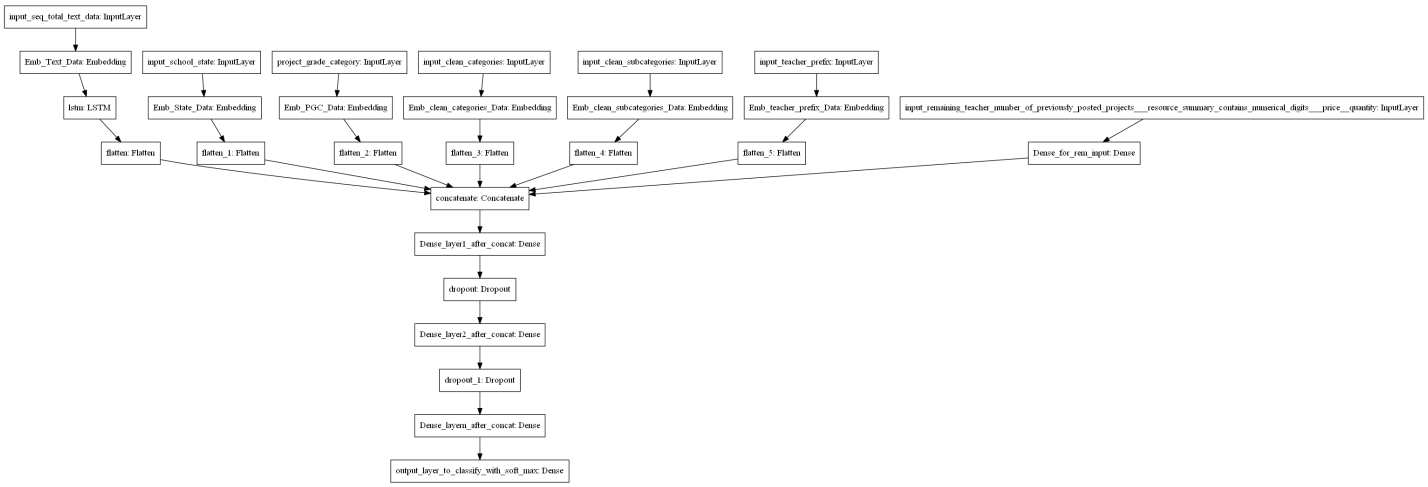
Assignment : 14

```
1. Preprocess data from DonorsChoose Dataset from train.csv
2. Combine 4 essay's into one column named - 'preprocessed_essays'.
3. After step 2 you have to train 3 types of models as discussed below.
4. For all the model use 'auc' as a metric. check this for using auc as a metric
5. You are free to choose any number of layers/hiddden units but you have to use same type of architectures shown below.
6. You can use any one of the optimizers and choice of Learning rate and momentum, resources: cs231n class notes, cs231n class vide
7. For all the model's use TensorBoard and plot the Metric value and Loss with epoch. While submitting, take a screenshot of plots
8. Use Categorical Cross Entropy as Loss to minimize.
```

Model-1

Build and Train deep neural network as shown below



ref: https://i.imgur.com/w395Yk9.png

- **Input_seq_total_text_data** --- You have to give Total text data columns. After this use the Embedding layer to get word vectors. Use given predefined glove word vectors, don't train any word vectors. After this use LSTM and get the LSTM output and Flatten that output.
- **Input_school_state** --- Give 'school_state' column as input to embedding layer and Train the Keras Embedding layer.
- **Project_grade_category** --- Give 'project_grade_category' column as input to embedding layer and Train the Keras Embedding layer.
- **Input_clean_categories** --- Give 'input_clean_categories' column as input to embedding layer and Train the Keras Embedding layer.
- **Input_clean_subcategories** --- Give 'input_clean_subcategories' column as input to embedding layer and Train the Keras Embedding layer.
- **Input_clean_subcategories** --- Give 'input_teacher_prefix' column as input to embedding layer and Train the Keras Embedding layer.
- **Input_remaining_teacher_number_of_previously_posted_projects._resource_summary_contains_numerical_digits._price._quantity** --- concatenate remaining columns and add a Dense layer after that.

- For LSTM, you can choose your sequence padding methods on your own or you can train your LSTM without padding, there is no restriction on that.

Below is an example of embedding layer for a categorical columns. In below code all are dummy values, we gave only for referance.

```
# https://stats.stackexchange.com/questions/270546/how-does-keras-embedding-layer-work
#input_layer = Input(shape=(n,))
#embedding = Embedding(no_1, no_2, input_length=n)(input_layer)
#flatten = Flatten()(embedding)
```

1. Go through this blog, if you have any doubt on using predefined Embedding values in Embedding layer - https://machinelearningmastery.com/use-word-embedding-layers-deep-learning-keras/

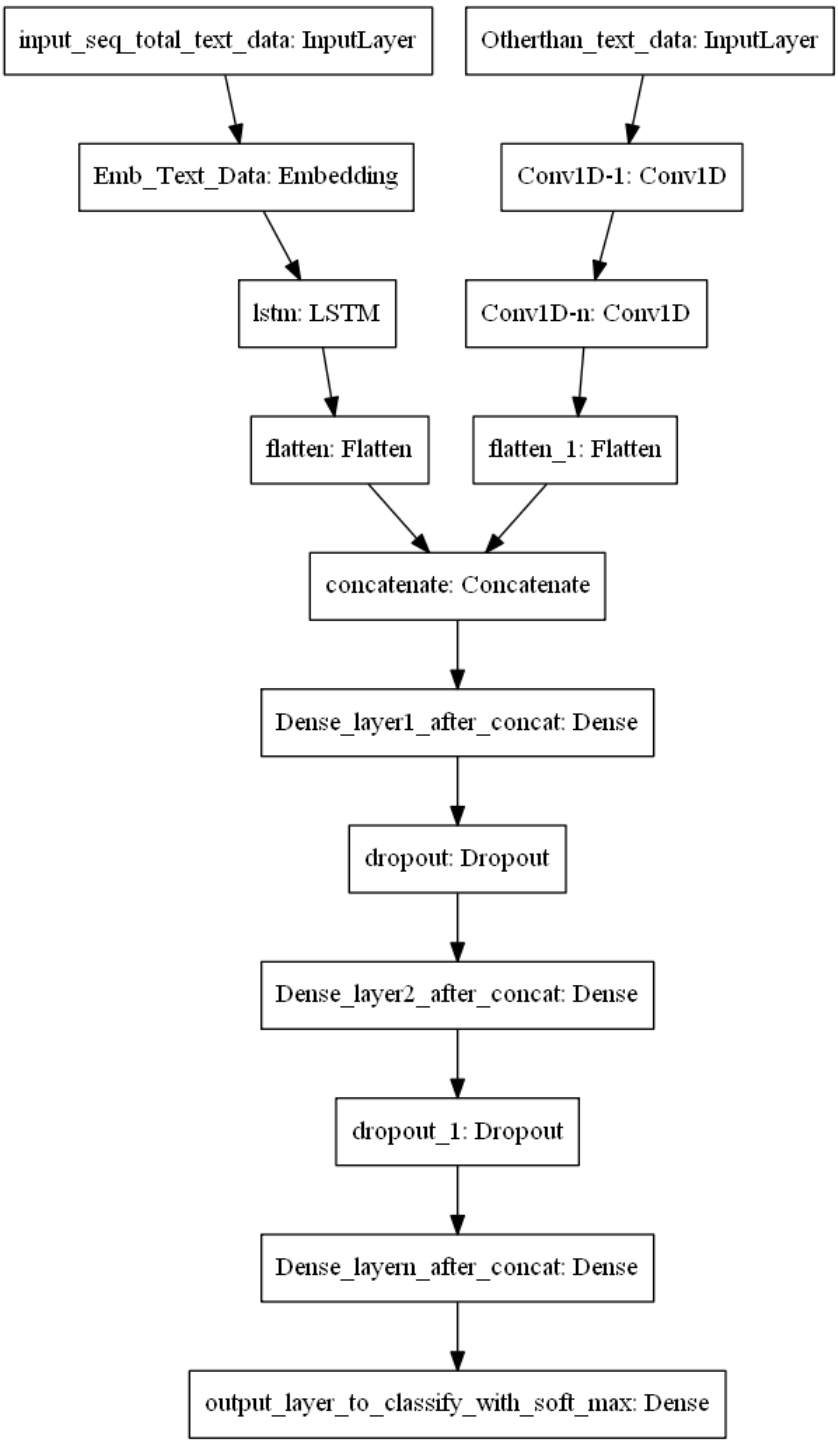
2. Please go through this link https://keras.io/getting-started/functional-api-guide/ and check the 'Multi-input and multi-output models' then you will get to know how to give multiple inputs.

Model 2

Use the same model as above but for 'input_seq_total_text_data' give only some words in the sentance not all the words. Filter the words as below.

```
1. Train the TF-IDF on the Train data

2. Get the idf value for each word we have in the train data.

3. Remove the low idf value and high idf value words from our data. Do some analysis on the Idf values and based on those values ch

4. Train the LSTM after removing the Low and High idf value words. (In model-1 Train on total data but in Model-2 train on data aft
```

Model-3



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

- **input_seq_total_text_data**:

. Use text column('essay'), and use the Embedding layer to get word vectors.

. Use given predefined glove word vectors, don't train any word vectors.

. Use LSTM that is given above, get the LSTM output and Flatten that output.

. You are free to preprocess the input text as you needed.

- **Other_than_text_data**:

. Convert all your Categorical values to onehot coded and then concatenate all these onehot vectors

. Neumerical values and use CNN1D as shown in above figure.

. You are free to choose all CNN parameters like kernel sizes, stride.

```
from google.colab import drive
drive.mount('/gdrive')
%cd /gdrive
```

```
Go to this URL in a browser: https://accounts.google.com/o/oauth2/auth?client_id=947318989803-6bn6qk8qdgf4n4g3pfee6491hc0brc4i.apps.googleuserc

Enter your authorization code:
··********··
Mounted at /gdrive
/gdrive
```

Preprocessing:

```
%matplotlib inline
import warnings
warnings.filterwarnings("ignore")

import pandas as pd
import numpy as np
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.feature_extraction.text import CountVectorizer


import re
# Tutorial about Python regular expressions: https://pymotw.com/2/re/

from nltk.corpus import stopwords
import pickle

from tqdm import tqdm
import os


resource_data = pd.read_csv('/gdrive/My Drive/LSTM_Assignment/resources.csv')


project_data = pd.read_csv('/gdrive/My Drive/LSTM_Assignment/train_data.csv')
print("Number of data points in train data", project_data.shape)
print('-'*50)
print("The attributes of data :", project_data.columns.values)
# https://stackoverflow.com/questions/36383821/pandas-dataframe-apply-function-to-column-strings-based-on-other-column-value
project_data['project_grade_category'] = project_data['project_grade_category'].str.replace(' ','_')
project_data['project_grade_category'] = project_data['project_grade_category'].str.replace('-','_')
project_data['project_grade_category'] = project_data['project_grade_category'].str.lower()

project_data['project_subject_categories'] = project_data['project_subject_categories'].str.replace(' The ','')
project_data['project_subject_categories'] = project_data['project_subject_categories'].str.replace(' ','')
project_data['project_subject_categories'] = project_data['project_subject_categories'].str.replace('&','_')
project_data['project_subject_categories'] = project_data['project_subject_categories'].str.replace(',','_')
project_data['project_subject_categories'] = project_data['project_subject_categories'].str.lower()
```

```
project_data['project_subject_categories'] = project_data['project_subject_categories'].str.lower()

print(project_data['teacher_prefix'].isnull().values.any())
print("number of nan values",project_data['teacher_prefix'].isnull().values.sum())
project_data['teacher_prefix']=project_data['teacher_prefix'].fillna('Mrs.')

project_data['teacher_prefix'] = project_data['teacher_prefix'].str.replace('.','')
project_data['teacher_prefix'] = project_data['teacher_prefix'].str.lower()


project_data['project_subject_subcategories'] = project_data['project_subject_subcategories'].str.replace(' The ','')
project_data['project_subject_subcategories'] = project_data['project_subject_subcategories'].str.replace(' ','')
project_data['project_subject_subcategories'] = project_data['project_subject_subcategories'].str.replace('&','_')
project_data['project_subject_subcategories'] = project_data['project_subject_subcategories'].str.replace(',','_')
project_data['project_subject_subcategories'] = project_data['project_subject_subcategories'].str.lower()

project_data['school_state'] = project_data['school_state'].str.lower()

# https://stackoverflow.com/a/47091490/4084039
import re

def decontracted(phrase):
    # specific
    phrase = re.sub(r"won't", "will not", phrase)
    phrase = re.sub(r"can\'t", "can not", phrase)

    # general
    phrase = re.sub(r"n\'t", " not", phrase)
    phrase = re.sub(r"\'re", " are", phrase)
    phrase = re.sub(r"\'s", " is", phrase)
    phrase = re.sub(r"\'d", " would", phrase)
    phrase = re.sub(r"\'ll", " will", phrase)
    phrase = re.sub(r"\'t", " not", phrase)
    phrase = re.sub(r"\'ve", " have", phrase)
    phrase = re.sub(r"\'m", " am", phrase)
    return phrase

# https://gist.github.com/sebleier/554280
# we are removing the words from the stop words list: 'no', 'nor', 'not'
stopwords= ['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', 'this', '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', 'all', 'any', 'both', 'each', 'few', 'more',\
            'most', 'other', 'some', 'such', 'only', 'own', 'same', 'so', 'than', 'too', 'very', \
            's', 't', 'can', 'will', 'just', 'don', "don't", 'should', "should've", 'now', 'd', 'll', 'm', 'o', 're', \
            've', 'y', 'ain', 'aren', "aren't", 'couldn', "couldn't", 'didn', "didn't", 'doesn', "doesn't", 'hadn',\
            "hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "isn't", 'ma', 'mightn', "mightn't", 'mustn',\
            "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "shouldn't", 'wasn', "wasn't", 'weren', "weren't", \
            'won', "won't", 'wouldn', "wouldn't"]

# Combining all the above stundents
from tqdm import tqdm
def preprocess_text(text_data):
    preprocessed_text = []
    # tqdm is for printing the status bar
    for sentance in tqdm(text_data):
        sent = decontracted(sentance)
        sent = sent.replace('\\r', ' ')
        sent = sent.replace('\\n', ' ')
        sent = sent.replace('\\"', ' ')
        sent = re.sub('[^A-Za-z0-9]+', ' ', sent)
        # https://gist.github.com/sebleier/554280
        sent = ' '.join(e for e in sent.split() if e.lower() not in stopwords)
        preprocessed_text.append(sent.lower().strip())
    return preprocessed_text

preprocessed_titles = preprocess_text(project_data['project_title'].values)
# merge two column text dataframe:
project_data["essay"] = project_data["project_essay_1"].map(str) +\
                        project_data["project_essay_2"].map(str) + \
                        project_data["project_essay_3"].map(str) + \
                        project_data["project_essay_4"].map(str)

preprocessed_essays = preprocess_text(project_data['essay'].values)
project_data['price'] = resource_data['price']
```

```
Number of data points in train data (109248, 17)
--------------------------------------------------
The attributes of data : ['Unnamed: 0' 'id' 'teacher_id' 'teacher_prefix' 'school_state'
 'project_submitted_datetime' 'project_grade_category'
 'project_subject_categories' 'project_subject_subcategories'
 'project_title' 'project_essay_1' 'project_essay_2' 'project_essay_3'
 'project_essay_4' 'project_resource_summary'
 'teacher_number_of_previously_posted_projects' 'project_is_approved']
True
number of nan values 3
100%|██████████| 109248/109248 [00:02<00:00, 45216.18it/s]
100%|██████████| 109248/109248 [00:58<00:00, 1857.47it/s]
```

```
project_data = project_data.drop(columns=['Unnamed: 0','id'])
project_data = project_data.drop(columns=['teacher_id','project_submitted_datetime','project_essay_1','project_essay_2','project_essay_3','project_es
```

```
project_data.shape
```

```
(109248, 9)
```

```
project_data['clean_categories'] = project_data['project_subject_categories']
project_data['clean_subcategories'] = project_data['project_subject_subcategories']
project_data= project_data.drop(labels=['project_subject_categories','project_subject_subcategories'],axis=1)
project_data.head()
```

	teacher_prefix	school_state	project_grade_category	teacher_number_of_previously_posted_projects	project_is_approved	essay	price	
0	mrs	in	grades_prek_2	0	0	My students are English learners that are work...	149.00	
1	mr	fl	grades_6_8	7	1	Our students arrive to our school eager to lea...	14.95	
2	ms	az	grades_6_8	1	0	\r\n\"True champions aren't always the ones th...	8.45	
3	mrs	ky	grades_prek_2	4	1	I work at a unique school filled with both ESL...	13.59	lite
4	mrs	tx	grades_prek_2	1	1	Our second grade classroom next year will be m...	24.95	

```
project_data.shape
```

```
(109248, 9)
```

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
preprocessed_csv = pd.read_csv('/gdrive/My Drive/LSTM_Assignment/preprocessed_data.csv')
preprocessed_csv.head()
```

	school_state	teacher_prefix	project_grade category	teacher_number_of_previously_posted_projects	project_is_approved	clean_categories
0	ca	mrs	grades_prek_2	53	1	math_science
1	ut	ms	grades_3_5	4	1	specialneeds
2	ca	mrs	grades_prek_2	10	1	literacy_language
3	ga	mrs	grades_prek_2	2	1	appliedlearning

```
y = preprocessed_csv['project_is_approved']
X = preprocessed_csv.drop(['project_is_approved'],axis=1)
print(len(X))
print(len(y))
```

```
109248
109248
```

```
preprocessed_csv.shape
```

```
(109248, 9)
```

```
# Splitting data
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size = 0.2)
```

```
embeddings_index = dict()
f = open('/gdrive/My Drive/glove.840B.300d-char.txt', 'rb')
for line in f:
  values = line.split()
  word = values[0]
  coefs = np.asarray(values[1:],dtype='float32')
  embeddings_index[word] = coefs
f.close()
```

```
from keras.preprocessing.text import Tokenizer
from keras.preprocessing.sequence import pad_sequences

embedding_words = embeddings_index.keys()
embedding_Words = set(embedding_words)
tokenizer = Tokenizer(num_words=5000)
tokenizer.fit_on_texts(X_train['essay'].values)
vocab_size = len(tokenizer.word_index)+1
X_train_en = tokenizer.texts_to_sequences(X_train['essay'].values)
X_test_en = tokenizer.texts_to_sequences(X_test['essay'].values)
```

```
X_train_padded = pad_sequences(X_train_en,maxlen=500, padding='post')
X_test_padded = pad_sequences(X_test_en,maxlen=500,padding='post')
```

```
from numpy import zeros
vocab_size = min(len(tokenizer.word_index) + 1,5000)
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
```

```
len(embedding_matrix)
```

```
5000
```

```
# School state
from sklearn.preprocessing import LabelEncoder
encoder = LabelEncoder()
unq_state = len(X_train['school_state'].unique())
embed_state = int(min(np.ceil((unq_state)/2),50))
X_train_state = encoder.fit_transform(X_train['school_state'])
X_test_state = encoder.transform(X_test['school_state'])

print("After Encoding:")
print(X_train_state.shape, y_train.shape)
print(X_test_state.shape, y_test.shape)
```

```
After Encoding:
(87398,) (87398,)
(21850,) (21850,)
```

```
#Teacher Prefix
from sklearn.preprocessing import LabelEncoder
encoder = LabelEncoder()
unq_tp = len(X_train['teacher_prefix'].unique())
embed_tp = int(min(np.ceil((unq_tp)/2),50))
X_train_teacher = encoder.fit_transform(X_train['teacher_prefix'])
X_test_teacher = encoder.transform(X_test['teacher_prefix'])

print("After Encoding:")
print(X_train_teacher.shape, y_train.shape)
print(X_test_teacher.shape, y_test.shape)
```

```
After Encoding:
(87398,) (87398,)
(21850,) (21850,)
```

```
#Project Grade Category
from sklearn.preprocessing import LabelEncoder
encoder = LabelEncoder()
unq_pg = len(X_train['project_grade_category'].unique())
embed_pg = int(min(np.ceil((unq_pg)/2),50))
X_train_grade = encoder.fit_transform(X_train['project_grade_category'])
X_test_grade = encoder.transform(X_test['project_grade_category'])

print("After Encoding:")
print(X_train_grade.shape, y_train.shape)
print(X_test_grade.shape, y_test.shape)
```

```
After Encoding:
(87398,) (87398,)
(21850,) (21850,)
```

```
#Clean categories
from sklearn.preprocessing import LabelEncoder
encoder = LabelEncoder()
unq_cc = len(X_train['clean_categories'].unique())
embed_cc = int(min(np.ceil((unq_cc)/2),50))
X_train_clean_cat = encoder.fit_transform(X_train['clean_categories'])
classes = (encoder.classes_).tolist()
X_test['clean_categories'] = X_test['clean_categories'].map(lambda s: 'other' if s not in encoder.classes_ else s)
#https://stackoverflow.com/questions/40321232/handling-unknown-values-for-label-encoding

encoder.classes_ = np.append(classes,'other')

X_test_clean_cat = encoder.transform(X_test['clean_categories'])

print("After Encoding:")
print(X_train_clean_cat.shape, y_train.shape)
print(X_test_clean_cat.shape, y_test.shape)
```

```
After Encoding:
(87398,) (87398,)
(21850,) (21850,)
```

```
#Clean Sub Categories
from sklearn.preprocessing import LabelEncoder
encoder = LabelEncoder()
unq_csc = len(X_train['clean_subcategories'].unique())
embed_csc = int(min(np.ceil((unq_csc)/2),50))
encoder.fit(X_train['clean_subcategories'])
X_train_clean_subcat = encoder.transform(X_train['clean_subcategories'])
classes = (encoder.classes_).tolist()
X_test['clean_subcategories'] = X_test['clean_subcategories'].map(lambda s: 'other' if s not in encoder.classes_ else s)
#https://stackoverflow.com/questions/40321232/handling-unknown-values-for-label-encoding

encoder.classes_ = np.append(classes,'other')
X_test_clean_subcat = encoder.transform(X_test['clean_subcategories'])

print("After Encoding:")
print(X_train_clean_subcat.shape, y_train.shape)
print(X_test_clean_subcat.shape, y_test.shape)
```

```
After Encoding:
(87398,) (87398,)
(21850,) (21850,)
```

```
from sklearn.preprocessing import MinMaxScaler
min_max = MinMaxScaler()
# normalizer.fit(X_train['price'].values)
# this will rise an error Expected 2D array, got 1D array instead:
# array=[105.22 215.96  96.01 ... 368.98  80.53 709.67].
# Reshape your data either using
# array.reshape(-1, 1) if your data has a single feature
# array.reshape(1, -1)  if it contains a single sample.
min_max.fit(X_train['teacher_number_of_previously_posted_projects'].values.reshape(-1,1))

X_train_teacher_prev_min_max = min_max.transform(X_train['teacher_number_of_previously_posted_projects'].values.reshape(-1,1))
X_test_teacher_prev_min_max = min_max.transform(X_test['teacher_number_of_previously_posted_projects'].values.reshape(-1,1))
```

```
print("After vectorizations")
print(X_train_teacher_prev_min_max.shape, y_train.shape)
print(X_test_teacher_prev_min_max.shape, y_test.shape)
print("="*100)
```

```
After vectorizations
(87398, 1) (87398,)
(21850, 1) (21850,)
====================================================================================================
```

```
min_max = MinMaxScaler()
# normalizer.fit(X_train['price'].values)
# this will rise an error Expected 2D array, got 1D array instead:
# array=[105.22 215.96  96.01 ... 368.98  80.53 709.67].
# Reshape your data either using
# array.reshape(-1, 1) if your data has a single feature
# array.reshape(1, -1)  if it contains a single sample.
min_max.fit(X_train['price'].values.reshape(-1,1))

X_train_price_min_max = min_max.transform(X_train['price'].values.reshape(-1,1))
X_test_price_min_max= min_max.transform(X_test['price'].values.reshape(-1,1))
```

```
print("After vectorizations")
print(X_train_price_min_max.shape, y_train.shape)
print(X_test_price_min_max.shape, y_test.shape)
print("="*100)
```

After vectorizations

```
import os
import datetime
import sys
import numpy as np
import keras
import tensorflow as tf
from keras.preprocessing.text import Tokenizer
from keras.utils import to_categorical
from keras.preprocessing.sequence import pad_sequences
from keras.layers import Activation, Conv2D, Input, Embedding, Reshape, MaxPool2D, Concatenate, Flatten, Dropout, Dense, Conv1D, MaxPooling1D, BatchN
from keras.layers import MaxPool1D
from keras.models import Model
from keras.callbacks import ModelCheckpoint
from keras.optimizers import Adam
from keras.layers import LSTM, concatenate
from sklearn.metrics import roc_auc_score
def auc(y_true,y_pred):
  if len(np.unique(y_true[:,1]))==1:
    return 0.5
  elif len(np.unique(y_test[:,1]))==1:
    return 0.5
  else:
    return roc_auc_score(y_true,y_pred)

def auroc(y_true, y_pred): # https://stackoverflow.com/questions/41032551/how-to-compute-receiving-operating-characteristic-roc-and-auc-in-keras
    return tf.py_function(auc, (y_true, y_pred), tf.double)




X_train_num = np.concatenate((X_train_teacher_prev_min_max,X_train_price_min_max), axis = 1)
X_test_num = np.concatenate((X_test_teacher_prev_min_max,X_test_price_min_max ), axis=1)


from keras.utils import to_categorical
y_train = to_categorical(y_train,2)
y_test = to_categorical(y_test,2)


#callbacks
reduce_lr = tf.keras.callbacks.ReduceLROnPlateau(monitor='val_loss')# factor=0.2,patience=1,mode = 'min',min_lr=min_lr)
es = tf.keras.callbacks.EarlyStopping(monitor="val_loss",min_delta=0,patience=1,verbose=1,mode="auto",baseline=None, restore_best_weights=False)


import datetime, os
import warnings
from tensorflow.keras.callbacks import TensorBoard
from tensorflow.keras.callbacks import ModelCheckpoint
essay = Input(shape= (500,), name = 'essay')
first= Embedding(5000, 300, input_length= 500, weights = [embedding_matrix],trainable =False)(essay)
lstm_essay = LSTM(128, return_sequences= True)(first)
flatten1 = Flatten()(lstm_essay)

ss = Input(shape = ( 1, ), name ="school_state" ) #51
second = Embedding( input_dim = unq_state, output_dim= embed_state)(ss)
flatten2 = Flatten()(second)

tp = Input(shape = (1, ), name = 'teacher_prefix') #1
third = Embedding (input_dim = unq_tp, output_dim= embed_tp)(tp)
flatten3 = Flatten()(third)

cc = Input(shape =(1,), name ='clean_categories') #9
fourth =Embedding (input_dim = unq_cc, output_dim= embed_cc)(cc)
flatten4 = Flatten()(fourth)

csc = Input(shape =(1,), name ='clean_subcategories') #9
fifth =Embedding (input_dim = 1+unq_csc, output_dim= embed_csc)(csc)
flatten5 = Flatten()(fifth)

pgc = Input(shape =(1,), name ='project_grade_category') #4
sixth =Embedding (input_dim = unq_pg, output_dim= embed_pg)(pgc)
flatten6 = Flatten()(sixth)

num = Input(shape = (2,), name = 'numerical_features') #2
seventh = Dense(128, activation='relu')(num)

concat = concatenate([flatten1, flatten2,flatten3, flatten4,flatten5,flatten6, seventh])

model = Dense(64, activation='relu', kernel_initializer='he_normal')(concat)
```

```
model = Dense(64, activation='relu', kernel_initializer='he_normal')(concat)
model = Dropout(0.1)(model)

model = Dense(32, activation='relu', kernel_initializer='he_normal')(model)
model = Dropout(0.1)(model)

model = Dense(32, activation='relu', kernel_initializer='he_normal')(model)
model = BatchNormalization()(model)
Output = Dense(2, activation = 'softmax', name ='output')(model)
#adam = tf.keras.optimizers.Adam(learning_rate=0.00001)
sgd = tf.keras.optimizers.SGD(learning_rate=0.001, momentum=0.6, nesterov=False, name='SGD')
model = Model(inputs= [essay,ss,tp,cc,csc,pgc,num], outputs = [Output])
model.compile(loss = 'binary_crossentropy', optimizer=sgd, metrics = ['accuracy', auroc])

checkpoint_path = "/gdrive/My Drive/LSTM_Assignment/logs1/my_model_weights.hdf5"
checkpoint_dir = os.path.dirname(checkpoint_path)
checkpoint = tf.keras.callbacks.ModelCheckpoint(filepath='/gdrive/My Drive/LSTM_Assignment/logs1/my_model_weights{epoch}.hdf5',  verbose=1, save_weig
log_dir="/gdrive/My Drive/LSTM_Assignment/logs1/fit/" + datetime.datetime.now().strftime("%Y%m%d-%H%M%S")
tensorboard_callback = tf.keras.callbacks.TensorBoard(log_dir=log_dir,histogram_freq=1, write_graph=True,write_grads=True)

training_input = [X_train_padded,X_train_state, X_train_teacher,X_train_clean_cat,X_train_clean_subcat, X_train_grade, X_train_num]
val_input = [X_test_padded,X_test_state,X_test_teacher,X_test_clean_cat,X_test_clean_subcat,X_test_grade, X_test_num ]
model.fit(training_input,y_train,validation_data=(val_input,y_test), epochs = 25, callbacks=[reduce_lr,tensorboard_callback])
model.save_weights('/gdrive/My Drive/LSTM_Assignment/logs1/my_model_weights{epoch}.hdf5')
model.save('/gdrive/My Drive/LSTM_Assignment/logs1')
```

```
WARNING:tensorflow:`write_grads` will be ignored in TensorFlow 2.0 for the `TensorBoard` Callback.
Epoch 1/25
   1/2732 [..............................] - ETA: 0s - loss: 0.6927 - accuracy: 0.5000 - auroc: 0.5217WARNING:tensorflow:From /usr/local/lib/py
Instructions for updating:
use `tf.profiler.experimental.stop` instead.
2732/2732 [==============================] - 2531s 926ms/step - loss: 0.4409 - accuracy: 0.8461 - auroc: 0.5176 - val_loss: 0.4263 - val_accura
Epoch 2/25
2732/2732 [==============================] - 2563s 938ms/step - loss: 0.4240 - accuracy: 0.8488 - auroc: 0.5419 - val_loss: 0.4248 - val_accura
Epoch 3/25
2732/2732 [==============================] - 2571s 941ms/step - loss: 0.4217 - accuracy: 0.8488 - auroc: 0.5656 - val_loss: 0.4236 - val_accura
Epoch 4/25
2732/2732 [==============================] - 2609s 955ms/step - loss: 0.4200 - accuracy: 0.8488 - auroc: 0.5811 - val_loss: 0.4205 - val_accura
Epoch 5/25
2732/2732 [==============================] - 2651s 970ms/step - loss: 0.4183 - accuracy: 0.8488 - auroc: 0.5944 - val_loss: 0.4299 - val_accura
Epoch 6/25
2732/2732 [==============================] - 2614s 957ms/step - loss: 0.4177 - accuracy: 0.8488 - auroc: 0.5947 - val_loss: 0.4180 - val_accura
Epoch 7/25
2732/2732 [==============================] - 2554s 935ms/step - loss: 0.4173 - accuracy: 0.8488 - auroc: 0.5984 - val_loss: 0.4172 - val_accura
Epoch 8/25
2732/2732 [==============================] - 2606s 954ms/step - loss: 0.4169 - accuracy: 0.8488 - auroc: 0.6011 - val_loss: 0.4174 - val_accura
Epoch 9/25
2732/2732 [==============================] - 2580s 944ms/step - loss: 0.4164 - accuracy: 0.8488 - auroc: 0.6045 - val_loss: 0.4166 - val_accura
Epoch 10/25
2732/2732 [==============================] - 2563s 938ms/step - loss: 0.4163 - accuracy: 0.8488 - auroc: 0.6093 - val_loss: 0.4193 - val_accura
Epoch 11/25
2732/2732 [==============================] - 2544s 931ms/step - loss: 0.4163 - accuracy: 0.8488 - auroc: 0.6041 - val_loss: 0.4187 - val_accura
Epoch 12/25
2732/2732 [==============================] - 2596s 950ms/step - loss: 0.4167 - accuracy: 0.8488 - auroc: 0.6061 - val_loss: 0.4196 - val_accura
Epoch 13/25
2732/2732 [==============================] - 2605s 953ms/step - loss: 0.4163 - accuracy: 0.8488 - auroc: 0.6074 - val_loss: 0.4194 - val_accura
Epoch 14/25
2732/2732 [==============================] - 2582s 945ms/step - loss: 0.4164 - accuracy: 0.8488 - auroc: 0.6066 - val_loss: 0.4180 - val_accura
Epoch 15/25
2732/2732 [==============================] - 2602s 952ms/step - loss: 0.4157 - accuracy: 0.8488 - auroc: 0.6105 - val_loss: 0.4181 - val_accura
Epoch 16/25
2732/2732 [==============================] - 2613s 956ms/step - loss: 0.4154 - accuracy: 0.8488 - auroc: 0.6114 - val_loss: 0.4195 - val_accura
Epoch 17/25
2079/2732 [=====================>........] - ETA: 9:35 - loss: 0.4147 - accuracy: 0.8491 - auroc: 0.6165
```

```
model.save('/gdrive/My Drive/LSTM_Assignment/logs1')
```

```
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/tensorflow/python/training/tracking/tracking.py:111: Model.state_updates (from t
Instructions for updating:
This property should not be used in TensorFlow 2.0, as updates are applied automatically.
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/tensorflow/python/training/tracking/tracking.py:111: Layer.updates (from tensorf
Instructions for updating:
This property should not be used in TensorFlow 2.0, as updates are applied automatically.
INFO:tensorflow:Assets written to: /gdrive/My Drive/LSTM_Assignment/logs1/assets
```

```
model = tf.keras.models.load_model('/gdrive/My Drive/LSTM_Assignment/logs1',custom_objects={'auroc':auroc})
```

```
import datetime, os
import warnings
from tensorflow.keras.callbacks import TensorBoard
```

```
from tensorflow.keras.callbacks import TensorBoard
from tensorflow.keras.callbacks import ModelCheckpoint
model = tf.keras.models.load_model('/gdrive/My Drive/LSTM_Assignment/logs1',custom_objects={'auroc':auroc})
training_input = [X_train_padded,X_train_state, X_train_teacher,X_train_clean_cat,X_train_clean_subcat, X_train_grade, X_train_num]
val_input = [X_test_padded,X_test_state,X_test_teacher,X_test_clean_cat,X_test_clean_subcat,X_test_grade, X_test_num ]

checkpoint_path = "/gdrive/My Drive/LSTM_Assignment/logs1/my_model_weights.hdf5"
checkpoint_dir = os.path.dirname(checkpoint_path)
checkpoint = tf.keras.callbacks.ModelCheckpoint(filepath='/gdrive/My Drive/LSTM_Assignment/logs1/my_model_weights{epoch}.hdf5',  verbose=1, save_weig
log_dir="/gdrive/My Drive/LSTM_Assignment/logs1/fit/" + datetime.datetime.now().strftime("%Y%m%d-%H%M%S")
tensorboard_callback = tf.keras.callbacks.TensorBoard(log_dir=log_dir,histogram_freq=1, write_graph=True,write_grads=True)


model.fit(training_input,y_train,validation_data=(val_input,y_test), epochs = 15, callbacks=[tensorboard_callback])
model.save_weights('/gdrive/My Drive/LSTM_Assignment/logs1/my_model_weights{epoch}.hdf5')
model.save('/gdrive/My Drive/LSTM_Assignment/logs1')
```

```
write_grads` will be ignored in TensorFlow 2.0 for the `TensorBoard` Callback.

.....................] - ETA: 0s - loss: 0.5333 - accuracy: 0.7812 - auroc: 0.7771WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages
ating:
erimental.stop` instead.
====================] - 3277s 1s/step - loss: 0.4215 - accuracy: 0.8481 - auroc: 0.5808 - val_loss: 0.4186 - val_accuracy: 0.8504 - val_auroc:

====================] - 3248s 1s/step - loss: 0.4164 - accuracy: 0.8481 - auroc: 0.6134 - val_loss: 0.4060 - val_accuracy: 0.8504 - val_auroc:

====================] - 3296s 1s/step - loss: 0.4055 - accuracy: 0.8482 - auroc: 0.6632 - val_loss: 0.4871 - val_accuracy: 0.8504 - val_auroc:

====================] - 3345s 1s/step - loss: 0.3954 - accuracy: 0.8482 - auroc: 0.6958 - val_loss: 0.3851 - val_accuracy: 0.8512 - val_auroc:

.....................] - ETA: 46:38 - loss: 0.3954 - accuracy: 0.8466 - auroc: 0.7044
---------------------------------------------------------------------------
                              Traceback (most recent call last)
98b24ff400> in <module>()


aining_input,y_train,validation_data=(val_input,y_test), epochs = 15, callbacks=[tensorboard_callback])
eights('/gdrive/My Drive/LSTM_Assignment/logs1/my_model_weights{epoch}.hdf5')
/gdrive/My Drive/LSTM_Assignment/logs1')

                              8 frames
n3.6/dist-packages/tensorflow/python/eager/execute.py in quick_execute(op_name, num_outputs, inputs, attrs, ctx, name)
re_initialized()
= pywrap_tfe.TFE_Py_Execute(ctx._handle, device_name, op_name,
                            inputs, attrs, num_outputs)
e._NotOkStatusException as e:
is not None:
```

FLOW

```
model.summary()
```

```
Model: "functional_3"
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
essay (InputLayer)              [(None, 500)]        0
__________________________________________________________________________________________________
embedding_6 (Embedding)         (None, 500, 300)     1500000     essay[0][0]
__________________________________________________________________________________________________
school_state (InputLayer)       [(None, 1)]          0
__________________________________________________________________________________________________
teacher_prefix (InputLayer)     [(None, 1)]          0
__________________________________________________________________________________________________
clean_categories (InputLayer)   [(None, 1)]          0
__________________________________________________________________________________________________
clean_subcategories (InputLayer [(None, 1)]          0
__________________________________________________________________________________________________
project_grade_category (InputLa [(None, 1)]          0
__________________________________________________________________________________________________
lstm_1 (LSTM)                   (None, 500, 128)     219648      embedding_6[0][0]
__________________________________________________________________________________________________
embedding_7 (Embedding)         (None, 1, 26)        1326        school_state[0][0]
__________________________________________________________________________________________________
embedding_8 (Embedding)         (None, 1, 3)         15          teacher_prefix[0][0]
__________________________________________________________________________________________________
embedding_9 (Embedding)         (None, 1, 26)        1326        clean_categories[0][0]
__________________________________________________________________________________________________
embedding_10 (Embedding)        (None, 1, 50)        19750       clean_subcategories[0][0]
__________________________________________________________________________________________________
embedding_11 (Embedding)        (None, 1, 2)         8           project_grade_category[0][0]
__________________________________________________________________________________________________
numerical_features (InputLayer) [(None, 2)]          0
__________________________________________________________________________________________________
flatten_6 (Flatten)             (None, 64000)        0           lstm_1[0][0]
__________________________________________________________________________________________________
flatten_7 (Flatten)             (None, 26)           0           embedding_7[0][0]
__________________________________________________________________________________________________
flatten_8 (Flatten)             (None, 3)            0           embedding_8[0][0]
__________________________________________________________________________________________________
flatten_9 (Flatten)             (None, 26)           0           embedding_9[0][0]
__________________________________________________________________________________________________
flatten_10 (Flatten)            (None, 50)           0           embedding_10[0][0]
__________________________________________________________________________________________________
flatten_11 (Flatten)            (None, 2)            0           embedding_11[0][0]
__________________________________________________________________________________________________
```

```
%load_ext tensorboard
!kill 465
%tensorboard --logdir "/gdrive/My Drive/LSTM_Assignment/logs1/fit/"
```

Model:	AUC-ROC Score:	Accuracy:
2	0.7276	85.07%
3	0.7663	85.47%

Model 3's Auc score is higher hence that is the best performance. Conv1D layers to other than text data has made an impact.

```
/bin/bash: line 0: kill: (465) - No such process
```

TensorBoard SCALARS GRAPHS DISTRIBUTIONS HISTOGRAMS INACTIVE

Show data download links

Ignore outliers in chart scaling

Filter tags (regular expressions supported)

epoch_accuracy

Model 2:

```
from sklearn.feature_extraction.text import TfidfVectorizer
tfidf = TfidfVectorizer()
tfidf_essay = tfidf.fit_transform(X_train['essay'])
#create a dataframe
lst1 = list(tfidf.get_feature_names())
lst2 = list(tfidf.idf_)
df = pd.DataFrame(list(zip(lst1, lst2)), columns =['Word', 'Idf'])
```

```
df.sort_values(by ='Idf', inplace = True)
df.head()
```

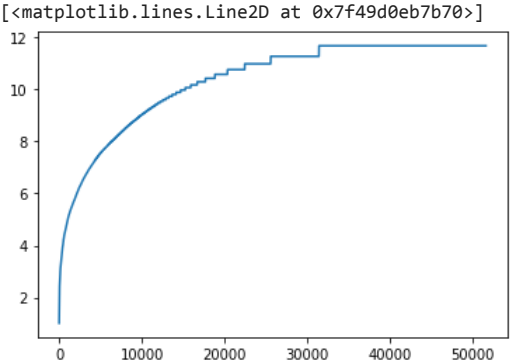
	Word	Idf
44096	students	1.007684
30773	nannan	1.045074
40201	school	1.160289
30658	my	1.244351
26636	learning	1.363322

```
#Analysis
idf_values = list(tfidf.idf_)
print('Min value:', min(idf_values))
print('Max value:', max(idf_values))
```

```
Min value: 1.007683996252072
Max value: 11.685091939370627
```

```
import matplotlib.pyplot as plt
plt.plot(sorted(idf_values))
```

```
[<matplotlib.lines.Line2D at 0x7f49d0eb7b70>]
```




```
import seaborn as sns
sns.boxplot(y=sorted(idf_values))
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f49d09e4630>
```

```
print('PERCENTILES:')
print('*'*40)
print('0%:  ',np.percentile(idf_values,0))
print('25%: ',np.percentile(idf_values,25))
print('75%: ',np.percentile(idf_values,75))
print('90%: ',np.percentile(idf_values,90))
print('100%:',np.percentile(idf_values,100))
```

```
PERCENTILES:
****************************************
0%:   1.007683996252072
25%:  9.670188918828364
75%:  11.685091939370627
90%:  11.685091939370627
100%: 11.685091939370627
```

```
idf_new = (df['Idf']>=2) & (df['Idf']<=11)
new_df = df[idf_new]
print(df[idf_new].shape)
```

```
(25560, 2)
```

```
new_text = new_df['Word'].tolist()
tokenizer1 = Tokenizer()
tokenizer1.fit_on_texts(new_text)
X_train['new_text'] = tokenizer1.texts_to_sequences(X_train['essay'])
X_test['new_text'] = tokenizer1.texts_to_sequences(X_test['essay'])
```

```
max_vocab = len(tokenizer1.word_index)
```

```
X_train_padded_model_2 = pad_sequences(X_train['new_text'].values, maxlen = 500, padding='post')
X_test_padded_model_2 = pad_sequences(X_test['new_text'].values, maxlen = 500, padding='post')
```

```
embeddings_index = dict()
f = open('/gdrive/My Drive/glove.840B.300d-char.txt', 'rb')
for line in f:
  values = line.split()
  word = values[0]
  coefs = np.asarray(values[1:],dtype='float32')
  embeddings_index[word] = coefs
f.close()
```

```
max_vocab = len(tokenizer1.word_index)
from numpy import zeros
embedding_matrix = zeros((max_vocab+1,300))
for word,i in tokenizer1.word_index.items():
  embedding_vector = embeddings_index.get(word)
  if embedding_vector is not None:
    embedding_matrix[i] = embedding_vector
```

```
essay = Input(shape= (500,), name = 'essay')
first= Embedding(input_dim = max_vocab +1, output_dim= 300, input_length= 500, weights = [embedding_matrix],trainable =False)(essay)
lstm_essay = LSTM(64, return_sequences= True)(first)
flatten1 = Flatten()(lstm_essay)

ss = Input(shape = ( 1, ), name ="school_state" )
second = Embedding( input_dim = unq_state, output_dim= embed_state)(ss)
flatten2 = Flatten()(second)

tp = Input(shape = (1, ), name = 'teacher_prefix')
third = Embedding (input_dim = unq_tp, output_dim= embed_tp)(tp)
flatten3 = Flatten()(third)

cc = Input(shape =(1,), name ='clean_categories')
fourth =Embedding (input_dim = unq_cc, output_dim= embed_cc)(cc)
flatten4 = Flatten()(fourth)

csc = Input(shape =(1,), name ='clean_subcategories')
fifth =Embedding (input_dim = 1+unq_csc, output_dim= embed_csc)(csc)
flatten5 = Flatten()(fifth)

pgc = Input(shape =(1,), name ='project_grade_category')
sixth =Embedding (input_dim = unq_pg, output_dim= embed_pg)(pgc)
```

```
flatten6 = Flatten()(sixth)

num = Input(shape = (2,), name = 'numerical_features')
seventh = Dense(1, activation='relu')(num)

concat = concatenate([flatten1, flatten2,flatten3, flatten4,flatten5,flatten6, seventh])

model2 = Dense(128, activation='relu', kernel_initializer='he_normal')(concat)
model2 = Dropout(0.2)(model2)

model2 = Dense(64, activation='relu', kernel_initializer='he_normal')(model2)
model2 = Dropout(0.1)(model2)

model2= Dense(64, activation='relu', kernel_initializer='he_normal')(model2)
model2 = BatchNormalization()(model2)
Output = Dense(2, activation = 'softmax', name ='output')(model2)
#adam = tf.keras.optimizers.Adam(learning_rate=0.0001)
adamax = tf.keras.optimizers.Adamax(learning_rate=0.001, beta_1=0.9, beta_2=0.999, epsilon=1e-07, name='Adamax')
model2 = Model(inputs= [essay,ss,tp,cc,csc,pgc,num], outputs = [Output])
model2.compile(loss = 'categorical_crossentropy', optimizer=adamax, metrics = ['accuracy', auroc])

checkpoint_path = "/gdrive/My Drive/LSTM_Assignment/logs2/my_model_weights.hdf5"
checkpoint_dir = os.path.dirname(checkpoint_path)
checkpoint = tf.keras.callbacks.ModelCheckpoint(filepath='/gdrive/My Drive/LSTM_Assignment/logs2/my_model_weights{epoch}.hdf5',monitor ='val_loss',
log_dir="/gdrive/My Drive/LSTM_Assignment/logs2/fit/" + datetime.datetime.now().strftime("%Y%m%d-%H%M%S")
tensorboard_callback = tf.keras.callbacks.TensorBoard(log_dir=log_dir,histogram_freq=1, write_graph=True,write_grads=True)


training_input = [X_train_padded_model_2,X_train_state, X_train_teacher,X_train_clean_cat,X_train_clean_subcat, X_train_grade, X_train_num]
val_input = [X_test_padded_model_2,X_test_state,X_test_teacher,X_test_clean_cat,X_test_clean_subcat,X_test_grade, X_test_num ]
model2.fit(training_input,y_train,validation_data=(val_input,y_test), epochs = 20, callbacks= [tensorboard_callback,reduce_lr])

model2.save_weights('/gdrive/My Drive/LSTM_Assignment/logs2/my_model_weights{epoch}.hdf5')
model2.save('/gdrive/My Drive/LSTM_Assignment/logs2')
```

```
WARNING:tensorflow:`write_grads` will be ignored in TensorFlow 2.0 for the `TensorBoard` Callback.
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
<ipython-input-46-e66e74a6049b> in <module>()
     52 training_input = [X_train_padded_model_2,X_train_state, X_train_teacher,X_train_clean_cat,X_train_clean_subcat, X_train_grade,
X_train_num]
     53 val_input = [X_test_padded_model_2,X_test_state,X_test_teacher,X_test_clean_cat,X_test_clean_subcat,X_test_grade, X_test_num ]
---> 54 model2.fit(training_input,y_train,validation_data=(val_input,y_test), epochs = 20, callbacks= [tensorboard_callback,reduce_lr])
     55
     56 model2.save_weights('/gdrive/My Drive/LSTM_Assignment/logs2/my_model_weights{epoch}.hdf5')

                                   7 frames
/usr/local/lib/python3.6/dist-packages/tensorflow/python/ops/summary_ops_v2.py in _serialize_graph(arbitrary_graph)
    987 def _serialize_graph(arbitrary_graph):
    988   if isinstance(arbitrary_graph, ops.Graph):
--> 989     return arbitrary_graph.as_graph_def(add_shapes=True).SerializeToString()
    990   else:
    991     return arbitrary_graph.SerializeToString()

KeyboardInterrupt:
```

SEARCH STACK OVERFLOW

```
model2 = tf.keras.models.load_model('/gdrive/My Drive/LSTM_Assignment/logs2',custom_objects={'auroc':auroc})



checkpoint_path = "/gdrive/My Drive/LSTM_Assignment/logs2/my_model_weights.hdf5"
checkpoint_dir = os.path.dirname(checkpoint_path)
checkpoint = tf.keras.callbacks.ModelCheckpoint(filepath='/gdrive/My Drive/LSTM_Assignment/logs2/my_model_weights{epoch}.hdf5',monitor ='val_loss',
log_dir="/gdrive/My Drive/LSTM_Assignment/logs2/fit/" + datetime.datetime.now().strftime("%Y%m%d-%H%M%S")
tensorboard_callback = tf.keras.callbacks.TensorBoard(log_dir=log_dir,histogram_freq=1, write_graph=True,write_grads=True)


training_input = [X_train_padded_model_2,X_train_state, X_train_teacher,X_train_clean_cat,X_train_clean_subcat, X_train_grade, X_train_num]
val_input = [X_test_padded_model_2,X_test_state,X_test_teacher,X_test_clean_cat,X_test_clean_subcat,X_test_grade, X_test_num ]
model2.compile(loss = 'categorical_crossentropy', optimizer=tf.keras.optimizers.Adam(), metrics = ['accuracy', auroc])
model2.fit(training_input,y_train,validation_data=(val_input,y_test), epochs = 20, callbacks= [tensorboard_callback,reduce_lr])
model2.save('/gdrive/My Drive/LSTM_Assignment/logs2')
```

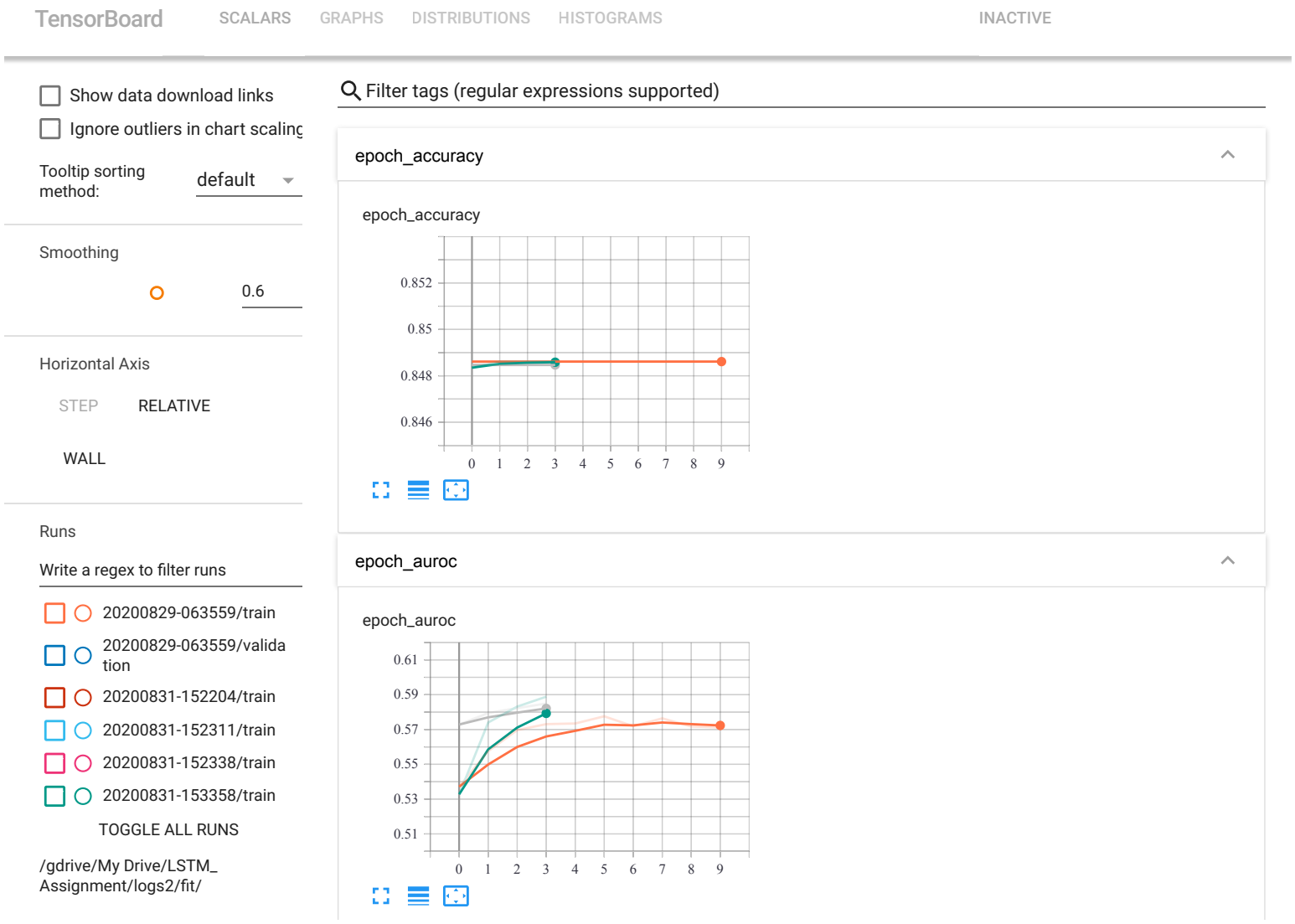
```
_grads` will be ignored in TensorFlow 2.0 for the `TensorBoard` Callback.

...............] - ETA: 0s - loss: 0.5521 - accuracy: 0.7812 - auroc: 0.4743WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/tens

ntal.stop` instead.
==============] - 1829s 669ms/step - loss: 0.4270 - accuracy: 0.8481 - auroc: 0.5451 - val_loss: 0.3930 - val_accuracy: 0.8468 - val_auroc: 0.

==============] - 1821s 666ms/step - loss: 0.3863 - accuracy: 0.8493 - auroc: 0.7088 - val_loss: 0.3877 - val_accuracy: 0.8507 - val_auroc: 0.

=====>.........] - ETA: 8:28 - loss: 0.3585 - accuracy: 0.8530 - auroc: 0.7773
---------------------------------------------------------------------------
                    Traceback (most recent call last)
47977> in <module>()
st_padded_model_2,X_test_state,X_test_teacher,X_test_clean_cat,X_test_clean_subcat,X_test_grade, X_test_num ]
ss = 'categorical_crossentropy', optimizer=tf.keras.optimizers.Adam(), metrics = ['accuracy', auroc])
ng_input,y_train,validation_data=(val_input,y_test), epochs = 20, callbacks= [tensorboard_callback,reduce_lr])
ive/My Drive/LSTM_Assignment/logs2')

8 frames

dist-packages/tensorflow/python/eager/execute.py in quick_execute(op_name, num_outputs, inputs, attrs, ctx, name)
itialized()
rap_tfe.TFE_Py_Execute(ctx._handle, device_name, op_name,
                       inputs, attrs, num_outputs)
OkStatusException as e:
: None:
```

```
model2.summary()
```

```
Model: "functional_1"
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
essay (InputLayer)              [(None, 500)]        0
__________________________________________________________________________________________________
embedding (Embedding)           (None, 500, 300)     7684800     essay[0][0]
__________________________________________________________________________________________________
school_state (InputLayer)       [(None, 1)]          0
__________________________________________________________________________________________________
teacher_prefix (InputLayer)     [(None, 1)]          0
__________________________________________________________________________________________________
clean_categories (InputLayer)   [(None, 1)]          0
__________________________________________________________________________________________________
```

```
%load_ext tensorboard
!kill 465
%tensorboard --logdir "/gdrive/My Drive/LSTM_Assignment/logs2/fit/"
```

```
/bin/bash: line 0: kill: (465) - No such process
```




```
Total params: 11,923,422
```

Model 3:

```
# text data
embeddings_index = dict()
f = open('/gdrive/My Drive/glove.840B.300d-char.txt', 'rb')
for line in f:
  values = line.split()
  word = values[0]
  coefs = np.asarray(values[1:],dtype='float32')
  embeddings_index[word] = coefs
f.close()
```

```
from keras.preprocessing.text import Tokenizer
```

```
from keras.preprocessing.sequence import pad_sequences

embedding_words = embeddings_index.keys()
embedding_Words = set(embedding_words)
tokenizer = Tokenizer(num_words=5000)
tokenizer.fit_on_texts(X_train['essay'].values)
vocab_size = len(tokenizer.word_index)+1
X_train_en = tokenizer.texts_to_sequences(X_train['essay'].values)
X_test_en = tokenizer.texts_to_sequences(X_test['essay'].values)



X_train_padded = pad_sequences(X_train_en,maxlen=500, padding='post')
X_test_padded = pad_sequences(X_test_en,maxlen=500,padding='post')
from numpy import zeros
vocab_size = min(len(tokenizer.word_index) + 1,5000)
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


#Other than input text:
# School state
from sklearn.feature_extraction.text import CountVectorizer
vectorizer = CountVectorizer(binary=True)
X_train_state = vectorizer.fit_transform(X_train['school_state'])

X_test_state = vectorizer.transform(X_test['school_state'])

print("After Encoding:")
print(X_train_state.shape, y_train.shape)
print(X_test_state.shape, y_test.shape)
```

```
After Encoding:
(87398, 51) (87398, 2)
(21850, 51) (21850, 2)
```

```
#Teacher Prefix
from sklearn.feature_extraction.text import CountVectorizer
vectorizer = CountVectorizer(binary=True)
X_train_tp = vectorizer.fit_transform(X_train['teacher_prefix'])
X_test_tp = vectorizer.transform(X_test['teacher_prefix'])

print("After Encoding:")
print(X_train_tp.shape, y_train.shape)
print(X_test_tp.shape, y_test.shape)
```

```
After Encoding:
(87398, 5) (87398, 2)
(21850, 5) (21850, 2)
```

```
#Project Grade Category
from sklearn.feature_extraction.text import CountVectorizer
vectorizer = CountVectorizer(binary=True)
X_train_pgc = vectorizer.fit_transform(X_train['project_grade_category'])

X_test_pgc = vectorizer.transform(X_test['project_grade_category'])

print("After Encoding:")
print(X_train_pgc.shape, y_train.shape)
print(X_test_pgc.shape, y_test.shape)
```

```
After Encoding:
(87398, 4) (87398, 2)
(21850, 4) (21850, 2)
```

```
#Clean Categories
from sklearn.feature_extraction.text import CountVectorizer
vectorizer = CountVectorizer(binary=True)
X_train_cc = vectorizer.fit_transform(X_train['clean_categories'])

X_test_cc = vectorizer.transform(X_test['clean_categories'])

print("After Encoding:")
print(X_train_cc.shape, y_train.shape)
print(X_test_cc.shape, y_test.shape)
```

```
After Encoding:
(87398, 9) (87398, 2)
(21850, 9) (21850, 2)
```

```
#Clean Sub Categories
from sklearn.feature_extraction.text import CountVectorizer
vectorizer = CountVectorizer(binary=True)
X_train_csc = vectorizer.fit_transform(X_train['clean_subcategories'])

X_test_csc = vectorizer.transform(X_test['clean_subcategories'])

print("After Encoding:")
print(X_train_csc.shape, y_train.shape)
print(X_test_csc.shape,y_test.shape)
```

```
After Encoding:
(87398, 30) (87398, 2)
(21850, 30) (21850, 2)
```

```
from sklearn.preprocessing import MinMaxScaler
min_max = MinMaxScaler()
# normalizer.fit(X_train['price'].values)
# this will rise an error Expected 2D array, got 1D array instead:
# array=[105.22 215.96  96.01 ... 368.98  80.53 709.67].
# Reshape your data either using
# array.reshape(-1, 1) if your data has a single feature
# array.reshape(1, -1)  if it contains a single sample.
min_max.fit(X_train['teacher_number_of_previously_posted_projects'].values.reshape(-1,1))

X_train_teacher_prev_min_max = min_max.transform(X_train['teacher_number_of_previously_posted_projects'].values.reshape(-1,1))
X_test_teacher_prev_min_max = min_max.transform(X_test['teacher_number_of_previously_posted_projects'].values.reshape(-1,1))



print("After vectorizations")
print(X_train_teacher_prev_min_max.shape, y_train.shape)
print(X_test_teacher_prev_min_max.shape, y_test.shape)
print("="*100)
```

```
After vectorizations
(87398, 1) (87398, 2)
(21850, 1) (21850, 2)
====================================================================================================
```

```
min_max = MinMaxScaler()
# normalizer.fit(X_train['price'].values)
# this will rise an error Expected 2D array, got 1D array instead:
# array=[105.22 215.96  96.01 ... 368.98  80.53 709.67].
# Reshape your data either using
# array.reshape(-1, 1) if your data has a single feature
# array.reshape(1, -1)  if it contains a single sample.
min_max.fit(X_train['price'].values.reshape(-1,1))

X_train_price_min_max = min_max.transform(X_train['price'].values.reshape(-1,1))
X_test_price_min_max= min_max.transform(X_test['price'].values.reshape(-1,1))



print("After vectorizations")
print(X_train_price_min_max.shape, y_train.shape)
print(X_test_price_min_max.shape, y_test.shape)
print("="*100)
```

```
After vectorizations
(87398, 1) (87398, 2)
(21850, 1) (21850, 2)
====================================================================================================
```

```
#stacking
from scipy.sparse import hstack
X_tr0 = hstack((X_train_state, X_train_tp, X_train_pgc, X_train_cc,X_train_csc, X_train_teacher_prev_min_max,X_train_price_min_max)).todense()
X_te0 = hstack((X_test_state, X_test_tp, X_test_pgc, X_test_cc,X_test_csc, X_test_teacher_prev_min_max,X_test_price_min_max)).todense()


X_tr0 = np.expand_dims(X_tr0,2)
X_te0 = np.expand_dims(X_te0,2)
```

```
X_tr0.shape
```

```
(87398, 101, 1)
```

```
essay = Input(shape= (500,), name = 'essay')
first= Embedding(5000, 300, input_length= 500, weights = [embedding_matrix],trainable =False)(essay)
lstm_essay = LSTM(32, return_sequences= True)(first)
flatten1 = Flatten()(lstm_essay)
other_than_text = Input(shape = (101,1), name = 'Input')
Conv1 = Conv1D(128, 3, activation='relu',kernel_initializer='he_normal')(other_than_text)
Conv2 = Conv1D(128,3, activation = 'relu', kernel_initializer='he_normal')(Conv1)
flatten2 = Flatten()(Conv2)
concat = concatenate([flatten1,flatten2])

model3 = Dense(32, activation='relu', kernel_initializer='he_normal')(concat)
model3 = Dropout(0.5)(model3)

model3= Dense(8, activation='relu', kernel_initializer='he_normal')(model3)
model3 = Dropout(0.1)(model3)

model3= Dense(4, activation='relu', kernel_initializer='he_normal')(model3)
Output = Dense(2, activation = 'softmax', name ='output')(model3)
adam = tf.keras.optimizers.Adam(learning_rate=0.0001)
model3 = Model(inputs= [essay,other_than_text], outputs = [Output])
model3.compile(loss = 'categorical_crossentropy', optimizer=adam, metrics = ['accuracy', auroc])

checkpoint_path = "/gdrive/My Drive/LSTM_Assignment/logs3/my_model_weights.hdf5"
checkpoint_dir = os.path.dirname(checkpoint_path)
checkpoint = tf.keras.callbacks.ModelCheckpoint(filepath='/gdrive/My Drive/LSTM_Assignment/logs3/my_model_weights{epoch}.hdf5',monitor ='val_loss',
log_dir="/gdrive/My Drive/LSTM_Assignment/logs3/fit/" + datetime.datetime.now().strftime("%Y%m%d-%H%M%S")
tensorboard_callback = tf.keras.callbacks.TensorBoard(log_dir=log_dir,histogram_freq=1, write_graph=True,write_grads=True)

training_input = [X_train_padded,X_tr0]
val_input = [X_test_padded,X_te0 ]
model3.fit(training_input,y_train,validation_data=(val_input,y_test), epochs = 10, callbacks=[tensorboard_callback,reduce_lr])
model3.save_weights('/gdrive/My Drive/LSTM_Assignment/logs3/my_model_weights{epoch}.hdf5')
model3.save('/gdrive/My Drive/LSTM_Assignment/logs3')
```

```
WARNING:tensorflow:`write_grads` will be ignored in TensorFlow 2.0 for the `TensorBoard` Callback.
Epoch 1/10
2732/2732 [==============================] - 752s 275ms/step - loss: 0.4498 - accuracy: 0.8442 - auroc: 0.5314 - val_loss: 0.4249 - val_accurac
Epoch 2/10
2732/2732 [==============================] - 757s 277ms/step - loss: 0.4301 - accuracy: 0.8490 - auroc: 0.5463 - val_loss: 0.4202 - val_accurac
Epoch 3/10
2732/2732 [==============================] - 757s 277ms/step - loss: 0.4219 - accuracy: 0.8490 - auroc: 0.5784 - val_loss: 0.4219 - val_accurac
Epoch 4/10
2732/2732 [==============================] - 761s 279ms/step - loss: 0.4191 - accuracy: 0.8490 - auroc: 0.5899 - val_loss: 0.4178 - val_accurac
Epoch 5/10
2732/2732 [==============================] - 763s 279ms/step - loss: 0.4178 - accuracy: 0.8490 - auroc: 0.5934 - val_loss: 0.4176 - val_accurac
Epoch 6/10
2732/2732 [==============================] - 758s 277ms/step - loss: 0.4171 - accuracy: 0.8490 - auroc: 0.6011 - val_loss: 0.4178 - val_accurac
Epoch 7/10
2732/2732 [==============================] - 758s 278ms/step - loss: 0.4168 - accuracy: 0.8490 - auroc: 0.6007 - val_loss: 0.4184 - val_accurac
Epoch 8/10
2732/2732 [==============================] - 760s 278ms/step - loss: 0.4158 - accuracy: 0.8490 - auroc: 0.6056 - val_loss: 0.4172 - val_accurac
Epoch 9/10
2732/2732 [==============================] - 764s 279ms/step - loss: 0.4155 - accuracy: 0.8490 - auroc: 0.6078 - val_loss: 0.4172 - val_accurac
Epoch 10/10
2732/2732 [==============================] - 762s 279ms/step - loss: 0.4155 - accuracy: 0.8490 - auroc: 0.6073 - val_loss: 0.4169 - val_accurac
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/tensorflow/python/training/tracking/tracking.py:111: Model.state_updates (from t
Instructions for updating:
This property should not be used in TensorFlow 2.0, as updates are applied automatically.
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/tensorflow/python/training/tracking/tracking.py:111: Layer.updates (from tensorf
Instructions for updating:
This property should not be used in TensorFlow 2.0, as updates are applied automatically.
INFO:tensorflow:Assets written to: /gdrive/My Drive/LSTM_Assignment/logs3/assets
```

```
model3.save('/gdrive/My Drive/LSTM_Assignment/logs3')
```

```
INFO:tensorflow:Assets written to: /gdrive/My Drive/LSTM_Assignment/logs3/assets
```

```
model3 = tf.keras.models.load_model('/gdrive/My Drive/LSTM_Assignment/logs3',custom_objects={'auroc':auroc})
```

```
checkpoint_path = "/gdrive/My Drive/LSTM_Assignment/logs3/my_model_weights.hdf5"
checkpoint_dir = os.path.dirname(checkpoint_path)
checkpoint = tf.keras.callbacks.ModelCheckpoint(filepath='/gdrive/My Drive/LSTM_Assignment/logs3/my_model_weights{epoch}.hdf5',monitor ='val_loss',
```

```
log_dir="/gdrive/My Drive/LSTM_Assignment/logs3/fit/" + datetime.datetime.now().strftime("%Y%m%d-%H%M%S")
tensorboard_callback = tf.keras.callbacks.TensorBoard(log_dir=log_dir,histogram_freq=1, write_graph=True,write_grads=True)


training_input = [X_train_padded,X_tr0]
val_input = [X_test_padded,X_te0 ]
model3.compile(loss = 'categorical_crossentropy', optimizer=tf.keras.optimizers.Adam(), metrics = ['accuracy', auroc])
model3.fit(training_input,y_train,validation_data=(val_input,y_test), epochs = 25, callbacks= [tensorboard_callback,reduce_lr])
model3.save('/gdrive/My Drive/LSTM_Assignment/logs3')
```

```
will be ignored in TensorFlow 2.0 for the `TensorBoard` Callback.

.........] - ETA: 0s - loss: 0.4966 - accuracy: 0.8125 - auroc: 0.5000WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/tensorflow/

op` instead.
=========] - 1058s 387ms/step - loss: 0.3956 - accuracy: 0.8477 - auroc: 0.7072 - val_loss: 0.3711 - val_accuracy: 0.8509 - val_auroc: 0.7496

=========] - 1049s 384ms/step - loss: 0.3694 - accuracy: 0.8495 - auroc: 0.7610 - val_loss: 0.3754 - val_accuracy: 0.8509 - val_auroc: 0.7498

=========] - 1049s 384ms/step - loss: 0.3509 - accuracy: 0.8527 - auroc: 0.7885 - val_loss: 0.3709 - val_accuracy: 0.8516 - val_auroc: 0.7442

=========] - 1068s 391ms/step - loss: 0.3240 - accuracy: 0.8661 - auroc: 0.8283 - val_loss: 0.3784 - val_accuracy: 0.8503 - val_auroc: 0.7373

.........] - ETA: 9:35 - loss: 0.2777 - accuracy: 0.8924 - auroc: 0.8768
---------------------------------------------
          Traceback (most recent call last)
in <module>()
ed,X_te0 ]
ategorical_crossentropy', optimizer=tf.keras.optimizers.Adam(), metrics = ['accuracy', auroc])
t,y_train,validation_data=(val_input,y_test), epochs = 25, callbacks= [tensorboard_callback,reduce_lr])
Drive/LSTM_Assignment/logs3')

- 8 frames
ckages/tensorflow/python/eager/execute.py in quick_execute(op_name, num_outputs, inputs, attrs, ctx, name)
ed()
.TFE_Py_Execute(ctx._handle, device_name, op_name,
                inputs, attrs, num_outputs)
usException as e:
```

```
model3.save('/gdrive/My Drive/LSTM_Assignment/logs3')
```

```
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/tensorflow/python/training/tracking/tracking.py:111: Model.state_updates (from t
Instructions for updating:
This property should not be used in TensorFlow 2.0, as updates are applied automatically.
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/tensorflow/python/training/tracking/tracking.py:111: Layer.updates (from tensorf
Instructions for updating:
This property should not be used in TensorFlow 2.0, as updates are applied automatically.
INFO:tensorflow:Assets written to: /gdrive/My Drive/LSTM_Assignment/logs3/assets
```

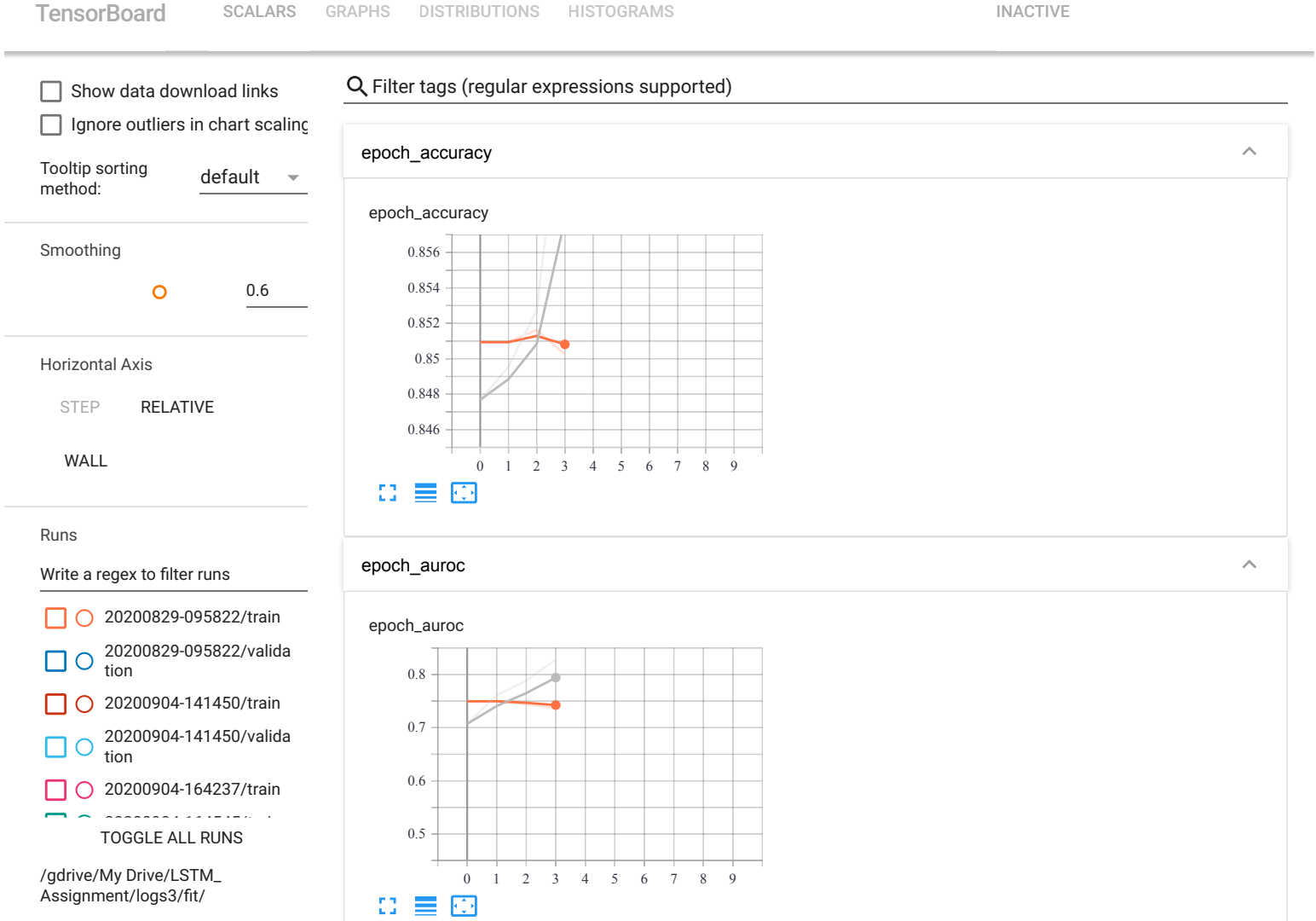
```
heckpoint_path = "/gdrive/My Drive/LSTM_Assignment/logs3/my_model_weights.hdf5"
heckpoint_dir = os.path.dirname(checkpoint_path)
heckpoint = tf.keras.callbacks.ModelCheckpoint(filepath='/gdrive/My Drive/LSTM_Assignment/logs3/my_model_weights{epoch}.hdf5',monitor ='val_loss',  v
og_dir="/gdrive/My Drive/LSTM_Assignment/logs3/fit/" + datetime.datetime.now().strftime("%Y%m%d-%H%M%S")
ensorboard_callback = tf.keras.callbacks.TensorBoard(log_dir=log_dir,histogram_freq=1, write_graph=True,write_grads=True)


raining_input = [X_train_padded,X_tr0]
al_input = [X_test_padded,X_te0 ]
odel3.compile(loss = 'categorical_crossentropy', optimizer=tf.keras.optimizers.Adam(), metrics = ['accuracy', auroc])
odel3.fit(training_input,y_train,validation_data=(val_input,y_test), epochs = 25, callbacks= [tensorboard_callback,reduce_lr])
odel3.save('/gdrive/My Drive/LSTM_Assignment/logs3')
```

```
WARNING:tensorflow:`write_grads` will be ignored in TensorFlow 2.0 for the `TensorBoard` Callback.
Epoch 1/25
   1/2732 [..............................] - ETA: 0s - loss: 0.4910 - accuracy: 0.7812 - auroc: 0.6154WARNING:tensorflow:From /usr/local/lib/py
Instructions for updating:
use `tf.profiler.experimental.stop` instead.
2732/2732 [==============================] - 1005s 368ms/step - loss: 0.3907 - accuracy: 0.8464 - auroc: 0.7067 - val_loss: 0.3728 - val_accura
Epoch 2/25
2732/2732 [==============================] - 1008s 369ms/step - loss: 0.3500 - accuracy: 0.8593 - auroc: 0.7848 - val_loss: 0.3622 - val_accura
Epoch 3/25
2732/2732 [==============================] - 1013s 371ms/step - loss: 0.3162 - accuracy: 0.8786 - auroc: 0.8310 - val_loss: 0.3813 - val_accura
Epoch 4/25
2732/2732 [==============================] - 1011s 370ms/step - loss: 0.2728 - accuracy: 0.8994 - auroc: 0.8732 - val_loss: 0.3909 - val_accura
Epoch 5/25
 802/2732 [=======>......................] - ETA: 11:23 - loss: 0.2200 - accuracy: 0.9228 - auroc: 0.9176
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
<ipython-input-44-5be40f2e21eb> in <module>()
      9 val_input = [X_test_padded,X_te0 ]
     10 model3.compile(loss = 'categorical_crossentropy', optimizer=tf.keras.optimizers.Adam(), metrics = ['accuracy', auroc])
---> 11 model3.fit(training_input,y_train,validation_data=(val_input,y_test), epochs = 25, callbacks= [tensorboard_callback,reduce_lr])
     12 model3.save('/gdrive/My Drive/LSTM_Assignment/logs3')
```

8 frames

```
%load_ext tensorboard
!kill 465
%tensorboard --logdir "/gdrive/My Drive/LSTM_Assignment/logs3/fit/"
```

```
/bin/bash: line 0: kill: (465) - No such process
```




```
model3.summary()
```

Conclusion:

Model:	AUC-ROC Score:	Accuracy:
1	0.7107	85.12%