep learning models won't work with sparse features, you have to convert them to dense features befor
```

In [15]:
```
categorical_input = ['school_state','project_grade_category','clean_categories', 'clean_subcategories','teacher_
```

In [ ]:

In [16]:
```
categorical_data_train = dict()
categorical_data_test = dict()
for i in categorical_input:
    vectorizer = CountVectorizer()
    train_enc = vectorizer.fit_transform(X_train[i].values)
    test_enc = vectorizer.transform(X_test[i].values)
    categorical_data_train[i+'_enc'] = train_enc#.todense()
    categorical_data_test[i+'_test_enc'] = test_enc#.todense()
    print(i,train_enc.shape,test_enc.shape)
```

```
school_state (81936, 51) (27312, 51)
project_grade_category (81936, 4) (27312, 4)
clean_categories (81936, 51) (27312, 51)
clean_subcategories (81936, 393) (27312, 393)
teacher_prefix (81936, 5) (27312, 5)
```

In [ ]:

In [17]:
```
numerical_input = ['teacher_number_of_previously_posted_projects',
                   'resource_summary_contains_numerical_digits',
                   'price','quantity']
```

In [ ]:

In [18]:
```
scaler  = preprocessing.MinMaxScaler().fit(X_train[numerical_input].values)
std_data_train = (pd.DataFrame(scaler.transform(X_train[numerical_input]),columns=numerical_input))
std_data_test = (pd.DataFrame(scaler.transform(X_test[numerical_input]),columns=numerical_input))
```

In [ ]:

In [19]:
```
other_than_text_data = hstack(((std_data_train),
                               categorical_data_train['school_state_enc'],
                               categorical_data_train['project_grade_category_enc'],
                               categorical_data_train['clean_categories_enc'],
                               categorical_data_train['clean_subcategories_enc'],
                               categorical_data_train['teacher_prefix_enc'],
                               )).todense()
```

In [20]:
```
#np.array(other_than_text_data).reshape(81936,507,1)
#other_than_text_data = other_than_text_data.reshape(81936,507,1)
```

In [21]:
```
other_than_text_data = other_than_text_data[:,:,np.newaxis]
```

In [22]:
```
type(other_than_text_data.reshape(81936,508,1))
```

Out[22]:
```
numpy.matrix
```

In [23]:
```
other_than_text_data_test = np.array(hstack(((std_data_test),
                                    categorical_data_test['school_state_test_enc'],
                                    categorical_data_test['project_grade_category_test_enc'],
                                    categorical_data_test['clean_categories_test_enc'],
                                    categorical_data_test['clean_subcategories_test_enc'],
                                    categorical_data_test['teacher_prefix_test_enc'],
                                   )).todense())
```

In [24]:
```
other_than_text_data_test = other_than_text_data_test[:,:,np.newaxis]
```

In [ ]:

In [25]:
```
from tensorflow.keras.layers import Conv1D

input_seq_total_text_data = Input(shape=(maxlen,),name='input_seq_total_text_data_')

emb_text_data = Embedding(input_dim=vocab_size,output_dim=300,
                          weights=[embedding_matrix], input_length=maxlen,trainable=False,
                          name='emb_text_data')(input_seq_total_text_data)
lstm = LSTM(units=128,activation='tanh',return_sequences=True)(emb_text_data)
flatten_text = Flatten()(lstm)

other_than_text_data_ = Input(shape=(508,1))

#other_than_text_data_ = Input(shape=(508,1),name='other_than_text_data1')
conv1 = Conv1D(filters=128,kernel_size=3,strides=1,)(other_than_text_data_)
```

```
conv2 = Conv1D(filters=128,kernel_size=3,strides=1,)(conv1)
flat1 = Flatten()(conv2)

concat = concatenate([flatten_text,flat1])

d1 = Dense(units=256,activation='relu',kernel_initializer='he_normal',
           kernel_regularizer=l2(0.000001),name='dense_layer_1')(concat)
drop1 = Dropout(0.5)(d1)

d2 = Dense(units=256,activation='relu',kernel_initializer='he_normal',
           kernel_regularizer=l2(0.000001),name='dense_layer_2')(drop1)
drop2 = Dropout(0.5)(d2)

bn1 = BatchNormalization()(drop2)

d3 = Dense(units=256,activation='relu',kernel_initializer='he_normal',
           kernel_regularizer=l2(0.000001),name='dense_layer_3')(bn1)
drop3 = Dropout(0.5)(d3)

d4 = Dense(units=256,activation='relu',kernel_initializer='he_normal',
           kernel_regularizer=l2(0.000001),name='dense_layer_4')(drop3)
drop4 = Dropout(0.5)(d4)



d5 = Dense(units=256,activation='relu',kernel_initializer='he_normal',
           kernel_regularizer=l2(0.000001),name='dense_layer_5')(drop4)
drop5 = Dropout(0.5)(d5)


output = Dense(units=2,activation='softmax')(drop5)
```

In [ ]:

In [26]:
```
m3 = Model(inputs=[input_seq_total_text_data,other_than_text_data_],
           outputs=[output])
```

In [27]:
```
m3.summary()
```

```
Model: "model"
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
input_seq_total_text_data_ (Inp [(None, 355)]        0
__________________________________________________________________________________________________
input_1 (InputLayer)            [(None, 508, 1)]     0
__________________________________________________________________________________________________
emb_text_data (Embedding)       (None, 355, 300)     17031600    input_seq_total_text_data_[0][0]
__________________________________________________________________________________________________
conv1d (Conv1D)                 (None, 506, 128)     512         input_1[0][0]
__________________________________________________________________________________________________
lstm (LSTM)                     (None, 355, 128)     219648      emb_text_data[0][0]
__________________________________________________________________________________________________
conv1d_1 (Conv1D)               (None, 504, 128)     49280       conv1d[0][0]
__________________________________________________________________________________________________
flatten (Flatten)               (None, 45440)        0           lstm[0][0]
__________________________________________________________________________________________________
flatten_1 (Flatten)             (None, 64512)        0           conv1d_1[0][0]
__________________________________________________________________________________________________
concatenate (Concatenate)       (None, 109952)       0           flatten[0][0]
                                                                 flatten_1[0][0]
__________________________________________________________________________________________________
dense_layer_1 (Dense)           (None, 256)          28147968    concatenate[0][0]
__________________________________________________________________________________________________
dropout (Dropout)               (None, 256)          0           dense_layer_1[0][0]
__________________________________________________________________________________________________
dense_layer_2 (Dense)           (None, 256)          65792       dropout[0][0]
__________________________________________________________________________________________________
dropout_1 (Dropout)             (None, 256)          0           dense_layer_2[0][0]
__________________________________________________________________________________________________
batch_normalization (BatchNorma (None, 256)          1024        dropout_1[0][0]
__________________________________________________________________________________________________
dense_layer_3 (Dense)           (None, 256)          65792       batch_normalization[0][0]
__________________________________________________________________________________________________
dropout_2 (Dropout)             (None, 256)          0           dense_layer_3[0][0]
__________________________________________________________________________________________________
dense_layer_4 (Dense)           (None, 256)          65792       dropout_2[0][0]
__________________________________________________________________________________________________
dropout_3 (Dropout)             (None, 256)          0           dense_layer_4[0][0]
__________________________________________________________________________________________________
dense_layer_5 (Dense)           (None, 256)          65792       dropout_3[0][0]
__________________________________________________________________________________________________
dropout_4 (Dropout)             (None, 256)          0           dense_layer_5[0][0]
__________________________________________________________________________________________________
dense (Dense)                   (None, 2)            514         dropout_4[0][0]
==================================================================================================
Total params: 45,713,714
Trainable params: 28,681,602
Non-trainable params: 17,032,112
__________________________________________________________________________________________________
```

In [ ]:

In [28]:
```
test_data = [test_padded,other_than_text_data_test]

train_data = [train_padded,other_than_text_data]

y_train_enc =  tensorflow.keras.utils.to_categorical(y_train, 2)
y_test_enc =  tensorflow.keras.utils.to_categorical(y_test, 2)
```

In [29]:
```
def auc1(y_true, y_pred):
    if len(np.unique(y_true[:,1])) == 1:
        return 0.5
    else:
        return roc_auc_score( y_true, y_pred, average='macro', sample_weight=None).astype('double')

def auroc(y_true, y_pred):
    return tensorflow.numpy_function(auc1, (y_true, y_pred), tensorflow.double)

callbacks = [
    tf.keras.callbacks.ModelCheckpoint('./LSTM_Model_3.h5', save_weights_only=False,save_best_only=True, \
                                       mode='max', monitor='val_auroc',verbose=1),
    tf.keras.callbacks.ReduceLROnPlateau(monitor='val_auroc', patience=2,mode='max',verbose=1),
]
```

In [30]:
```
m3.compile(optimizer='adam', loss='categorical_crossentropy', metrics=[auroc])
```

In [31]:
```
steps = len(y_train_enc)//128
```

In [ ]:

In [32]:
```
m3.fit(train_data,y_train_enc,
       validation_data=(test_data,y_test_enc),
       batch_size=128,
       epochs=50,
       callbacks=callbacks,
       verbose=1)
```

```
Epoch 1/50
641/641 [==============================] - 54s 69ms/step - loss: 0.5118 - auroc: 0.5022 - val_loss: 0.5550 - va
l_auroc: 0.4308

Epoch 00001: val_auroc improved from -inf to 0.43082, saving model to .\LSTM_Model_3.h5
Epoch 2/50
641/641 [==============================] - 43s 67ms/step - loss: 0.4400 - auroc: 0.5046 - val_loss: 0.5034 - va
l_auroc: 0.5878

Epoch 00002: val_auroc improved from 0.43082 to 0.58782, saving model to .\LSTM_Model_3.h5
Epoch 3/50
641/641 [==============================] - 43s 68ms/step - loss: 0.4362 - auroc: 0.5027 - val_loss: 0.4607 - va
l_auroc: 0.4182

Epoch 00003: val_auroc did not improve from 0.58782
Epoch 4/50
641/641 [==============================] - 43s 67ms/step - loss: 0.4340 - auroc: 0.4999 - val_loss: 0.4464 - va
l_auroc: 0.4063

Epoch 00004: val_auroc did not improve from 0.58782

Epoch 00004: ReduceLROnPlateau reducing learning rate to 0.00010000000474974513.
Epoch 5/50
641/641 [==============================] - 44s 68ms/step - loss: 0.4306 - auroc: 0.5147 - val_loss: 0.4420 - va
l_auroc: 0.5728

Epoch 00005: val_auroc did not improve from 0.58782
Epoch 6/50
641/641 [==============================] - 44s 68ms/step - loss: 0.4251 - auroc: 0.5709 - val_loss: 0.4403 - va
l_auroc: 0.5917

Epoch 00006: val_auroc improved from 0.58782 to 0.59168, saving model to .\LSTM_Model_3.h5
Epoch 7/50
641/641 [==============================] - 43s 68ms/step - loss: 0.4183 - auroc: 0.6282 - val_loss: 0.4433 - va
l_auroc: 0.6401

Epoch 00007: val_auroc improved from 0.59168 to 0.64011, saving model to .\LSTM_Model_3.h5
Epoch 8/50
641/641 [==============================] - 44s 68ms/step - loss: 0.4132 - auroc: 0.6569 - val_loss: 0.4341 - va
l_auroc: 0.6761

Epoch 00008: val_auroc improved from 0.64011 to 0.67609, saving model to .\LSTM_Model_3.h5
Epoch 9/50
641/641 [==============================] - 44s 68ms/step - loss: 0.4075 - auroc: 0.6781 - val_loss: 0.4418 - va
l_auroc: 0.6946

Epoch 00009: val_auroc improved from 0.67609 to 0.69463, saving model to .\LSTM_Model_3.h5
Epoch 10/50
641/641 [==============================] - 44s 68ms/step - loss: 0.3993 - auroc: 0.6966 - val_loss: 0.4381 - va
l_auroc: 0.7081

Epoch 00010: val_auroc improved from 0.69463 to 0.70806, saving model to .\LSTM_Model_3.h5
Epoch 11/50
641/641 [==============================] - 44s 68ms/step - loss: 0.3950 - auroc: 0.7085 - val_loss: 0.4498 - va
l_auroc: 0.7127

Epoch 00011: val_auroc improved from 0.70806 to 0.71275, saving model to .\LSTM_Model_3.h5
Epoch 12/50
641/641 [==============================] - 43s 68ms/step - loss: 0.3919 - auroc: 0.7163 - val_loss: 0.4535 - va
l_auroc: 0.7213

Epoch 00012: val_auroc improved from 0.71275 to 0.72134, saving model to .\LSTM_Model_3.h5
Epoch 13/50
641/641 [==============================] - 43s 68ms/step - loss: 0.3875 - auroc: 0.7243 - val_loss: 0.4437 - va
l_auroc: 0.7282

Epoch 00013: val_auroc improved from 0.72134 to 0.72824, saving model to .\LSTM_Model_3.h5
Epoch 14/50
641/641 [==============================] - 43s 68ms/step - loss: 0.3843 - auroc: 0.7325 - val_loss: 0.4197 - va
l_auroc: 0.7323

Epoch 00014: val_auroc improved from 0.72824 to 0.73231, saving model to .\LSTM_Model_3.h5
Epoch 15/50
641/641 [==============================] - 43s 68ms/step - loss: 0.3815 - auroc: 0.7380 - val_loss: 0.4375 - va
l_auroc: 0.7352

Epoch 00015: val_auroc improved from 0.73231 to 0.73521, saving model to .\LSTM_Model_3.h5
```

```
Epoch 16/50
641/641 [==============================] - 43s 68ms/step - loss: 0.3775 - auroc: 0.7447 - val_loss: 0.4171 - va
l_auroc: 0.7378

Epoch 00016: val_auroc improved from 0.73521 to 0.73778, saving model to .\LSTM_Model_3.h5
Epoch 17/50
641/641 [==============================] - 43s 68ms/step - loss: 0.3753 - auroc: 0.7513 - val_loss: 0.4349 - va
l_auroc: 0.7427

Epoch 00017: val_auroc improved from 0.73778 to 0.74266, saving model to .\LSTM_Model_3.h5
Epoch 18/50
641/641 [==============================] - 43s 68ms/step - loss: 0.3718 - auroc: 0.7564 - val_loss: 0.4117 - va
l_auroc: 0.7440

Epoch 00018: val_auroc improved from 0.74266 to 0.74399, saving model to .\LSTM_Model_3.h5
Epoch 19/50
641/641 [==============================] - 43s 68ms/step - loss: 0.3686 - auroc: 0.7616 - val_loss: 0.4129 - va
l_auroc: 0.7466

Epoch 00019: val_auroc improved from 0.74399 to 0.74662, saving model to .\LSTM_Model_3.h5
Epoch 20/50
641/641 [==============================] - 43s 68ms/step - loss: 0.3658 - auroc: 0.7666 - val_loss: 0.4286 - va
l_auroc: 0.7476

Epoch 00020: val_auroc improved from 0.74662 to 0.74756, saving model to .\LSTM_Model_3.h5
Epoch 21/50
641/641 [==============================] - 43s 68ms/step - loss: 0.3629 - auroc: 0.7724 - val_loss: 0.4080 - va
l_auroc: 0.7434

Epoch 00021: val_auroc did not improve from 0.74756
Epoch 22/50
641/641 [==============================] - 43s 68ms/step - loss: 0.3589 - auroc: 0.7773 - val_loss: 0.4190 - va
l_auroc: 0.7493

Epoch 00022: val_auroc improved from 0.74756 to 0.74928, saving model to .\LSTM_Model_3.h5
Epoch 23/50
641/641 [==============================] - 43s 68ms/step - loss: 0.3544 - auroc: 0.7843 - val_loss: 0.4033 - va
l_auroc: 0.7441

Epoch 00023: val_auroc did not improve from 0.74928
Epoch 24/50
641/641 [==============================] - 43s 68ms/step - loss: 0.3512 - auroc: 0.7898 - val_loss: 0.4430 - va
l_auroc: 0.7459

Epoch 00024: val_auroc did not improve from 0.74928

Epoch 00024: ReduceLROnPlateau reducing learning rate to 1.0000000474974514e-05.
Epoch 25/50
641/641 [==============================] - 44s 68ms/step - loss: 0.3407 - auroc: 0.8015 - val_loss: 0.4239 - va
l_auroc: 0.7467

Epoch 00025: val_auroc did not improve from 0.74928
Epoch 26/50
641/641 [==============================] - 43s 68ms/step - loss: 0.3372 - auroc: 0.8038 - val_loss: 0.4274 - va
l_auroc: 0.7443

Epoch 00026: val_auroc did not improve from 0.74928

Epoch 00026: ReduceLROnPlateau reducing learning rate to 1.0000000656873453e-06.
Epoch 27/50
641/641 [==============================] - 43s 68ms/step - loss: 0.3342 - auroc: 0.8086 - val_loss: 0.4291 - va
l_auroc: 0.7443

Epoch 00027: val_auroc did not improve from 0.74928
Epoch 28/50
641/641 [==============================] - 43s 68ms/step - loss: 0.3348 - auroc: 0.8081 - val_loss: 0.4284 - va
l_auroc: 0.7446

Epoch 00028: val_auroc did not improve from 0.74928

Epoch 00028: ReduceLROnPlateau reducing learning rate to 1.0000001111620805e-07.
Epoch 29/50
641/641 [==============================] - 43s 68ms/step - loss: 0.3333 - auroc: 0.8092 - val_loss: 0.4299 - va
l_auroc: 0.7446

Epoch 00029: val_auroc did not improve from 0.74928
Epoch 30/50
641/641 [==============================] - 43s 68ms/step - loss: 0.3349 - auroc: 0.8074 - val_loss: 0.4292 - va
l_auroc: 0.7445

Epoch 00030: val_auroc did not improve from 0.74928

Epoch 00030: ReduceLROnPlateau reducing learning rate to 1.000000082740371e-08.
```

```
Epoch 31/50
641/641 [==============================] - 43s 68ms/step - loss: 0.3346 - auroc: 0.8080 - val_loss: 0.4290 - va
l_auroc: 0.7445

Epoch 00031: val_auroc did not improve from 0.74928
Epoch 32/50
641/641 [==============================] - 43s 68ms/step - loss: 0.3329 - auroc: 0.8097 - val_loss: 0.4288 - va
l_auroc: 0.7446

Epoch 00032: val_auroc did not improve from 0.74928

Epoch 00032: ReduceLROnPlateau reducing learning rate to 1.000000082740371e-09.
Epoch 33/50
641/641 [==============================] - 44s 68ms/step - loss: 0.3347 - auroc: 0.8074 - val_loss: 0.4291 - va
l_auroc: 0.7446

Epoch 00033: val_auroc did not improve from 0.74928
Epoch 34/50
641/641 [==============================] - 43s 68ms/step - loss: 0.3337 - auroc: 0.8100 - val_loss: 0.4297 - va
l_auroc: 0.7445

Epoch 00034: val_auroc did not improve from 0.74928

Epoch 00034: ReduceLROnPlateau reducing learning rate to 1.000000082740371e-10.
Epoch 35/50
641/641 [==============================] - 43s 68ms/step - loss: 0.3355 - auroc: 0.8080 - val_loss: 0.4288 - va
l_auroc: 0.7445

Epoch 00035: val_auroc did not improve from 0.74928
Epoch 36/50
641/641 [==============================] - 43s 67ms/step - loss: 0.3341 - auroc: 0.8073 - val_loss: 0.4295 - va
l_auroc: 0.7445

Epoch 00036: val_auroc did not improve from 0.74928

Epoch 00036: ReduceLROnPlateau reducing learning rate to 1.000000082740371e-11.
Epoch 37/50
641/641 [==============================] - 44s 68ms/step - loss: 0.3349 - auroc: 0.8094 - val_loss: 0.4296 - va
l_auroc: 0.7445

Epoch 00037: val_auroc did not improve from 0.74928
Epoch 38/50
641/641 [==============================] - 43s 68ms/step - loss: 0.3341 - auroc: 0.8076 - val_loss: 0.4291 - va
l_auroc: 0.7445

Epoch 00038: val_auroc did not improve from 0.74928

Epoch 00038: ReduceLROnPlateau reducing learning rate to 1.000000082740371e-12.
Epoch 39/50
641/641 [==============================] - 43s 68ms/step - loss: 0.3349 - auroc: 0.8086 - val_loss: 0.4289 - va
l_auroc: 0.7445

Epoch 00039: val_auroc did not improve from 0.74928
Epoch 40/50
641/641 [==============================] - 43s 68ms/step - loss: 0.3333 - auroc: 0.8119 - val_loss: 0.4291 - va
l_auroc: 0.7445

Epoch 00040: val_auroc did not improve from 0.74928

Epoch 00040: ReduceLROnPlateau reducing learning rate to 1.0000001044244145e-13.
Epoch 41/50
641/641 [==============================] - 43s 68ms/step - loss: 0.3345 - auroc: 0.8102 - val_loss: 0.4288 - va
l_auroc: 0.7444

Epoch 00041: val_auroc did not improve from 0.74928
Epoch 42/50
641/641 [==============================] - 43s 68ms/step - loss: 0.3342 - auroc: 0.8097 - val_loss: 0.4296 - va
l_auroc: 0.7445

Epoch 00042: val_auroc did not improve from 0.74928

Epoch 00042: ReduceLROnPlateau reducing learning rate to 1.0000001179769417e-14.
Epoch 43/50
641/641 [==============================] - 44s 68ms/step - loss: 0.3350 - auroc: 0.8081 - val_loss: 0.4298 - va
l_auroc: 0.7446

Epoch 00043: val_auroc did not improve from 0.74928
Epoch 44/50
641/641 [==============================] - 44s 68ms/step - loss: 0.3350 - auroc: 0.8082 - val_loss: 0.4289 - va
l_auroc: 0.7445

Epoch 00044: val_auroc did not improve from 0.74928
```

```
Epoch 00044: ReduceLROnPlateau reducing learning rate to 1.0000001518582595e-15.
Epoch 45/50
641/641 [==============================] - 43s 68ms/step - loss: 0.3355 - auroc: 0.8080 - val_loss: 0.4291 - va
l_auroc: 0.7445

Epoch 00045: val_auroc did not improve from 0.74928
Epoch 46/50
641/641 [==============================] - 43s 68ms/step - loss: 0.3337 - auroc: 0.8098 - val_loss: 0.4285 - va
l_auroc: 0.7445

Epoch 00046: val_auroc did not improve from 0.74928

Epoch 00046: ReduceLROnPlateau reducing learning rate to 1.0000001095066122e-16.
Epoch 47/50
641/641 [==============================] - 43s 68ms/step - loss: 0.3357 - auroc: 0.8068 - val_loss: 0.4290 - va
l_auroc: 0.7445

Epoch 00047: val_auroc did not improve from 0.74928
Epoch 48/50
641/641 [==============================] - 44s 68ms/step - loss: 0.3340 - auroc: 0.8106 - val_loss: 0.4291 - va
l_auroc: 0.7445

Epoch 00048: val_auroc did not improve from 0.74928

Epoch 00048: ReduceLROnPlateau reducing learning rate to 1.0000000830368326e-17.
Epoch 49/50
641/641 [==============================] - 43s 68ms/step - loss: 0.3348 - auroc: 0.8093 - val_loss: 0.4294 - va
l_auroc: 0.7444

Epoch 00049: val_auroc did not improve from 0.74928
Epoch 50/50
641/641 [==============================] - 44s 68ms/step - loss: 0.3348 - auroc: 0.8088 - val_loss: 0.4288 - va
l_auroc: 0.7446

Epoch 00050: val_auroc did not improve from 0.74928

Epoch 00050: ReduceLROnPlateau reducing learning rate to 1.0000000664932204e-18.
```

Out[32]: <keras.callbacks.History at 0x20c6f49d340>

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: